Ajaxによる日本現存朝鮮古書印影写真画像データベース検索ツールの研究

18500079

平成１８年度～平成１９年度科学研究費補助金

（ 基盤研究（Ｃ） ）研究成果報告書

平成２０年３月

研究代表者　高　井　正　三

富山大学総合情報基盤センター教授

目　　　　次

１．はじめに

１．１ 研究目標

本研究「Ajax による日本現存朝鮮古書印影写真画像データベース検索ツールの研究」の目標は，分担者である元富山大学人文学部教授で現麗澤大学大学院言語教育研究科・教授の藤本幸夫が 30 年数年にわたって調査収集してきた日本現存朝鮮古書に関する書誌情報に関連して，撮影した原文画像・絵画上の著者印および蔵書印の印影画像と原文写真画像をディジタル化して，この印影画像を含む古文書の写真画像を検索し，検索結果の画像を表示するために，Google Maps 等で使用されている非同期型 Java Script 技術と XML（eXtensible Markup Language：拡張マークアップ言語＝利用者が自由にタグを定義でき，文書中の文字列に意味付けができる言語構造を持っているタグ言語で，メタ・データの定義に使用される.）技術を組み合わせた Ajax（「エイジャックス」と発音する．Asynchronous JavaScript + XML）技術を使用して，古文書画像を前後，左右に自在に移動させながら，かつ拡大・縮小を自在に行うことができる，究極の画像データベース検索ツールを研究し，実際にそのツールを開発するすることである．

また，Ajax 技術を用いた Unicode 漢字入力支援ツールと検索語類推入力支援ツールを開発して，DOKB データベース・システムのデータ追加・更新と検索の利便性を高めることを目標としている．

Web 上において CJK（Chinese Japanese Korean）用の IME(Input Method Editor)を実現させる技術も持ち合わせているので，印影画像のメタ・データベース（タイトルや著者などデータの内容に関する情報等をいう）を検索する場合，特定の国の IME が無くても，CJK 共通のローマ字入力，漢字変換が可能となる．この IME ツールと検索語推測入力支援ツール，Unicode 漢字入力支援ツールを合わせて研究・開発し，現行の DOKB データベース・システムに組み込むことができれば，世界の古文書データベースの標準化を進めることが可能となる．

１．２ 研究組織

研究代表者　：　高　井　正　三　（富山大学総合情報処理センター・教授）
研究分担者　：　藤　本　幸　夫　（元富山大学人文学部・教授，
　　現麗澤大学大学院言語教育研究科・教授）
（研究協力者　：　喜　多　啓　太　　富山大学大学院理工学教育部知能情報工学専攻(M2)
（研究協力者　：　米　田　恭　章　　富山大学大学院理工学教育部知能情報工学専攻(M2)

１．３ 交付決定額

交付決定額（配分額）　　（金額単位：千円）

区　　分	直 接 経 費	間 接 経 費	合　　計
平成１８年度	900	0	900
平成１９年度	900	270	1,170
総　　計	1,800	270	2,070

１．４ 研究発表

学会誌等

（論文１）

・高井正三，喜多啓太，米田恭章，Java フレームワークによる古文書データベース・システムの開発，学術情報処理研究，Vol.10，65-70，2006.

（論文２）

・高井正三，喜多啓太，米田恭章，Ajax 技法による日本現存朝鮮古書 DB 入力支援と画像 DB システムの開発，富山大学総合情報基盤センター広報，Vol.5，55-62，2008

１．５ 口頭発表

・高井正三，喜多啓太，米田恭章，Java フレームワークによる古文書データベース・システムの開発，第 10 回学術情報処理研究集会，2006.09.21，岩手大学工学部.

・喜多啓太，米田恭章，高井正三，Ajax による古文書画像閲覧システムの一提案，平成 18 年度電気関係学会北陸支部連合大会，講演論文集 E-81，2006.09.17，金沢工業大学.

・米田恭章，喜多啓太，高井正三，Ajax による古文書向け文字検索支援ツールの一提案，平成 18 年度電気関係学会北陸支部連合大会，講演論文集 E-82，2006.09.17，金沢工業大学.

・高井正三，喜多啓太，米田恭章，Ajax による古文書データベース検索語類推支援ツールの一提案，平成 18 年度電気関係学会北陸支部連合大会，講演論文集 E-83，2006.09.17，金沢工業大学.

１．６ 研究成果による工業所有権の出願・取得状況

特になし

２．研究目的と各年度の研究目標，研究計画及び成果

２．１ 研究目的

（１）研究の概要

日本現存朝鮮古書に関する書誌情報は，分担者である元富山大学人文学部教授で現麗澤大学大学院言語教育研究科・教授の藤本幸夫が 30 数年にわたって調査収集したものであり，28 項目に及ぶデータが既に 15,000 件以上，調査票に蓄積され，毎年数 100 件以上の新しいデータが追加されてきている．本研究の目的は，既にデータベース化された集部約 2,600 件のデータに，新規調査データを追加し，国際対応システムとして，Unicode 化，Java 化を実現し，Web 上でこの DOKB(Database of Old Korean Books)データベースの検索サービスするため，システムの改訂作業，データベースへのデータの追加・更新のための Unicode 漢字入力支援ツール，IME ツール，検索語類推支援ツールの開発を進めてきているところである[1].

この DOKB 開発作業と並行して，分担者の藤本幸夫が書誌情報をと共に撮影してきた原文画像フィルム約 600 本(約 22,000 枚)と写真 3,204 枚，絵画 1,306 枚及び文書 33 枚の画像を基に，日本現存する朝鮮古書の原文写真画像，フィルム画像の入力を進めてきており，収集されている写真画像の全部とフィルム画像をディジタル画像として TIFF 形式で入力し，そこから原文画像部分を切り出した画像を PNG 形式で保存し，さらに原文画像についてのメタ・データを同時に収録するすることとした.

本研究の課題の一つは，原文画像のデータベースを現行の DOKB と関連付けることであり，この原文画像とそのメタ・データによる「日本現存朝鮮古書原文画像データベース」の研究開発をすすめることにし，Ajax 技法による画像データベース検索システムの開発することにした.

Ajax 技術は, Google Maps 等で使用されている非同期型 Java Script 技術と XML（eXtensible Markup Language：拡張マークアップ言語＝利用者が自由にタグを定義でき，文書中の文字列に意味付けができる言語構造を持っているタグ言語で，メタ・データの定義に使用される.）を組み合わせた技術で, Ajax は「エイジャックス」と発音する[2] [3]. 通常は Asynchronous JavaScript + XML と略されている．この Ajax 技術を使用して，検索した古文書画像を前後，左右に自在に移動させながら，かつ拡大・縮小を自在に行って，表示することができれば，究極の古書原文画像データベース検索ツールとなると確信する．この「究極の古書原文画像データベース検索ツール」を研究し，実際にそのツールを開発するのが，本研究の最大の目的である。

また，本研究の課題の２つ目は，朝鮮古刊本の蔵書印や絵画の著者印の印影画像の切り出しを行って，印影画像のデータベースを作成し，Ajax 技法による検索結果の表示を行うことである．当初，刻手の版心部の印（しるし）と著者印影画像を同一視していたため，本研究課題を「……古書印影写真画像データベース検索ツールの研究」としていたが，刻手には印影はなく，絵画などの著者印及び古書（古文書）所有者の蔵書印の印影のみを対象とすべきであった．このため，刻手については，分担者の藤本が書誌情報とともに収集してきた刻手名データのマスターをデータベース化し，現行の DOKB と連動させることとし，印影画像の切り出しとデータベース化は，印影の切り出しの問題点を解決してから実施することとした.

更に, Ajax 技術をもちいた Unicode 漢字入力支援ツールを開発し，漢字の偏や旁を指定して，全体の画数から該当する Unicode 漢字を表示して，簡単に選択入力するツールを開発するとともに，現行 DOKB 検索システムへの実装を試みた.

Ajax 技術を用いた最近の画期的なサービスは Google Suggest 等に見られる検索語類推支援で

あり，このサービスにヒントを得て，DOKB の書名や撰者名の検索語が有限であり，DB のマスター・データからすべてのキーワードを抽出して DB 化すれば，古書書誌情報検索においても検索語類推支援ツール開発の可能性が充分あり，利用者が検索語の一文字をインクレメンタルに入力する度に，キーワードを類推表示すれば，無駄な検索語の入力を排除し，検索語入力の効率化と正確さを支援できる．ローマ字入力ならば Unicode 漢字入力の国際化に対応できるので，この検索語類推支援ツールの開発を第 4 番目の課題とすることにした．

Ajax 技術はまた，Web 上において CJK (Chinese Japanese Korean) 用の IME(Input Method Editor)を実現させる技術も持ち合わせているので，現行 DOKB 検索システムや印影画像のメタ・データベース（タイトルや著者などデータの内容に関する情報等をいう）を検索する場合，特定の国の IME が無くても，CJK 共通のローマ字入力，漢字変換が可能となる．汎用 IME ツールは，前述の検索語類推支援ツールと合わせて研究・開発して行くこととした．

本研究により，日本が世界に誇ることのできる Web 型 DOKB データベースを提供することができ，公開を待ち望む世界の朝鮮及び朝鮮本研究者に対し，研究能率の向上に貢献できるようになるのもと確信している．

（２）研究経過

筆者らは，平成 6 年度から，分担者の藤本氏が調査・収集した日本現存朝鮮古書の書誌情報を整理して，パーソナル・コンピューターのエディタを使用して，調査項目にタグを付け，常用漢字を使用しながらデータの入力を開始した．平成１７年まで入力した書誌情報は集部の約 3,000 件に達した．

これらの書誌情報は国際対応化を目指した DOKB とするため，Unicode に変換して，最終的に約 2,600 件のデータを 2006 年 12 月 4 日に Unicode 版 DOKB として公開した．

なお，分担者の藤本氏は平成 17 年度に科学研究費の「研究成果公開促進経費」の予算がついたので，更に 400 件弱の書誌情報を追加して，平成 18(2006)年 2 月 28 日に，「日本現存研究古刊本研究　集部」を京都大学学術出版会から刊行し，集部 3,000 件の書誌情報・所蔵情報を納めた．だだし，このデータにより，現行の DOKB と「日本現存研究古刊本研究　集部」に添付されている CD データの台帳と整合性がとれなくなったので, CD データ台帳の Shift-JIS を改めて Unicode 化する作業を開始せざるを得なくなった．

本研究では，従って，１）Ajax 技法による日本現存朝鮮古書原文画像のデータベース検索システムの開発，２）Ajax 技法による Unicode 漢字入力支援ツールの開発と，集部 3,000 件の書誌情報・所蔵情報の Unicode 化作業，３）Ajax 技法による検索語類推支援ツール，Unicode の CJK 用 IME ツールの開発，４）Ajax 技法による朝鮮古刊本印影画像のデータベースの開発，５）藤本氏は収集した刻手のデータベース開発，の 5 つを目標として研究を進めることとした．最終的には，印影画像と古文書の写真画像，刻手のデータを現行 DOKB データベース検索システムに組み込めるようＤＢシステムを再編成し, Ajax による画像データベース検索ツールを研究開発することが目的である．

（３）研究の学術的な特徴

本研究の特色は，かつて日本が朝鮮から収集してきた数多くの朝鮮古書の種類とその内容を収録した書誌情報，所蔵情報など，貴重な調査データを収録したデータベースを世界の研究者に公開するための，一連の研究であり，国内外に例を見ない調査・研究の成果を提供する．

公開を待ち望む韓国をはじめ，全世界の朝鮮及び朝鮮本の研究者に，印影画像と古書原文画像データ，さらには刻手のデータを追加すれば，古文書の発掘とその書物の同定が極めて簡単になり，古文書データベースでは不可欠のものとなる．

平成 18 年 2 月 28 日に刊行した「日本現存研究古刊本研究　集部」の冊子体目録だけでは，古文書データベースによる新しい古文書の発掘が難しく，印影画像及び古書原文画像，刻手などの日本現存朝鮮古書の画像データベース，刻手のデータベースが活用されることによって，世界に居ながら古文書の発掘，同定及び内容に関する研究ができるようになるものと確信する．

（4）独創的な点

Ajax 技術が Web ブラウザの画面遷移を伴わず，自由自在に画像移動，回転，縮小拡大ができることを証明し，隣接する画像エリアはインターネットを通じて，利用者が右側を見たければ，その操作を予測して，非同期に画像データを取りに行き(Fetch)，現在の表示されている画像と境界を合わせながら，次の操作を連続的に実施できるようになった．この Ajax 技法を筆者らが構築している DOKB の拡張機能として提供していこうとするものである．

CJK 漢字の入力についても，Google Suggest の様に，検索語を入力していくと，１字毎に検索語を推測していく機能，即ち検索語の候補を表示していく機能を盛り込んだ CJK 用 IME を開発すれば，専門以外の人でも，これらの学術情報データベースに簡単にアクセスできるようになる．

Ajax 技術は，枯れた技法だと言われるが，その応用範囲は実に広く，筆者らは画期的なデータベース検索ツールの時代に突入することを確信している．

（5）予想される結果と意義

本研究は，Java をベースにして開発され，Unicode を使用した国際対応の日本現存朝鮮古書の書誌及び画像，刻手データベースであるが，Ajax 技法を用いた画像検索ツールと Unicode を用いた国際対応の古文書データベース一般に，画期的な変革をもたらし，これからの古文書データベースの汎用的なスタイルになると予想される．非同期型先読みで，画面遷移のない画像検索技術としては，今後標準化される技法と考えられる．

（6）国内外の位置付け

CJK 用に，中国に対しては Pinyin 方式，韓国に対しては McCune Reischauer 方式，日本では Hepburn 方式のローマ字入力により，CJK-Xterm エミュレータのように，１字入力毎に検索語を推測していく機能を有しているので，各国語専用の IME（Input Method Editor:入力方式編集プログラム＝かな漢字変換などの事前処理プログラム）が不要で，Unicode 漢字が入力でき，かつ，画像検索の操作性は Google Maps と同じようになっているので，データベースの内容が国際性を持てば，このデータベース検索ツールの可能性は，漢籍古文書に限定されることがなくなる．

２．２ 研究計画・方法

２．２．１　研究の目標

本研究の目標は，日本現存朝鮮古書の印影画像と古書原文写真画像，刻手のデータをデータベースに組み込込めるよう DB システムを再編成し, Ajax 技法による画像データベース検索システム，刻手データベース検索システム，印影画像データベース検索システムを，２年間で研究開発することである．

また，Ajax 技法による Unicode 漢字入力支援ツール，検索語類推支援ツール，CJK 用 IME ツールなどの入力支援ツールを，データベース検索システムの開発と合わせて，研究開発することである．

２．２．２　研究計画・方法

この目標を達成するため，以下の実施手順にて研究を進める．

（１）刻手データの整備，印影画像と古文書の写真画像の入力，メタ・データの整備
（２）Ajax(Asynchronous JavaScript + XML)技法の研究
（３）Ajax 技法による画像データベース検索システムの開発・構築・テスト
（４）日本現存朝鮮古書データベース DOKB への画像データ追加による DB の再編成
（５）Ajax 技法による Unicode 漢字入力支援ツールの開発と実装
（６）集部 3,000 件の Shift-JIS マスター・データの Unicode への変換作業
（７）Ajax による検索語類推支援ツール，CJK 用 IME ツールの研究，開発
（８）画像データベース，刻手データベースと検索ツール，検索語推測システムの統合・テスト
（９）インターネット上に画像データベース・システムを公開テスト
（10）国内，海外の研究者・利用者による検索システムのレビュー
（11）学会，研究会にて成果を発表，正式公開，報告書作成

Web プラット・フォーム上で検索するシステムとしては既にいくつかの開発を試み，旧字体漢字と朝鮮固有外字を Unicode 化し，サービスしている現行 DOKB システムの他，開発対象としては，朝鮮古書原文画像データベース検索システム，刻手データベース検索システム，日本現存朝鮮古書印影画像データベース検索システムがある．また，各種検索支援ツールとしては，Ajax 技法による Unicode 漢字入力支援ツール，Ajax による検索語類推支援ツール，CJK 用 IME ツールの研究・開発を進める．

２．３ 平成１８年度の研究計画と方法

平成１８年度は，上記目標達成手順の（１）～（７）を，以下の通り具体的に研究を進める．

２．３．１　研究計画と方法

（１）刻手データの整備，印影画像と古文書の写真画像の入力，メタ・データの整備

１）刻手データは，分担者の藤本が平成 17・18 年度の科研費補助金で構築・整備し，高井が索引を作成した「朝鮮朝刊本刻手名集（第二版)」平成 19 年刊行予定のマスター・データを整備する．

２）印影画像と古文書の写真画像のディジタル化は，スキャナーから画像を 1200dpi（１インチ当たり 1,200 ドットの解像度）以上の解像度で読み込み，TIFF（Tagged Image File Format）形式で一旦保存し，原文部分を切り出して，JPEG(Joint Photograph Experts Group)形式または画像データ形式の可逆圧縮性を保証している次世代画像書式である PNG(Portable Network Graphic)形式ファイルとして保存し，更に，メタ・データを付与する．

３）Ajax 技法による画像 DB 検索システムの試作品に対応する解像度を決める．

４）既に入力されている画像データの解像度と書式を統一する．

５）解像度の低いデータはスキャンし直す．

６）メタ・データの再確認をする．

７）印影画像は，１文献につきすべての押印された印影を入力する．

８）古文書写真データは１文献につき１枚以上，重要な部分は無制限とする．

（２）Ajax(Asynchronous JavaScript + XML)技法の研究

１）Ajax, XML，JavaScript に関する知識を吸収し，技法を習得する．

２）Ajax 技法を使用している Google Maps，Google Earth などのシステムを分析し，その構造を解析する．

３）Ajax 技法による検索語類推システムを分析し，その構造を解析する．

４）Ajax を使った，簡単な図形，写真検索システムを試作する．

５）Ajax を使った，簡単な朝鮮古書データベース用の検索語類推システムを試作する．

（３）Ajax 技法による画像データベース検索システムの開発・構築・テスト

１）Ajax 技法による画像検索システムの要件定義ツールの設計

２）Ajax 技法による画像検索ツールの設計

- 機能分割とモジュール設計
- クラス，メソッドの設計
- フレームワーク分析，設計
- クラス・ライブラリの活用設計

３）非同期型 JavaScript, XML による画像検索ツールの開発

- モジュールのコード開発
- モジュール毎の単体テスト

４）JavaScript, XML のコード開発

５）画像検索ツールのテスト

（４）日本現存朝鮮古書データベース DOKB への画像データ追加による DB の再編成

１）既存 DOKB データのアンロード

２）データベース再編成のためのテーブル定義

３）既存データのロード

４）画像データ，メタ・データによるデータベースの更新

５）対話型データベースの検索テスト

（５）Ajax 技法による Unicode 漢字入力支援ツールの開発と実装

１）書誌情報検索入力画面の再設計・改良

２）Ajax による Unicode 漢字入力支援ツールの分析

３）Ajax による Unicode 漢字入力支援ツールの設計

４）Unicode ストローク・データの準備とデータベースの作成

５）Unicode 表示用フォントの整備

６）Unicode 文字入力支援ツールのコード開発・テスト

７）現行 DOKB システムへの実装

（６）集部 3,000 件の Shift-JIS マスター・データの Unicode への変換作業

１）データの準備

２）下駄文字「**=**」の Unicode への対応の確認資料の準備

３）Unicode テキスト・エディタ「秀丸」の準備

４）下駄文字「**=**」の Unicode へ変換作業

５）変換記録の記帳

（７）Ajax による検索語類推支援ツール，CJK 用 IME ツールの研究，開発

１）書誌情報検索入力検索語類推画面の改訂

２）Ajax による検索語類推支援ツールの分析

３）Ajax による検索語類推支援ツールの設計

４）試作品の作成・テスト

５）検索語データの調査，マスターからの切り出し準備

６）検索語の抽出，XML データベースの作成

７）現行 DOKB システムへの実装

２．３．２　経費と研究計画の関連性

（１）謝金

１）印影，古文書写真画像の入力，メタ・データの付与する人の雇用経費.

２）Ajax 技法を使用している Google Maps，Google Earth などのシステムを分析し，その構造を解析する補助者の雇用経費.

３）Ajax 技法による検索語類推システムを分析し，その構造を解析する補助者の雇用経費.

４）検索語調査とそれを収集する人の雇用経費.

５）非同期型 JavaScript, XML による画像検索ツールの開発補助者の雇用経費.

・JavaScript, XML の各コード開発
・HTML, SQL，Java のコード開発
・情報検索画面の作成

（2）外国旅費

1）画像データベースを研究，開発，サービスしている海外の研究者，会社を周り，システム開発の要件と技法を調査し，システム構築設計に関するレビューを受ける．

2）研究の成果を国際会議で発表する．

（3）国内旅費

1）画像データベースを研究，開発，サービスしている国内の研究者，会社を周り，システム開発の要件と技法を調査し，システム構築設計に関するレビューを受ける．

2）研究・開発の成果を学会，研究会で発表する．

（4）その他の経費

1）国際会議や学会の参加登録費．

（5）消耗品

1）データベースのデータ・バックアップ用保存ファイルの媒体購入費．

２．３．３　分担者

藤本幸夫氏には，既入力画像データとメタ・データの提供を受け，画像検索システム全体のユーザ・インターフェースを検討してもらい，種々の提言をお願いする．

２．４ 平成１８年度の成果

（１）刻手データの整備，印影画像と古文書の写真画像の入力，メタ・データの整備

・刻手データについては，研究分担者の藤本が平成 17・18 年度の科研費補助金で構築・整備し，高井が索引を作成した「朝鮮朝刊本刻手名集（第二版）」平成 19 年刊行予定のマスター・データ 1,356 件，約 3.3MB の文字コードを Shift-JIS から Unicode に変換し，DB 化する準備をした．
・印影画像と古文書の写真画像の入力については，当初の入力方法に従い，古文書の写真画像は 4,561 枚を，フィルム画像は 11,135 枚を入力し，そのメタ・データを収録してきた．写真画像は 1200dpi で，tiff 形式で収録し，古文書部分の切り出しを行い，png 形式で保存した．35 ㎜フィルム画像は，4,000dpi で収録し，オリジナルは tiff 形式，切り出し画像は png 形式で保存した．印影画像の切り出しは，原文文字との重なりが多く，この問題の解決策を解決してから実施することとした．
・メタ・データについては，古文書の写真画像は 4,561 枚を，フィルム画像は 11,135 枚の分を Excel 形式で作成した．

（２）Ajax(Asynchronous JavaScript + XML)技法の研究

Ajax 技法については，研究書籍を蒐集し，システムの試作とテストを行い，平成 18 年度の電気関係学会北陸支部連合大会で，試作品の成果を発表した（P.2 参照）．

１）Ajax による古文書画像閲覧システム
２）Ajax による古文書向け文字検索支援ツール
３）Ajax による古文書データベース検索語類推支援ツール

（３）Ajax 技法による画像データベース検索システムの開発・構築・テスト

Ajax 技法による画像検索システムの要件定義ツールの設計，画像検索ツールの設計として，・機能分割とモジュール設計，クラス，メソッドの設計，フレームワーク分析・設計，クラス・ライブラリの活用設計を行い，非同期型 JavaScript, XML による画像検索ツールの開発を行い，「Ajax による古文書画像閲覧システム」を試作し，前述のとおり発表した．

このシステムでは，古文書画像 1 枚について，Ajax 技法を用いた上下左右の画面遷移なし移動と，拡大縮小を行えるようにスライダーを設けた．

（４）日本現存朝鮮古書データベース DOKB への画像データ追加による DB の再編成

画像データベースのテーブル定義を中心とするシステム設計を行った．

（５）Ajax 技法による Unicode 漢字入力支援ツールの開発と実装

書誌情報検索入力画面の再設計・改良，Ajax による Unicode 漢字入力支援ツールの分析，設計を行い，Unicode Consortium の Unihan Databse から Unicode ストローク・データ，コード・ポイントなどのデータを取り込んでデータベースを作成，ベトナムで作成された，提供されている Unicode フォント HAN NOM A，HAN NOM B を Download して，ツール開発の準備をし，）Unicode 漢字入力支援ツールのコード開発・テストを行って，現行 DOKB システムへの実装を行った．このツールは（２）のとおり，開発結果を学会発表した．

（６）集部 3,000 件の Shift-JIS マスター・データの Unicode への変換作業

研究分担者の藤本から，京都大学学術出版会から刊行した「日本現存研究古刊本研究　集部」の完全データを入手してもらって，原稿の追加削除などを調べることにしたが，使用した外字や朝鮮固有文字約 1,000 字は全て Shift-JIS の下駄文字「=」に化けたので，これを Unicode 変換する作業を開始し，現行の DOKB マスター・データと「日本現存研究古刊本研究　集部」書籍，今までの外字変換帳により，Unicode 対応テキスト・エディタ「秀丸」を使用して，下駄文字「=」の Unicode へ変換作業を行った．変換記録はすべて記帳し，約 50%のマスター・データの「日本現存研究古刊本研究　集部」下駄文字「=」を変換した．

（７）Ajax による検索語類推支援ツール，CJK 用 IME ツールの研究，開発

Ajax による検索語類推支援ツールについては，DOKB マスター・データから一部の書名，撰者名の検索語を抽出し，XML データベースを作成した．JavaScript と非同期 HTTP リクエストの使い方をテストしながら，試作品を作成し，最終的にはかな漢字変換を経由しないで，直接 Hepburn 式ローマ字を入力することによる，検索語類推支援ツールの試作品を作成した．この結果は（２）のとおり開発結果を学会発表した．

なお，現行 DOKB システムへの実装は，Unicode による 3,000 件のマスター・データが完成するのを待つこととした．

以上のように，Ajax 技術による 3 つのシステム／ツールについては，試作品を作成し，開発の途中成果については，平成 18 年度の電気関係学会北陸支部連合大会に 3 件の発表を行った．

また，Java 版の DOKB 開発結果については，「Java フレームワークによる古文書データベース・システムの開発」として，研究成果を 2006 年度の学術情報処理研究会で発表した．更に，米国で開催の第 30 回 Unicode 国際会議に参加して，ＤＢシステムのレビューを行った．

２．５ 平成１９年度以降の研究計画と方法

平成１９年度は，目標達成手順のうち，（８）～（11）を中心に，平成 18 年度の未完成課題とともに，以下の通り具体的に研究を進める．

２．５．１　研究計画と方法

（１）画像データベース，刻手データベースと検索ツール，検索語推測システムの統合・テスト

１）画像データベース・システムを完成する．
２）刻手データベース・システムを完成する．
３）画像検索ツールを完成する．
４）検索語推測システムを完成する．
５）上記２つのシステムと 2 つのツールを統合するコードを開発する．
６）システムの統合テストを実施し，デバッグする．

（２）インターネット上に画像データベース・システムを公開テスト

１）インターネット上に画像データベース・システム移植し，テストする．
２）ユーザ ID，パスワードによる画像データベース・システムを公開テストにかける．
３）公開テストで見つかった不具合を修正する．

（３）国内，海外の研究者・利用者による検索システムのレビュー

１）国内の研究者に画像データベース・システムのレビューを受ける．
２）海外の研究者に検索語推測システムと合わせてレビューを受ける．
３）レビューを受けた結果，修正すべきところを修正する．

（４）学会，研究会にて成果を発表，正式公開，報告書作成

１）ＩＤ・パスワードなしで画像データベース・システムを公開する．
２）研究・開発の成果を学会，研究会で発表する．
３）研究・開発の成果を報告書としてまとめる．

（５）平成 18 年度の未完成課題

以下のうち，未完の課題．
１）刻手データの整備，印影画像と古文書の写真画像の入力，メタ・データの整備
２）Ajax(Asynchronous JavaScript + XML)技法の研究
３）Ajax 技法による画像データベース検索システムの開発・構築・テスト
４）日本現存朝鮮古書データベース DOKB への画像データ追加による DB の再編成
５）Ajax 技法による Unicode 漢字入力支援ツールの開発と実装
６）集部 3,000 件の Shift-JIS マスター・データの Unicode への変換作業
７）Ajax による検索語類推支援ツール，CJK 用 IME ツールの研究，開発

２．５．２　経費と研究計画の関連性

（１）謝金

1）刻手データ，印影画像，古文書写真画像の入力，メタ・データの付与する人の雇用経費.
2）Ajax 技法を使用して，画像データベース・システムの開発補助者の雇用経費.
3）Ajax 技法による検索語類推システムの開発補助者の雇用経費.
4）検索語のテスト，再検討，修正する人の雇用経費.
5）非同期型 JavaScript, XML による画像検索ツールの開発補助者の雇用経費.
・JavaScript, XML の各コード開発
・HTML, SQL，Java のコード開発
・検索の遷移画面，検索結果の PDF(Portable Document Format)形式表示画面の作成
（2）外国旅費
1）朝鮮関係古文書及び画像データベースを研究，開発，サービスしている韓国の研究者，研究機関及び情報サービス機関を周り，画像検索システムの使い勝手に関するレビューを受ける.
2）研究の成果を海外の学会，国際会議等で発表する.
（3）国内旅費
1）古文書及び画像データベースを研究，開発，サービスしている国内の研究者，研究機関及び情報サービス会社を周り，画像検索システムの使い勝手に関するレビューを受ける.
2）研究・開発の成果を学会，研究会で発表する.
（4）その他の経費
1）国際会議や学会の参加登録費.
2）研究成果報告書の印刷経費
（5）消耗品
1）データベースのデータ・バックアップ用保存ファイルの媒体購入費.

2．5．3　分担者

分担者の藤本幸夫氏には，追加入力された画像データとメタ・データの提供を受け，画像検索システム全体のレビュー，刻手データベース・システムのレビュー，提言をお願いする.

2．5．4　平成 19 年度研究計画の修正

印影画像の切り出しと Ajax 用の画像の細分化の実施について，印影は原文画像との重なりが多く，入力した原文画像からの切り出しが困難であること．絵画に多く押印されている印影については，逆に日本現存朝鮮古書との関連が困難なことが判明しため，原文画像から印影切り出しの具体的方法が決まるまで，印影画像の切り出しを中止し，先ず，分担者の藤本が蓄積してきた刻手名のデータベースを作成することとした.

２．６ 平成１９年度の成果

（１）画像データベース，刻手データベースと検索ツール，検索語推測システムの統合・テスト

・画像データベース・システムは古書原文画像を対象に，引き続き Java Framework を使用してコードの作成を行ってきた．ただし，完成するには至っていないが，システム開発工数の約７割を終了した．

・刻手データベース・システムについては，JavaEE を使って開発をすすめ，Application Server の Grass Fish を主体にして, JavaFramework JSF (Java Server Faces), EJB (Enterprise Java Beans)，JPA（Java Perrsistence API）を使用して開発を終了した．

・画像検索ツールは画像データベース・システムと並行して開発中である．

・検索語推測システム＝検索語類推緒支援ツールの開発は，ツールは完成しているが，3,000 件のマスター・データの Unicode 化完了を待って XML データベースを作成することとした..

・Ajax 技法による Unicode 漢字入力支援ツールは完成したので現行の日本現存朝鮮古書データベース・システム（DOKB）に組み込み，実際運用を開始した．

・現行 DOKB システムは，冊子目録の作成によってデータが S-JIS に逆戻りしたため，UTF-８への変換を 100%完了し，再編成を待つ段階である．

以上，上記２つのシステムと 2 つのツールを統合は，現在も進行中である．

（２）インターネット上に画像データベース・システムを公開テスト

・インターネット上に画像データベース・システムの公開は，開発途上にあり，完成は次年度以降になる．

（３）国内，海外の研究者・利用者による検索システムのレビュー

・Ajax 技法の有効性を探るため，第 31 回国際化 Unicode 会議に出席し，DOKB システムに関するレビューと意見交換を行った．また，平成 19 年 12 月 13 日（木）～14 日（金）に，京大会館で開催された「じんもんこん２００７（人文科学とコンピュータ・シンポジウム)」と，12 月 22 日(土)二松学舎大学九段キャンパスで開催された漢字情報処理研究会第 10 回大会に出席し，画像データベース・システム，検索語類推支援ツールについて，レビューを受けた．

（４）学会，研究会にて成果を発表，正式公開，報告書作成

・刻手データベースは，分担者の意向もあり，ＩＤ・パスワード付で公開している．

・研究成果は「Ajax 技法による日本現存朝鮮古書 DB 入力支援と画像 DB システムの開発」という論文を富山大学総合情報基盤センター広報 Vol.5 に公表した．

３．システム開発結果

３．１　Ajax 技法による Unicode 漢字入力支援ツールの開発・実装

３．１．１　Unicode の必要性と現在の使用環境

事の発端は，日本現存朝鮮古書（集部）の書誌情報の冊子体目録と藤本氏の研究成果成果である項目「研覈」を記載した「日本現存朝鮮本研究　集部」を刊行するにあたって，それまで蓄積し，Unicode に変換してデータベース化してきた，DB のマスター・データに新たに約 400 件のデータが追加され，同時に既存のデータに更新・加除があり，Unicode 文字＝固有外字／朝鮮固有文字は印刷メーカーで置換あるいは新規に作成され，固有コードが付けられて印刷された．そのため，**最終編集されたマスター・データが筆者に提供されたときには，総て Shift-JIS コードに逆戻りし，Unicode 文字で置換した固有外字／朝鮮固有文字は総て Shift-JIS の下駄文字「＝」に置換されてしまった．**最初の@99999 形式になっていれば，プログラムで変化することもできたが，**印刷業者が未だに Unicode を使用していないことが，事の発端であり，我が国の文化の後退をもたらしている**と言っても，過言ではないと思う．

この Shift-JIS の下駄文字「＝」を Unicode に戻す変換作業をスムースにするため，このツール「Unicode 漢字入力支援ツール」が作成されたといってよい．結果的には DOKB データベース検索に利用すればより効率的であるので，DOKB の漢字入力の標準インターフェースとしてこのツールを実装することにした．

DOKB システムでは，古書の書誌情報を正確に提供するため，Unicode5.0 による国際符号化文字集合 UCS（Universal Character Set）Transformation Set の 8 ビット符号化形式である UTF-8（ユー・ティ・エフ・エイト）を使用している．現在，Unicode では BMP（Basic Multilingual Plane＝基本多言語面）に加え，コードの先頭が２XXXX で始まる CJK 統合漢字拡張 B （CJK Unified Ideographs Extension B）のコード U+20000～U+2A6DF が公開され，それを表示できる Firefox などの Web Browser と，Viet Nam で開発された True Type Fonts HAN NOM A, HAN NOM B などが使用できるようになって，我々が使用している拡張漢字全 1,079 字のうち約 97%が表示できるようになった．

その他の使用環境としては，テキスト・エディタの秀丸や日本語ワードプロセッサ一太郎 2007 などで，UTF-8 文字パレットが使用できるようになった．

しかしながら，その文字入力はかなり大変で，IME 上で一々探すのに時間を要し，この入力を如何に迅速にするかを考えて，我々は Unicode 文字の入力支援ツールを開発し，DOKB 検索システムに実装した．

３．１．２　Unicode 漢字入力支援システム

DOKB 検索システムの画面上で，画面の遷移無しに Unicode 文字 を入力するには Ajax ツールを使って，通常の IME の様に漢字の部首と画数で漢字を表示・検索し，該当文字をクリックするだけで，検索語の入力域に文字が入力されるように設計した．開発手順は以下の通りである．

１）既定の 214 の部首ごとにコード・ポイントを収めた XML ファイルを作成する．

２）指定した部首の XML ファイルを非同期的に取得し，画数情報を解析する．

３）得られた画数情報から指定された画数の文字をブラウザに表示させる．

３．１．３　部首 XML ファイル

部首用の XML ファイルは，214 の部首毎に用意しており，例えば部首「丨」の部であればファイル名「radical2.xml」として，コード・ポイント・ファイルを図 3.1.1 のように定義してい

る.

radical_2.xml Page 1

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <radicalData>
 3 <radical_name>line</radical_name>
 4 <radical_num>2</radical_num>
 5 <stroke_num>0</stroke_num>
 6 <stroke_num>1</stroke_num>
 7 <stroke_num>2</stroke_num>
 8 <stroke_num>3</stroke_num>
 9 <stroke_num>4</stroke_num>
10 <stroke_num>5</stroke_num>
11 <stroke_num>6</stroke_num>
12 <stroke_num>7</stroke_num>
13 <stroke_num>8</stroke_num>
14 <stroke_num>9</stroke_num>
15 <stroke_num>10</stroke_num>
16 <stroke_num>11</stroke_num>
17 <stroke_num>13</stroke_num>
18 <stroke_num>14</stroke_num>
19 <stroke_num>17</stroke_num>
20 <stroke_num>18</stroke_num>
21 <stroke_num>21</stroke_num>
22 <stroke_0>
23 <codepoint>4E28</codepoint>
24 </stroke_0>
25 <stroke_1>
26 <codepoint>4E29</codepoint>
27 <codepoint>20061</codepoint>
28 <codepoint>20062</codepoint>
29 </stroke_1>
30 <stroke_2>
31 <codepoint>4E2A</codepoint>
32 <codepoint>4E2B</codepoint>
33 <codepoint>3403</codepoint>
34 <codepoint>3404</codepoint>
35 </stroke_2>
36 <stroke_3>
37 <codepoint>4E2D</codepoint>
38 <codepoint>4E2E</codepoint>
39 <codepoint>4E2F</codepoint>
40 <codepoint>4E30</codepoint>
41 <codepoint>4E66</codepoint>
42 <codepoint>20063</codepoint>
43 </stroke_3>
44 <stroke_4>
45 <codepoint>4E31</codepoint>
46 <codepoint>20065</codepoint>
47 </stroke_4>
48 <stroke_5>
49 <codepoint>20066</codepoint>
50 </stroke_5>
51 <stroke_6>
52 <codepoint>4E32</codepoint>
53 <codepoint>F905</codepoint>
54 <codepoint>20067</codepoint>
55 <codepoint>20068</codepoint>
56 </stroke_6>
57 <stroke_7>
58 <codepoint>4E33</codepoint>
59 <codepoint>20069</codepoint>
60 <codepoint>2006A</codepoint>
61 <codepoint>2006B</codepoint>
62 <codepoint>2006C</codepoint>
63 </stroke_7>
```

radical_2.xml Page 2

```
64 <stroke_8>
65 <codepoint>4E34</codepoint>
66 <codepoint>2006D</codepoint>
67 <codepoint>2006E</codepoint>
68 <codepoint>2006F</codepoint>
69 </stroke_8>
70 <stroke_9>
71 <codepoint>4E35</codepoint>
72 <codepoint>20070</codepoint>
73 <codepoint>20071</codepoint>
74 </stroke_9>
75 <stroke_10>
76 <codepoint>20073</codepoint>
77 </stroke_10>
78 <stroke_11>
79 <codepoint>20074</codepoint>
80 <codepoint>20075</codepoint>
81 </stroke_11>
82 <stroke_13>
83 <codepoint>20076</codepoint>
84 </stroke_13>
85 <stroke_14>
86 <codepoint>20078</codepoint>
87 </stroke_14>
88 <stroke_17>
89 <codepoint>20079</codepoint>
90 </stroke_17>
91 <stroke_18>
92 <codepoint>2007A</codepoint>
93 </stroke_18>
94 <stroke_21>
95 <codepoint>2007B</codepoint>
96 </stroke_21>
97 </radicalData>
```

図 3.1.1 部首毎の CodePoint 用 XML ファイル定義

３．１．４　Unicode 漢字入力支援ツール

Ajax 版 Unicode 漢字入力支援ツールは以下の様に試作し（図 3.1.2），MS-IME 2003（図 3.1.3），MS-IME 2007（図 3.1.4）と比較してみれと，Unicode 第２面補助的表意文字面（Supplementary Ideographic Plane）の CJK 統合漢字拡張 B まで支援する有効さと，使い勝手の良さが理解できる．

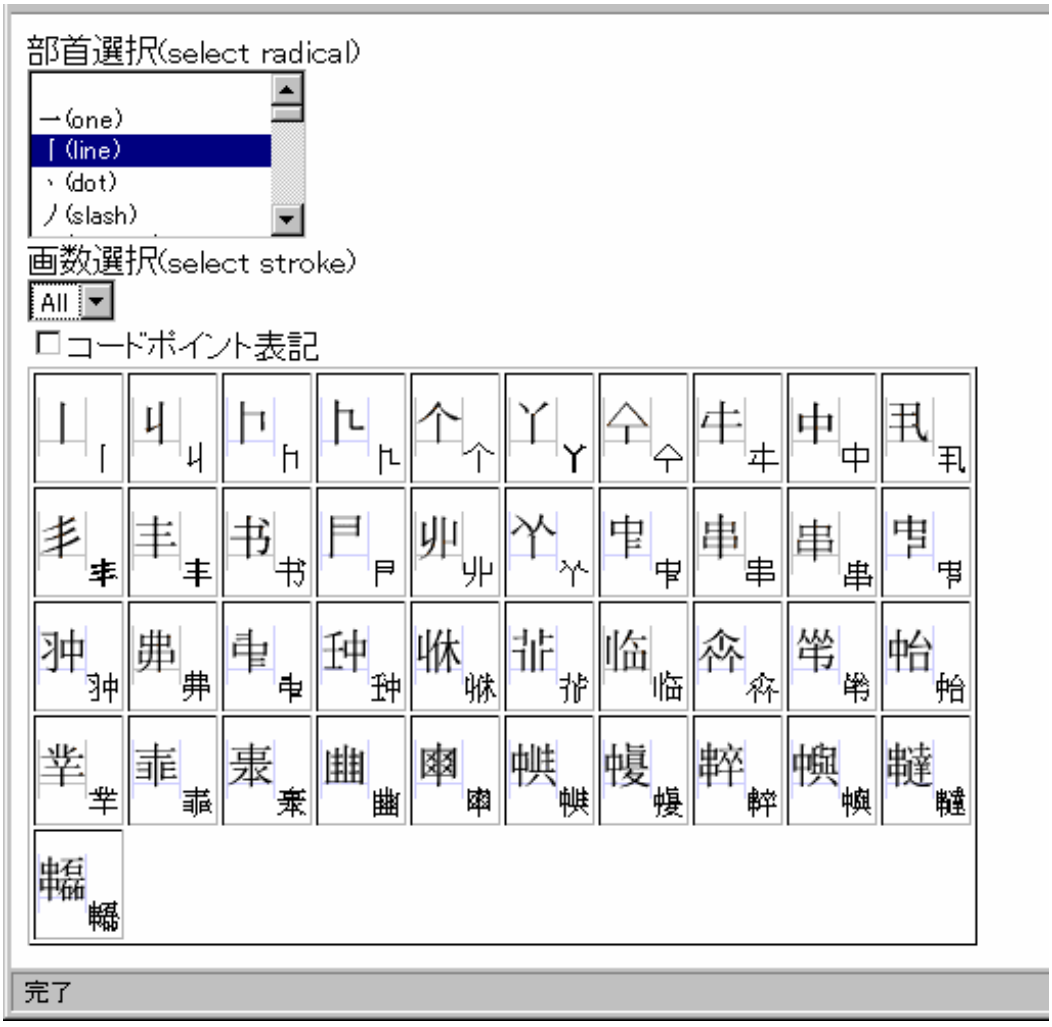


図 3.1.2　部首「丨」の全ての漢字を表示した例

図 3.1.3　MS-IME 2003 での部首「丨」での検索結果

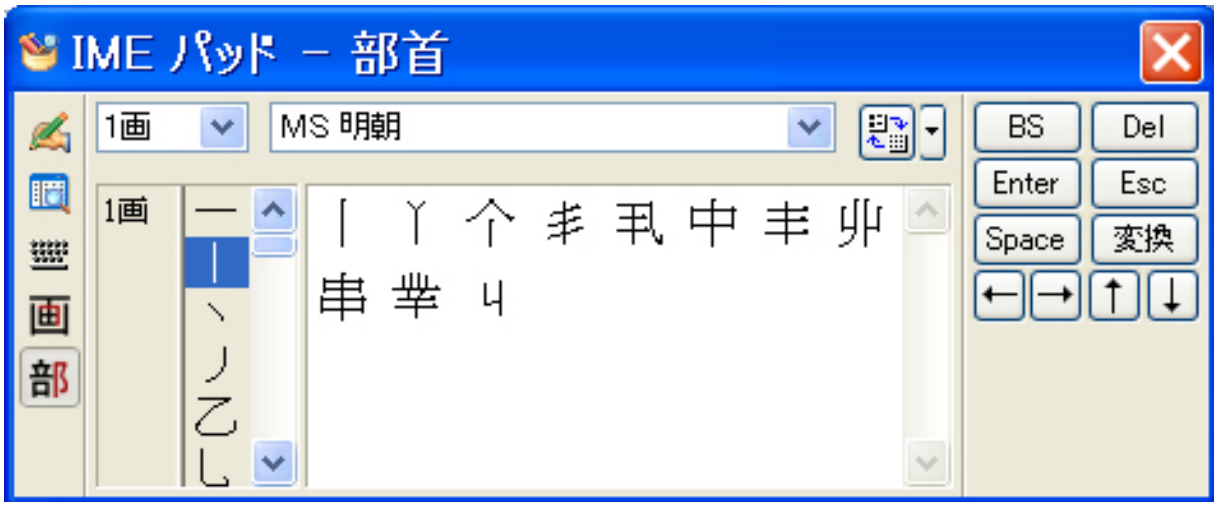


図 3.1.4　MS-IME 2007 での部首「丨」での検索結果

ここでは，馬偏（画数 10）の Unicode 文字で，以下の 2 文字（偏以外の画数 20）を検索し（図 3.1.5），

驫 U+9A6B　驫 U+299E2

「驫」を検索エリア入力した所（図 3.1.6）と，これらの検索文字のコード・ポイント表示した画面である（図 3.1.7）．実際の実装では，該当文字が多い場合，画面をスクロールしなければならないので，図 3.1.8 の様に 1 行 10 文字のみを表示し，画面表示の上に左右の→（矢印）マークを付けて，複数行の表示に対応させている．

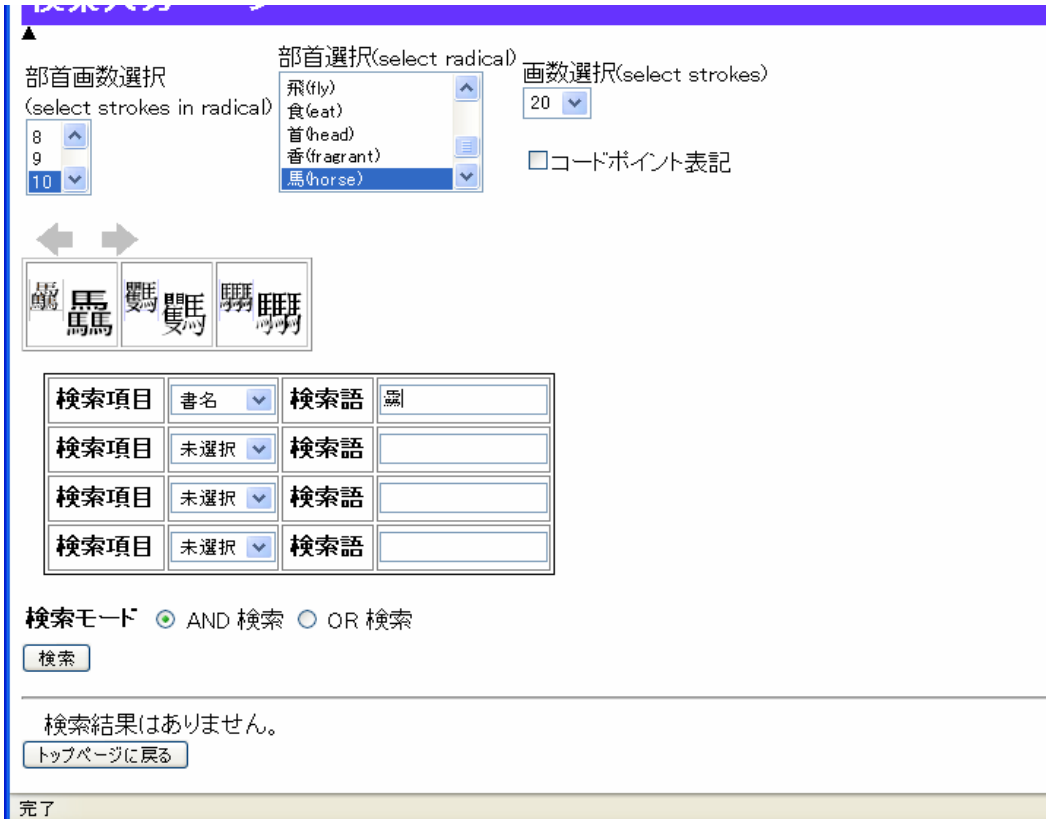


図 3.1.5　DOKB システムの実装した Unicode 漢字入力支援ツー

図 3.1.6　部首（馬偏）のその他の画数 20 の文字

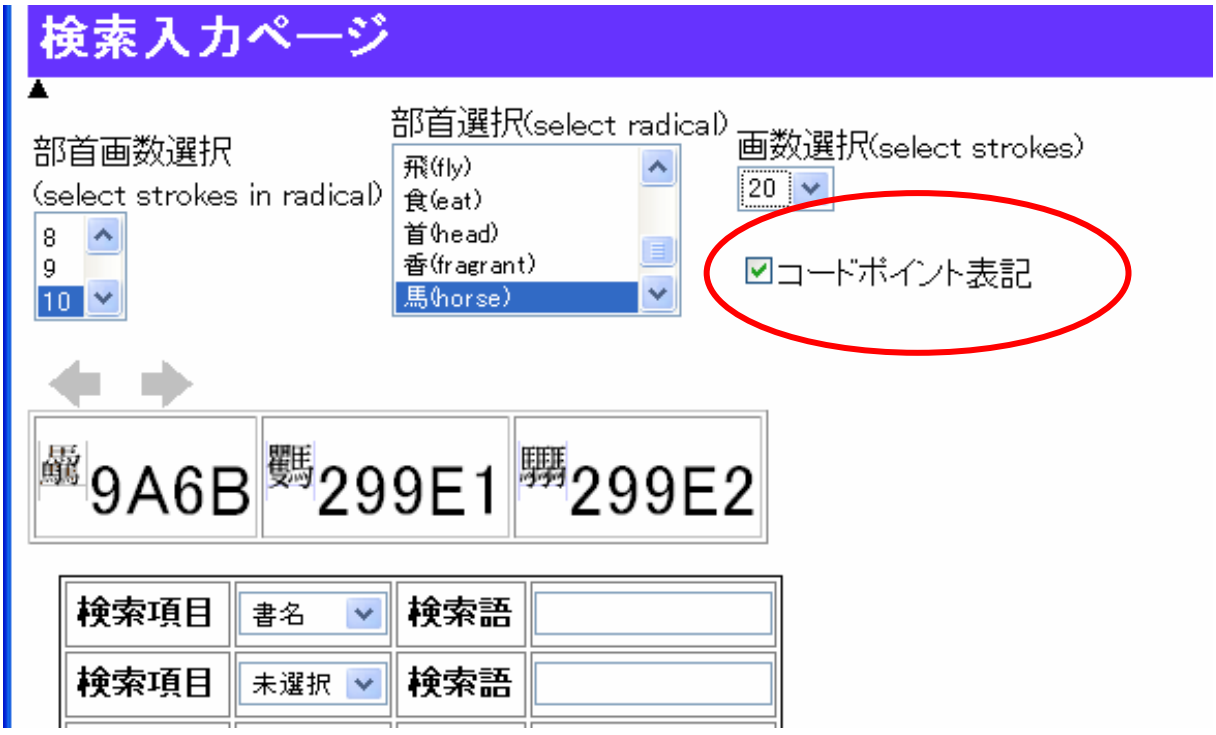


図 3.1.7　部首（馬偏）のコード・ポイント表示

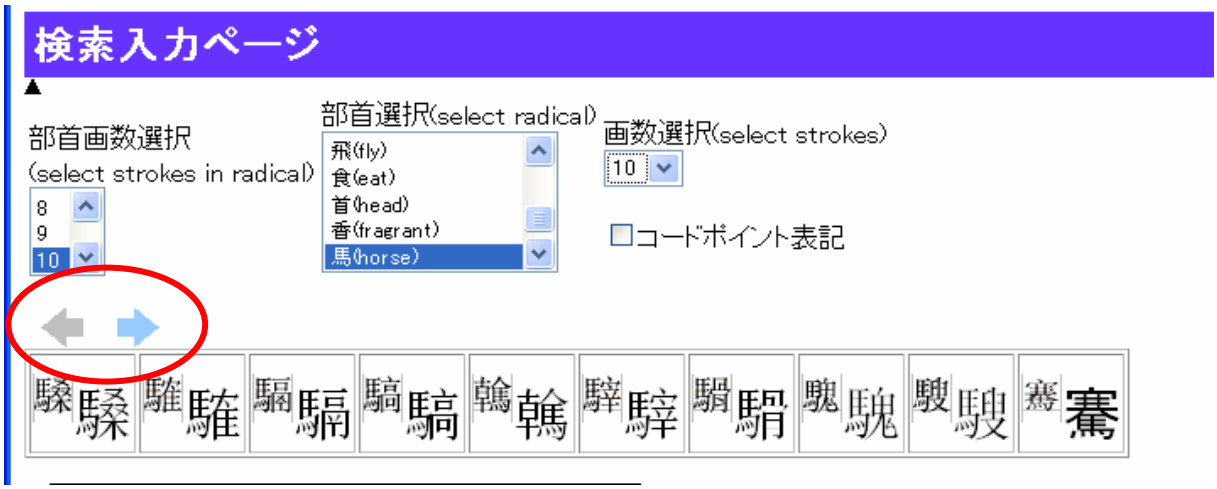


図 3.1.8　部首（馬偏）の複数行の場合 1 行 10 字表示

３．１．５　利用者側の環境に依存する問題

この入力支援ツールを利用する場合は，予め Unicode の拡張 A，拡張 B のフォント（HAN NOMA，HAN NOM B など）をインストールしておく必要がある．また，Imternet Explorer などの Web Browser では使用するフォントをブラウザ側で適切に設定を施す必要がある．Browser Firefox は自動的にフォントを探してくれるので，設定する必要はない．

３．２ 刻手名データベース検索システムの開発

３．２．１　刻手名データベースとその必要性

本データベースは，分担者の元富山大学人文学部教授の藤本幸夫先生が，数十年にわたって「日本現存朝鮮古刊本の調査とその語学的・書誌学的研究」をされ，平成１９年３月に出版された「朝鮮朝刊本刻手名集（第二版)」に掲載されたマスター・データをデータベース化したものである．

藤本教授曰く「刻手名を有する書籍は必ずしも多くはないが，刻手名によって，ある書籍の刊年と刊地を決定し得ることがある．どの地方で，どのような書籍が刊行されたかを知ることは，出版文化を考える場合，極めて重要である．」と．

筆者らが進める，印影・写真画像による日本現存朝鮮古書の画像データベースとの連動が実現すれば，原文画像と刻手名による古書の同定が進み，世界の各地に居ながら古文書の発掘，同定及び内容に関する研究ができるようになる．

本刻手名データベースは，日本現存朝鮮古書データベースと対をなす基盤データベースであり，今後，刻手名の画像や印影，古書原文画像データベースが加われば，古書研究に不可欠のものとなる．

３．２．２　刻手名データベースのテーブル定義

マスター・データのタグ番号と項目の関係を表 3.2.1 に示す．

表 3.2.1　刻手名データベースの項目

タグ番号	項目の内容	Discription
００	排列（配列）番号	Sequencial Number
０１	書名・巻数・冊数	Title
０２	撰者	Author
０３	刊年	Year of Publishing
０４	刊地	Place of Publishing
０５	所蔵者	Owner
０６	版心部刻手名	Graver Name in Center of Mountain Fold
０７	刊記部刻手名	Graver Name in Description of Publisher

※　ただし，該当項目未詳の場合は，その項目を省略する．

３．２．３　刻手名データベースの開発環境

刻手名データベース検索システムは，JavaEE を使って開発をすすめ，Application Server の Grass Fish を主体にして，JavaFramework JSF（Java Server Faces），EJB（Enterprise Java Beans），JPA（Java Perrsistence API）を使用して開発を終了した．

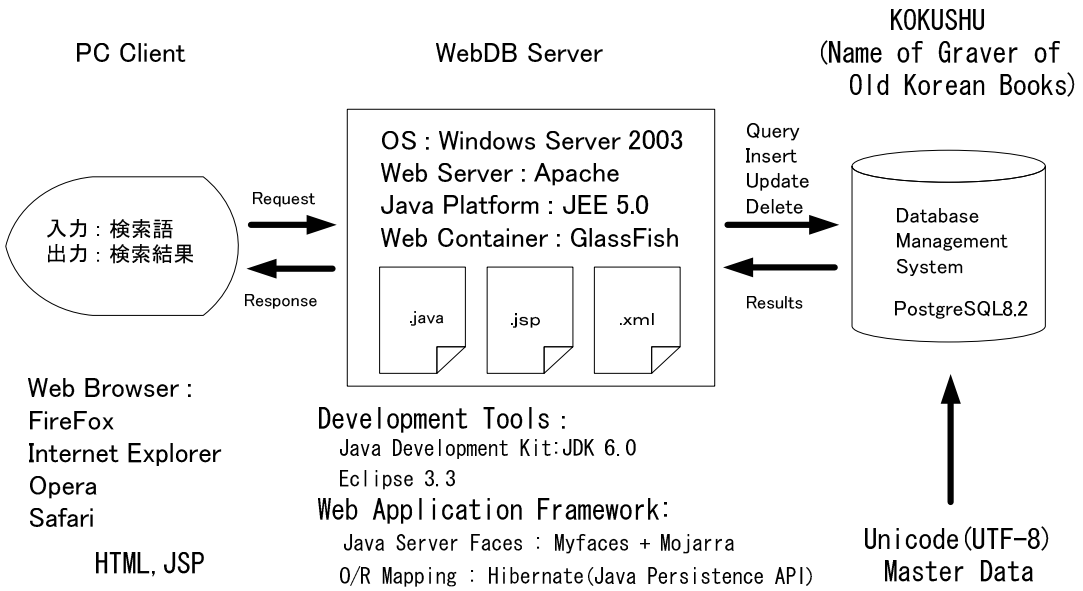
開発環境は Java Development Kit 6.0 と Ecripse 3.3 であり，Windows 2003 Server 上に KOKUSHU というデータベースを提供している．

手名データベースは，最初にユーザ認証を行ってから，検索画面に入ることができるようにしている．

データベースは Postgre-SQL を使用し，文献参照番号（K5）の他はすべて TEXT 属性で定義している．

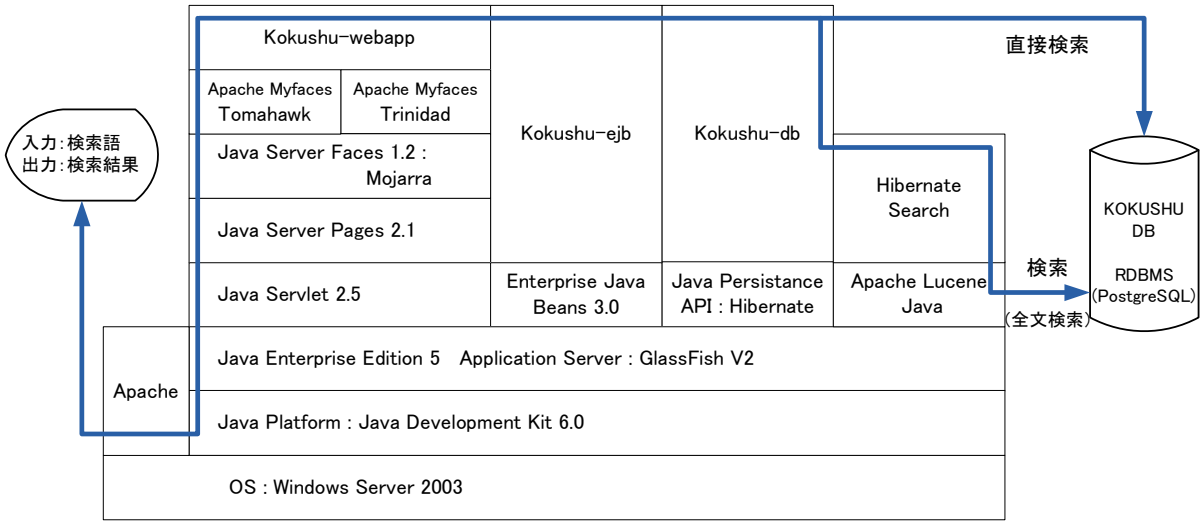
３．２．４　刻手名データベース検索システム・ソフトウェア構成

Software Configuration of Web Database System named KOKUSHU



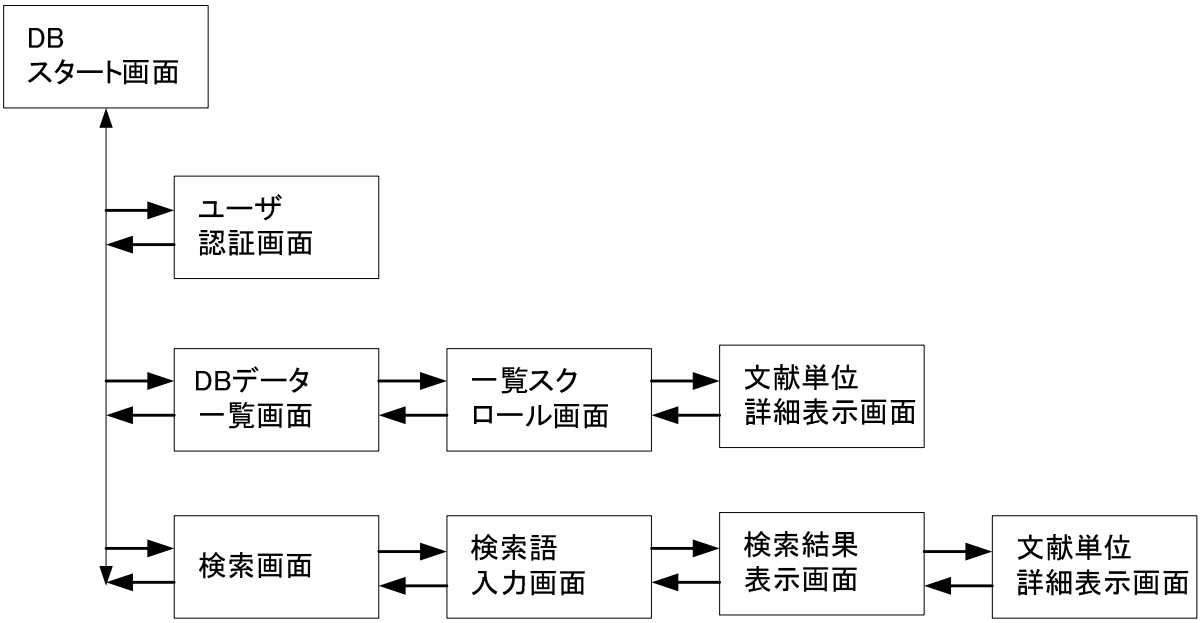
３．２．５　刻手名データベース検索システム・ソフトウェア構成部品

Software Components of KOKUSHU Database System



３．２．６　刻手名データベース検索システム画面遷移

刻手名データベース検索システム画面遷移図

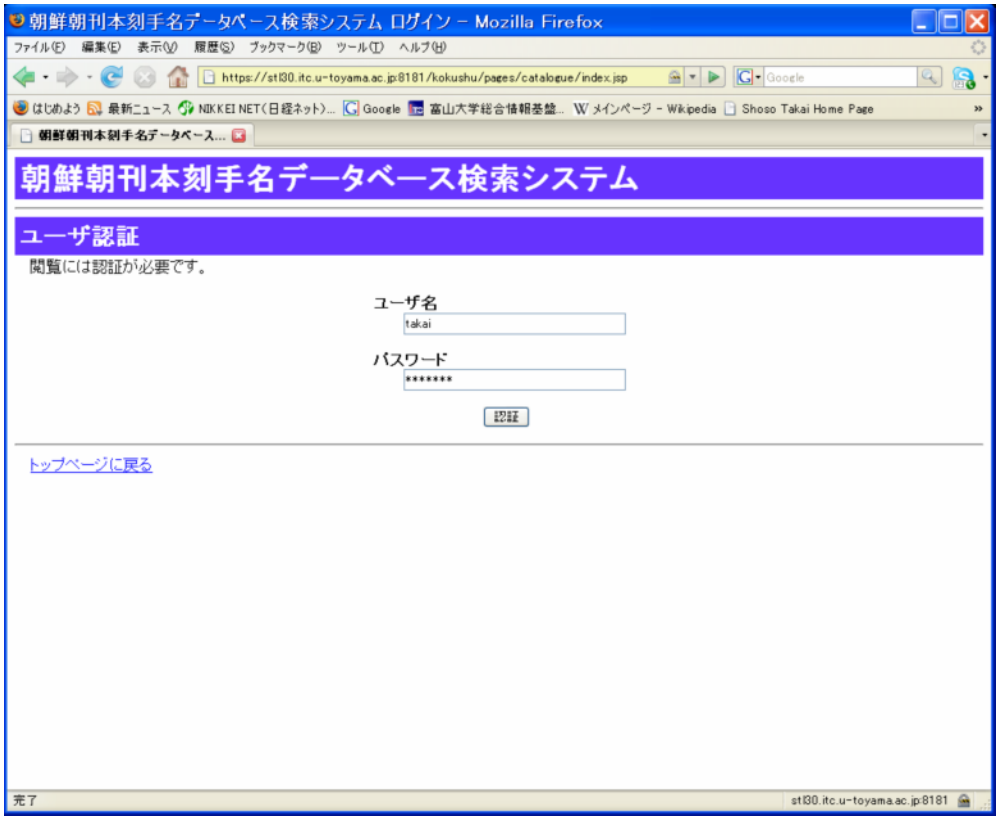


３．２．７　刻手名データベースの操作

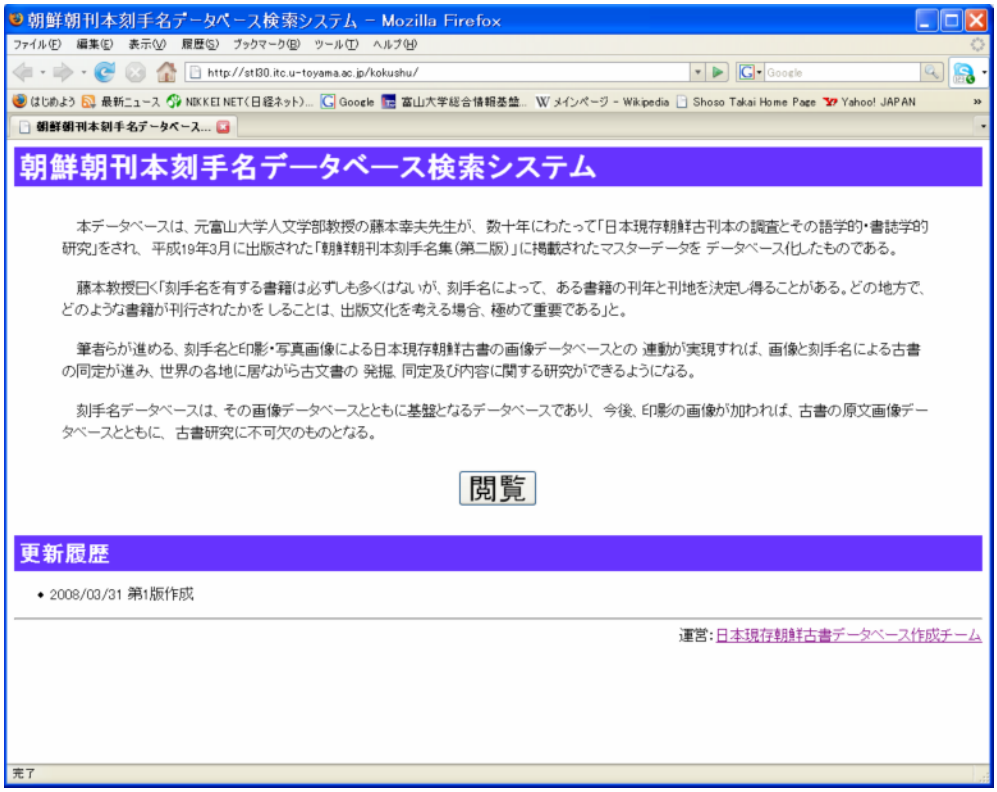
刻手名データベースは，最初にユーザ認証を行ってから，検索画面に入る．検索は一覧画面と全文検索画面から検索することができる．

画面の操作は次ページ以降の画面説明（１）～（18）の通りである．

（1）ユーザ認証画面

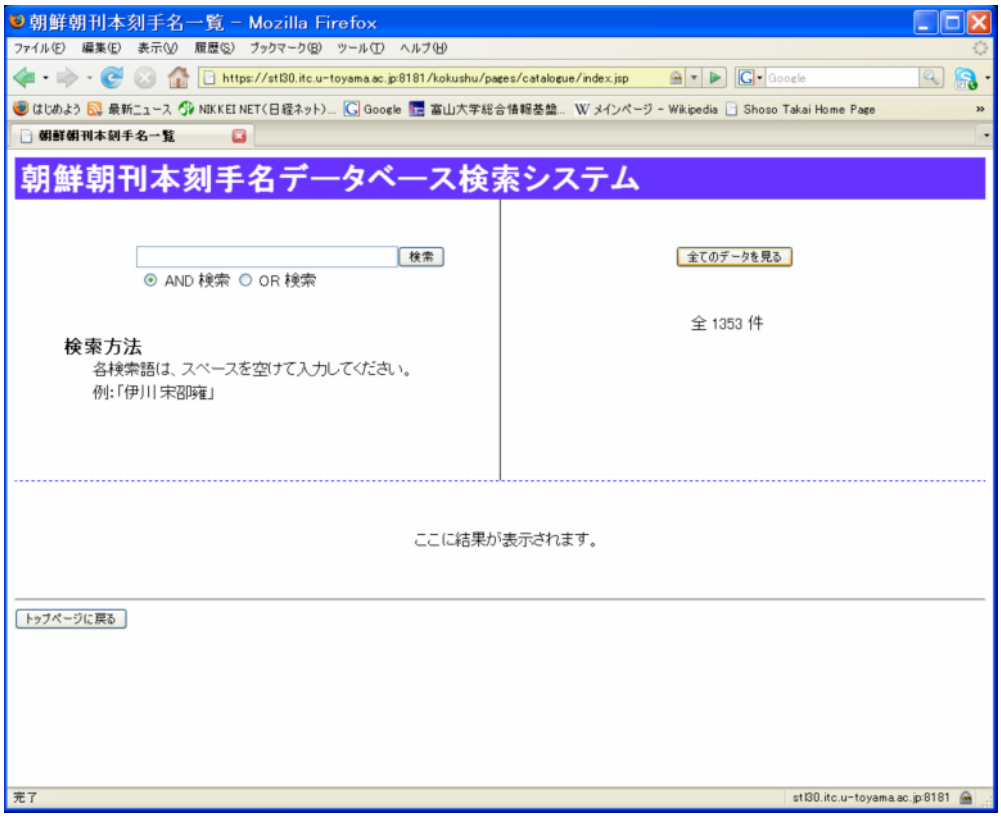


（2）朝鮮朝刊本刻手名データベース初期画面



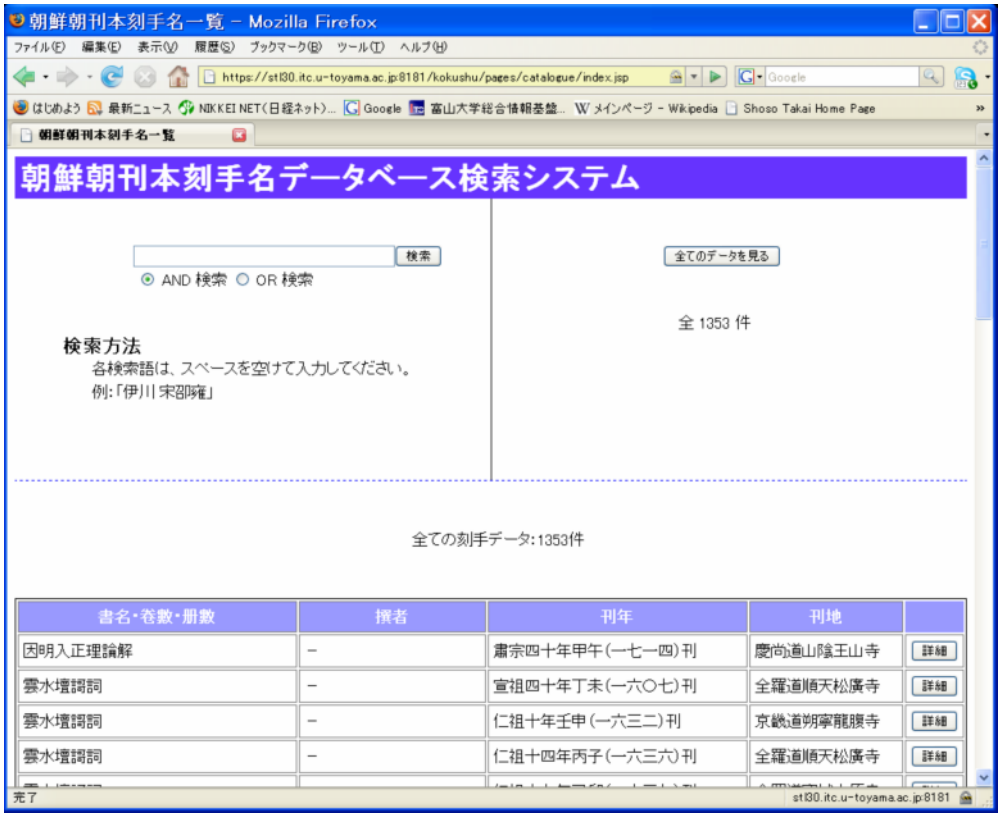
ここで，この刻手名データベースの概要と必要性を記している．
検索を開始するには［閲覧］ボタンを押下する．

（3）刻手名閲覧のための検索画面



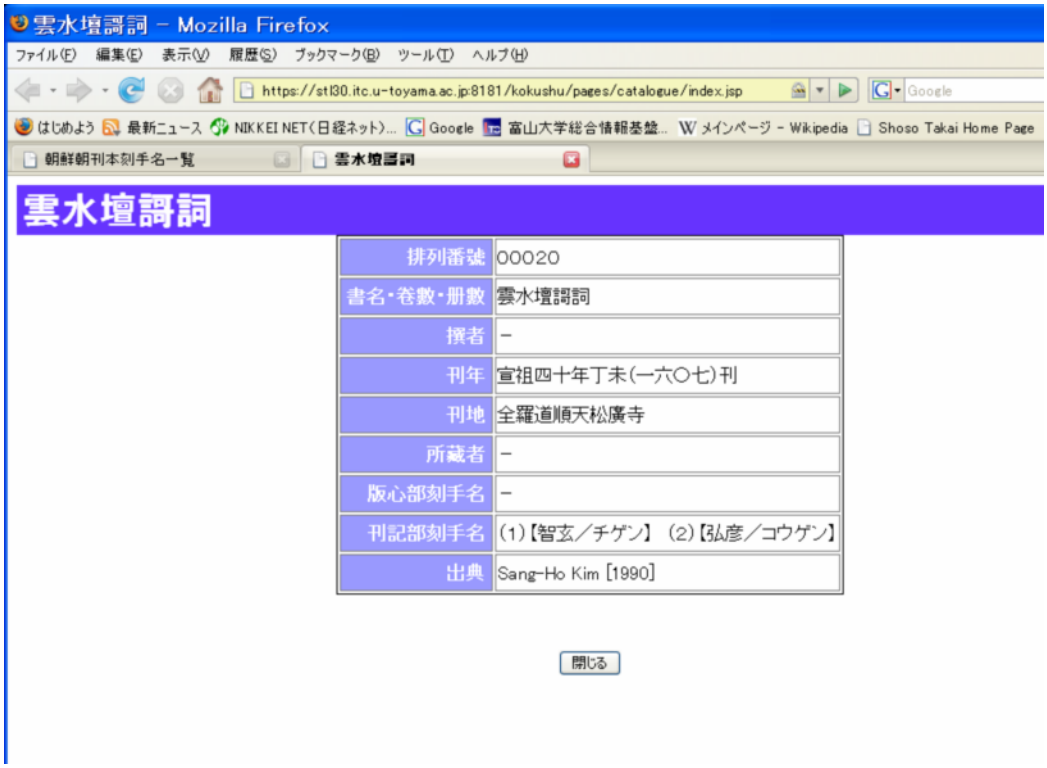
ここで，すべてのデータを見るために［全てのデータを見る］ボタンを押下巣する．

（4）全収録データ一覧画面

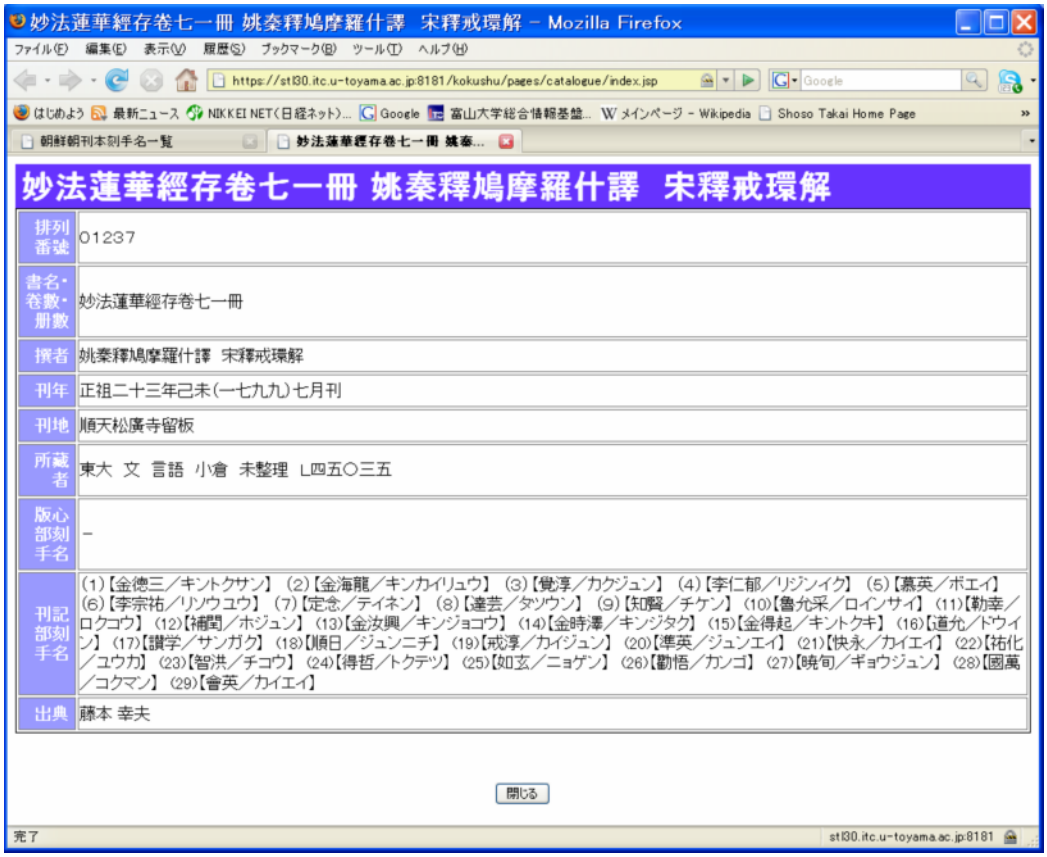


画面右側のスライダーを操作して該当書名を探します．

（５）一覧から書名「雲水壇謌詞」の詳細表示画面

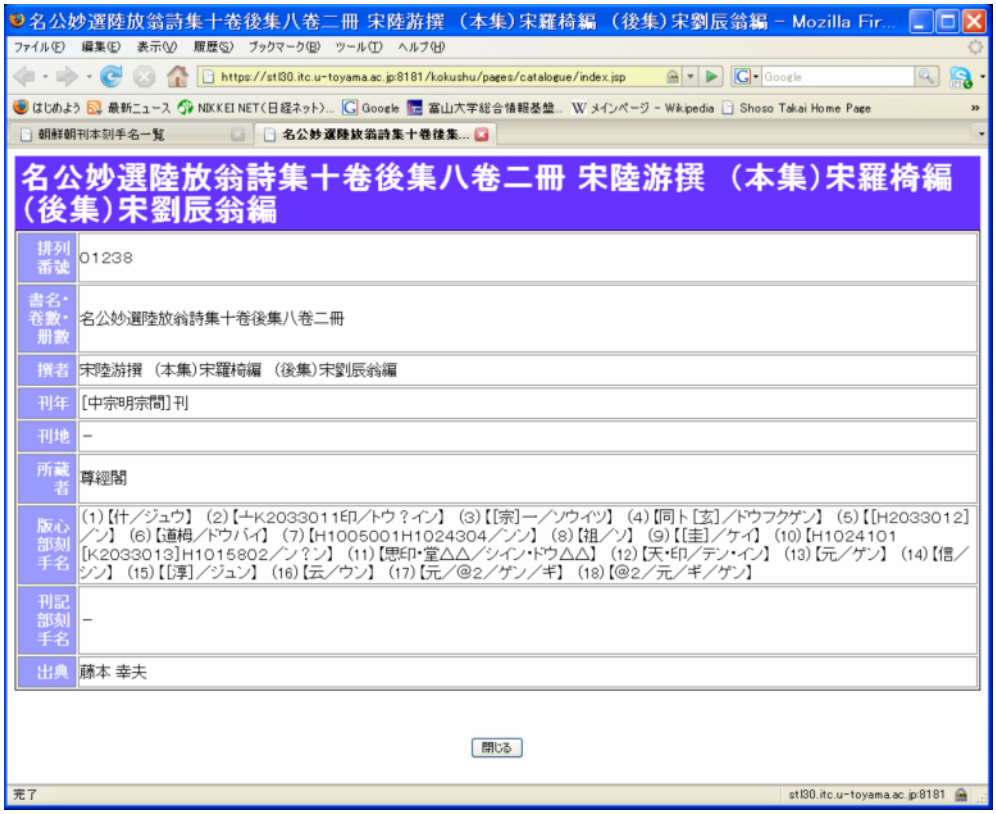


（６）一覧から 1237 番目の書名「妙法蓮華經存卷七一冊」の詳細表示画面



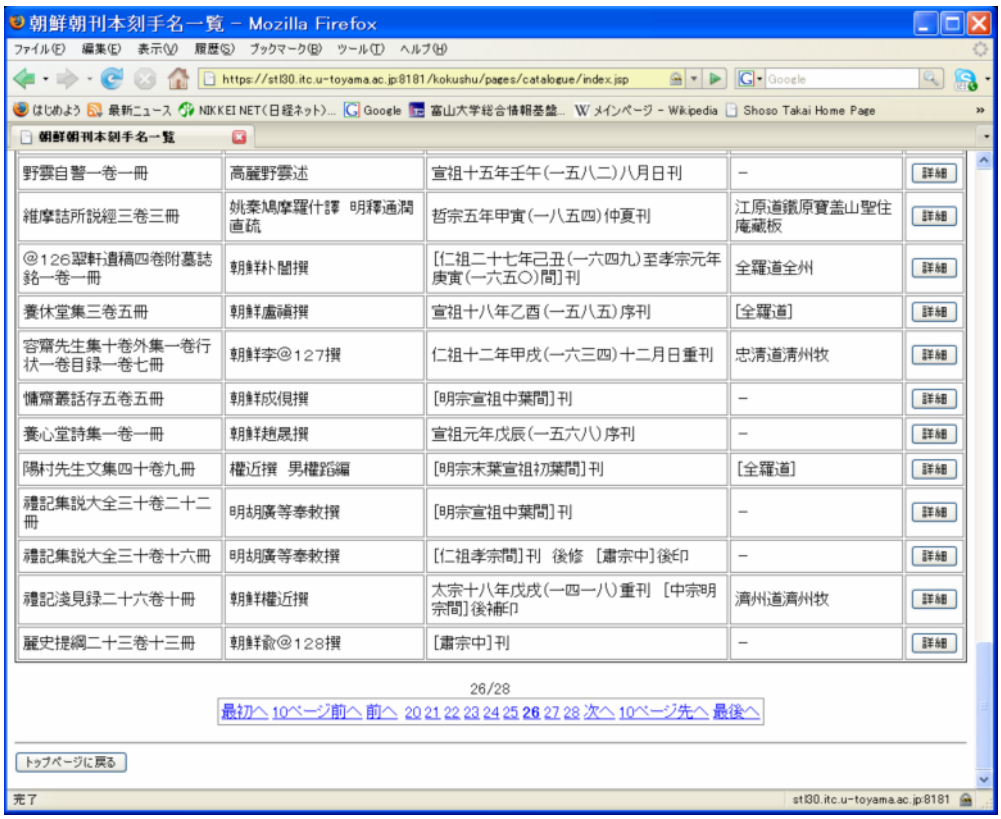
刻手名にはすべてカタカナで読みを表記している．

（７）一覧から 1238 番目の書名「名公妙選陸放翁詩集十巻後集八巻二冊」の詳細表示画面



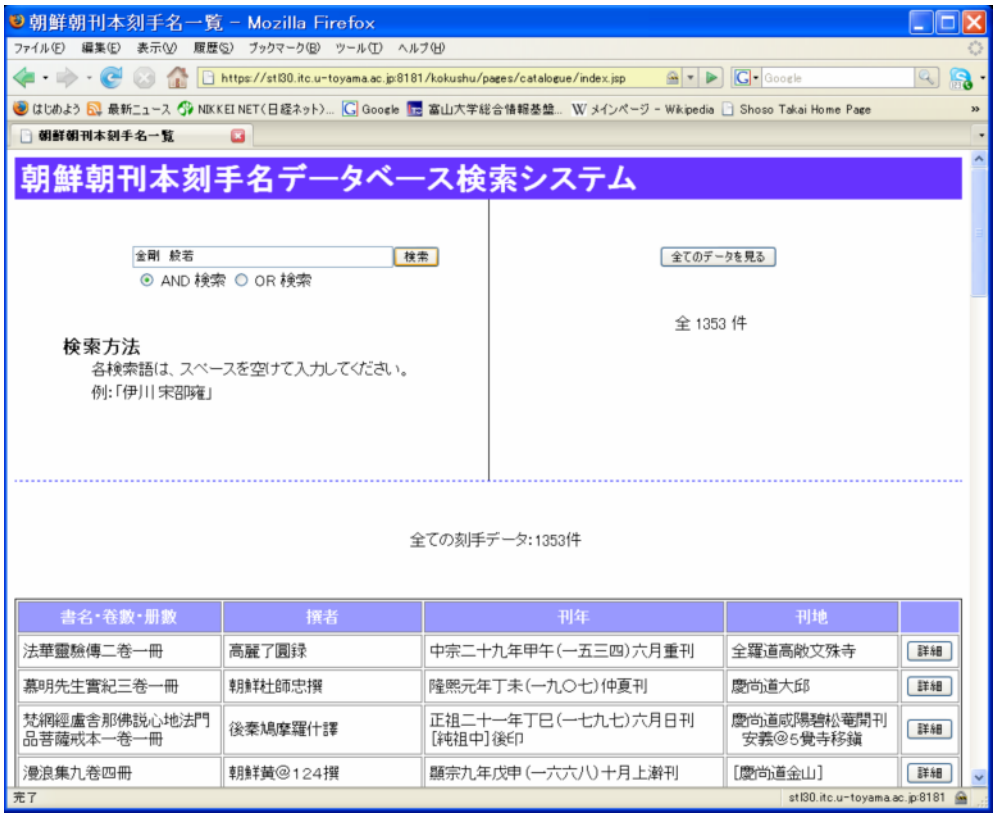
表記の「Ｋ２０３３０１１」は記号印，「Ｈ２０３３０１２」はハングルの置換表記である．

（８）一覧表示の 20 番ブロックの表示画面



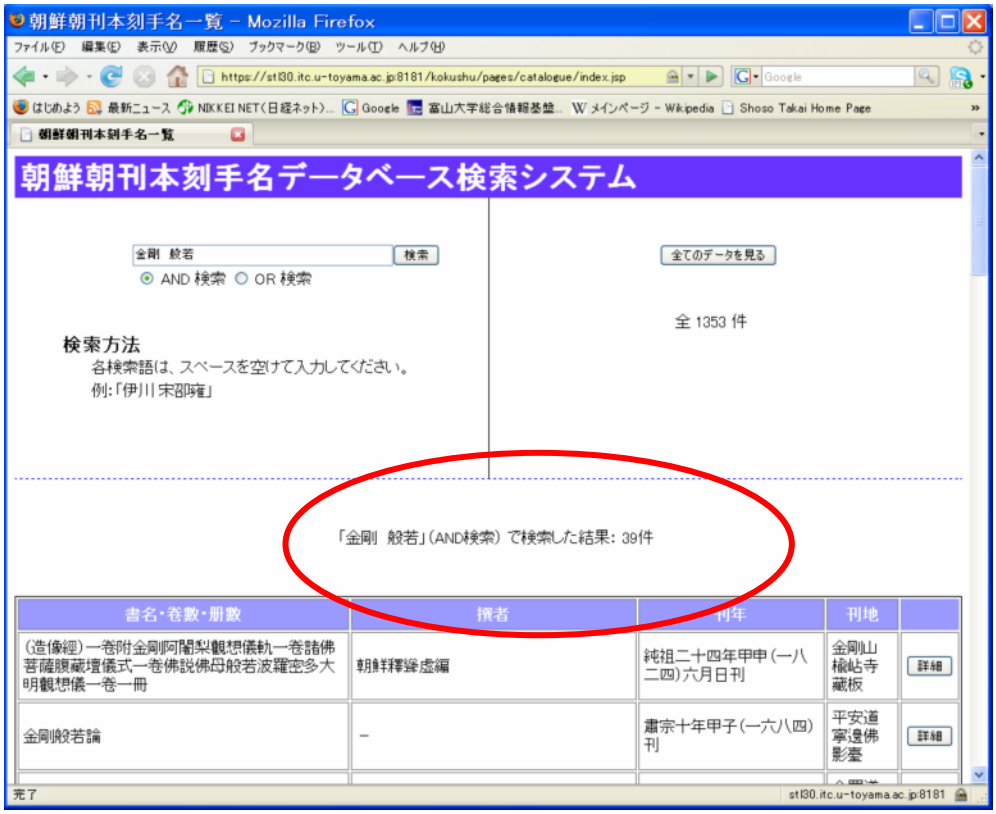
このように，最初へ，10 ページ前へ，前へ，･･･，次へ，10 ページ先へ，最後へと Jump 可能．

（9）キーワード検索画面



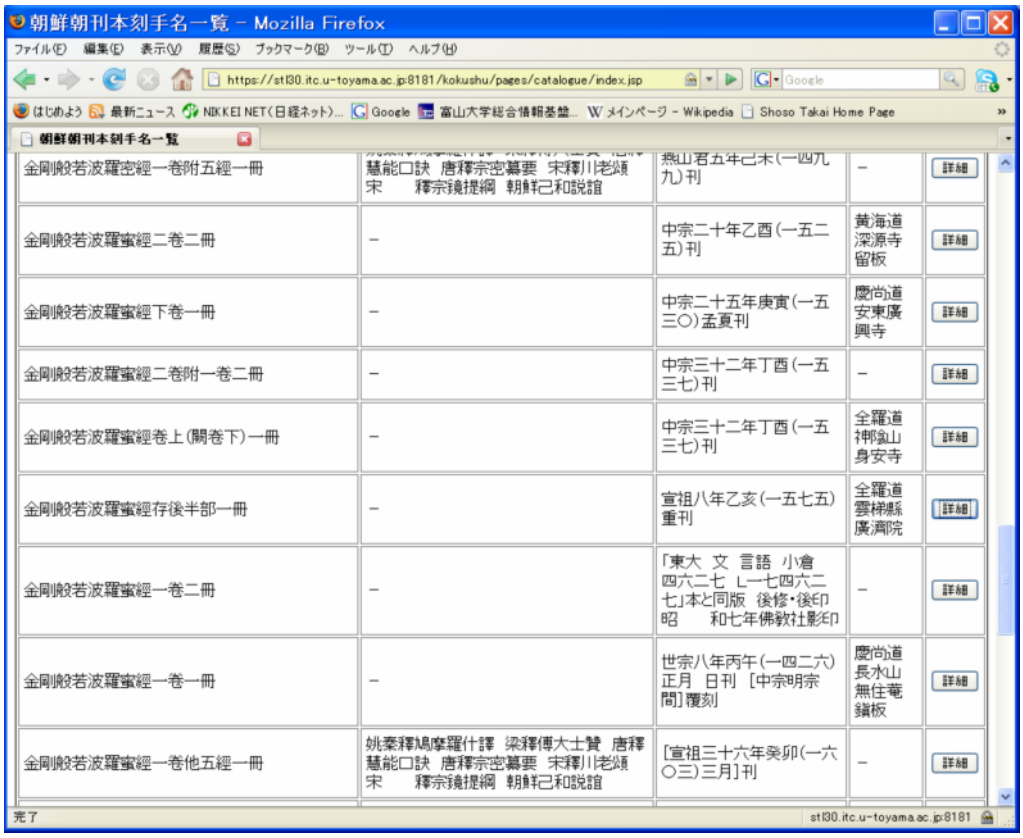
ここでは，AND 検索で「金剛」・「般若」を入力している．

（10）AND 検索「金剛」・「般若」で 39 件あったことの検索結果画面

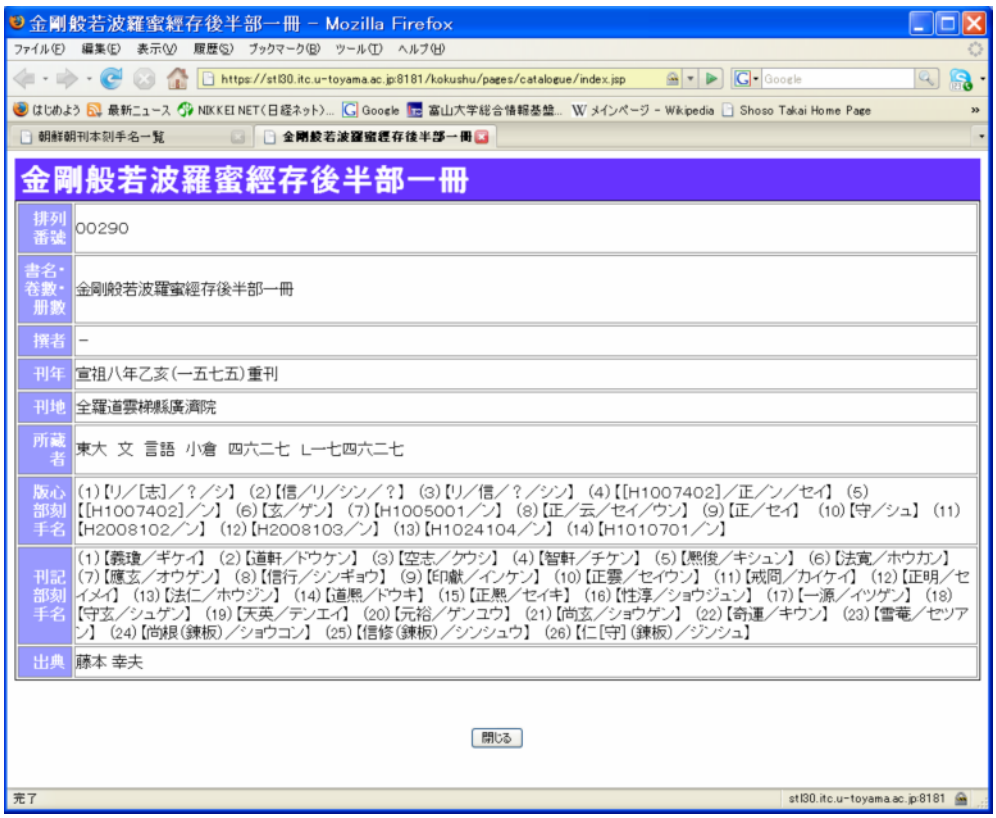


上記赤丸表示が検索結果の件数で，その下が 39 件の一覧表示である．

（11）AND 検索「金剛」・「般若」結果 39 件の続きを表示

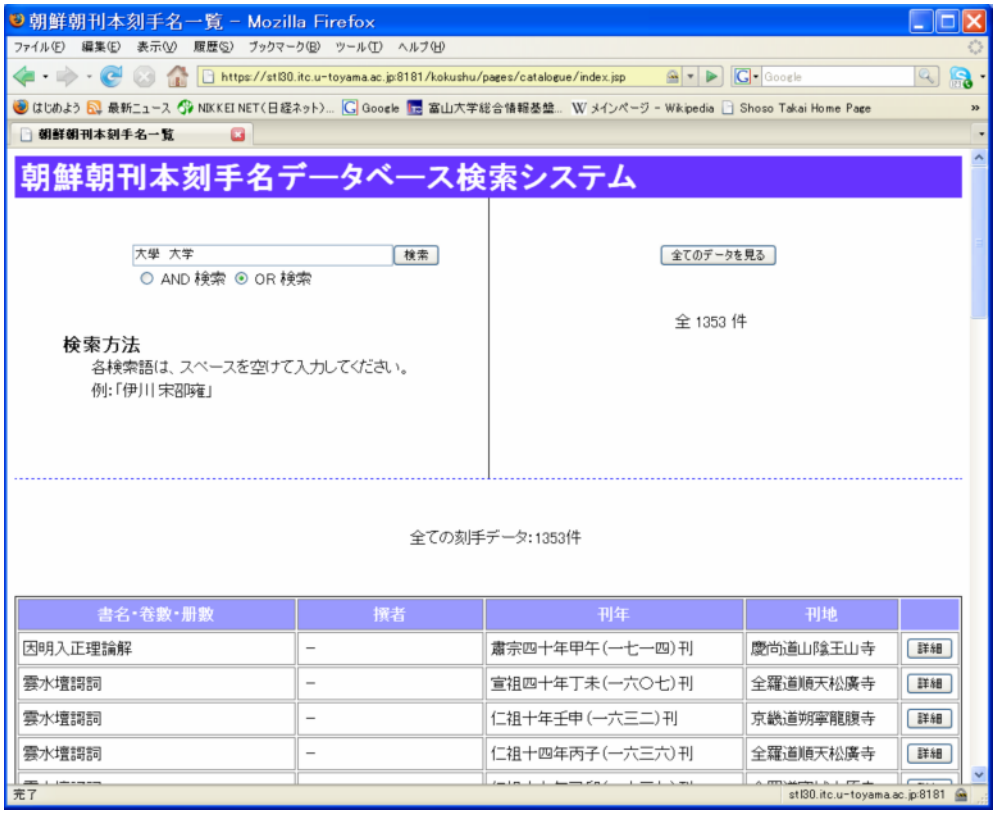


（12）検索結果の中から書名「金剛般若波羅蜜經存後半部一冊」の詳細表示画面



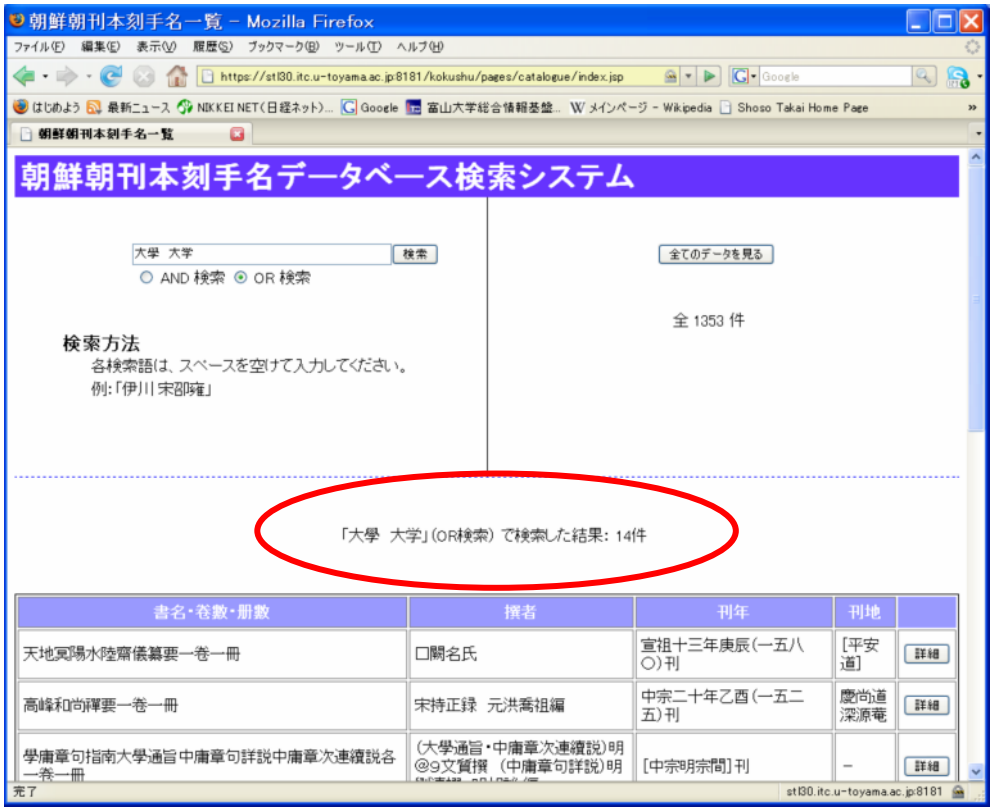
版心部の刻手名の大部分は，「Ｈ１００７４０２」などのハングル表記である．

（13）OR 検索で「大學」｜「大学」の入力画面



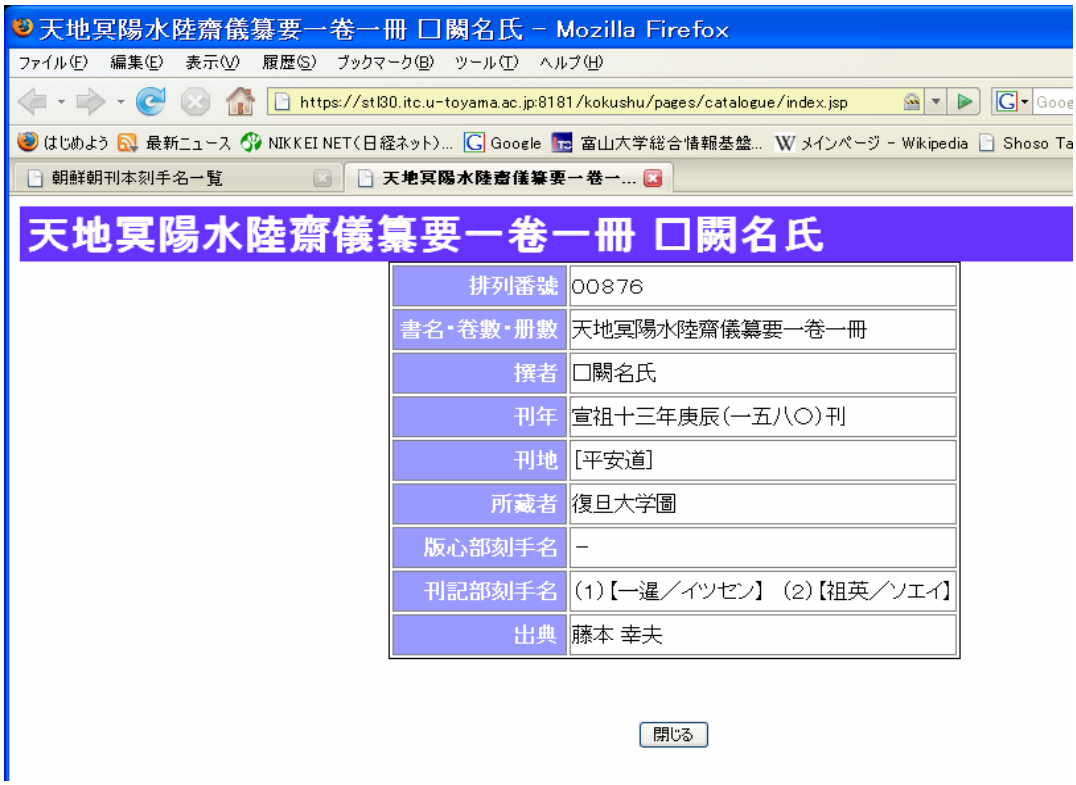
このような同一の新旧漢字も OR 検索することになる.

（14）OR 検索「大學」｜「大学」の結果表示画面



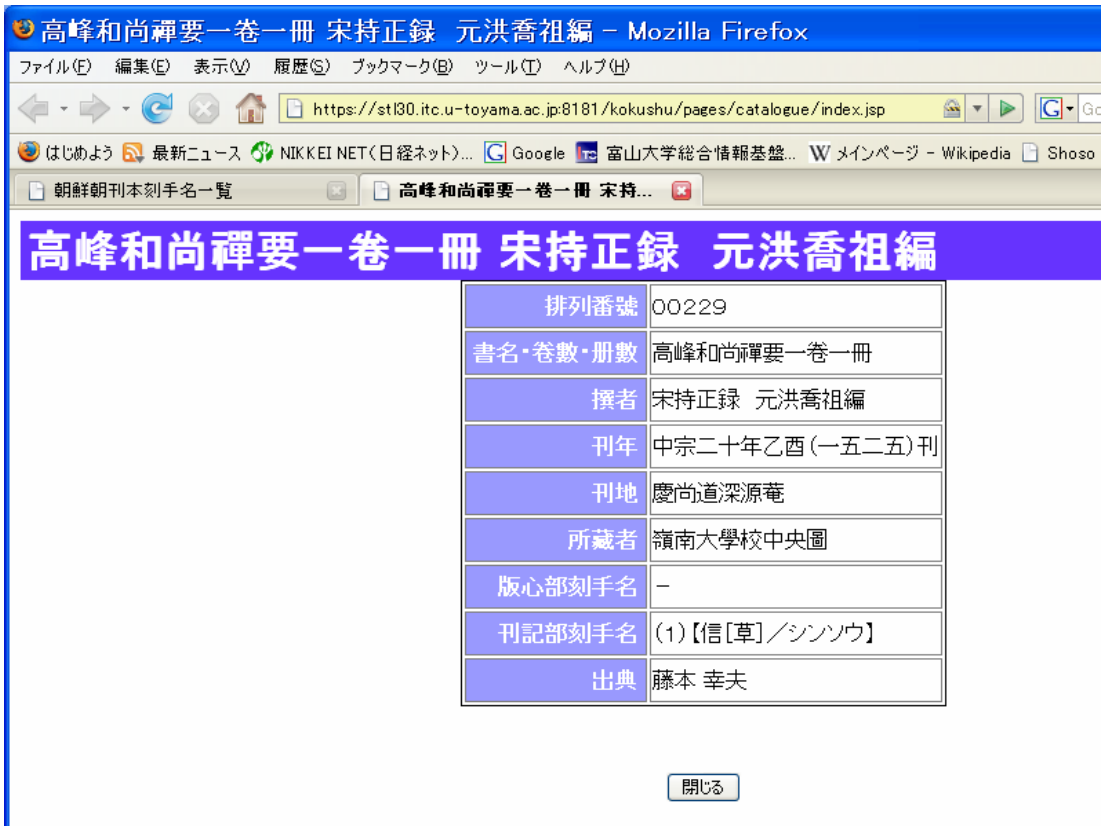
OR 検索「大學」｜「大学」で 14 件の該当書名があった.

（15）OR 検索結果から書名「天地冥陽水陸齋儀纂要一卷一冊」の詳細表示画面



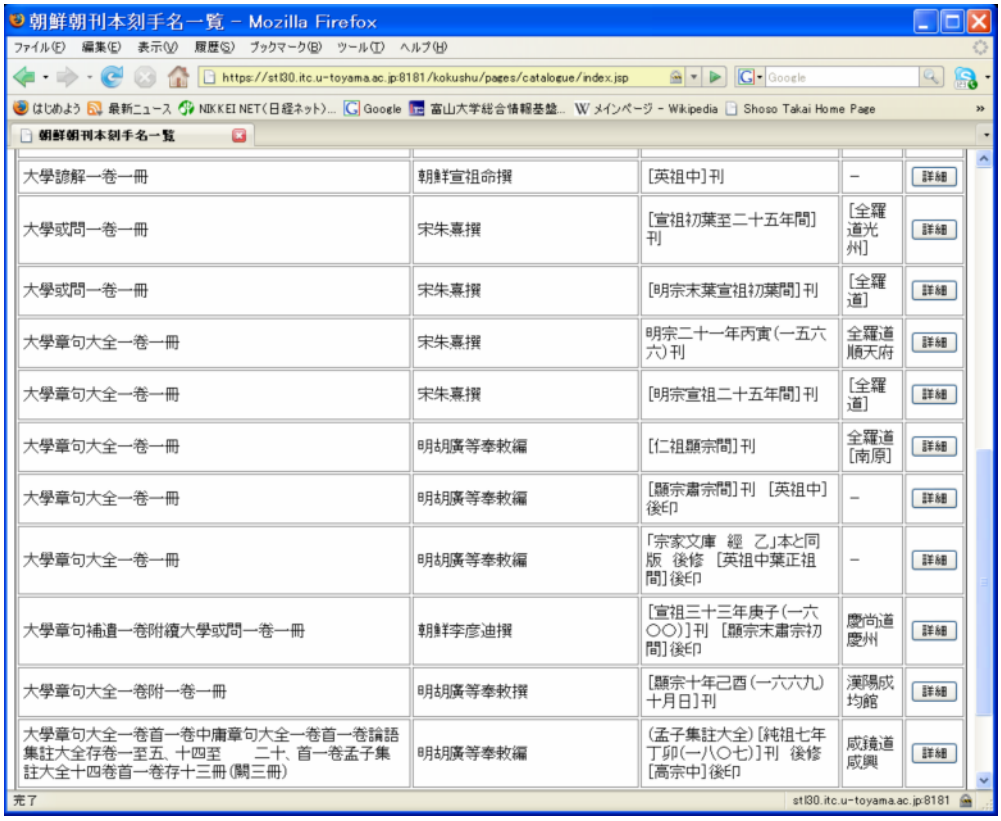
所藏者が「復旦大学圖」＝復旦「大学」図書館になっている.

（16）OR 検索結果から 2 番目の書名「高峰和尚禪要一卷一冊」の詳細表示画面



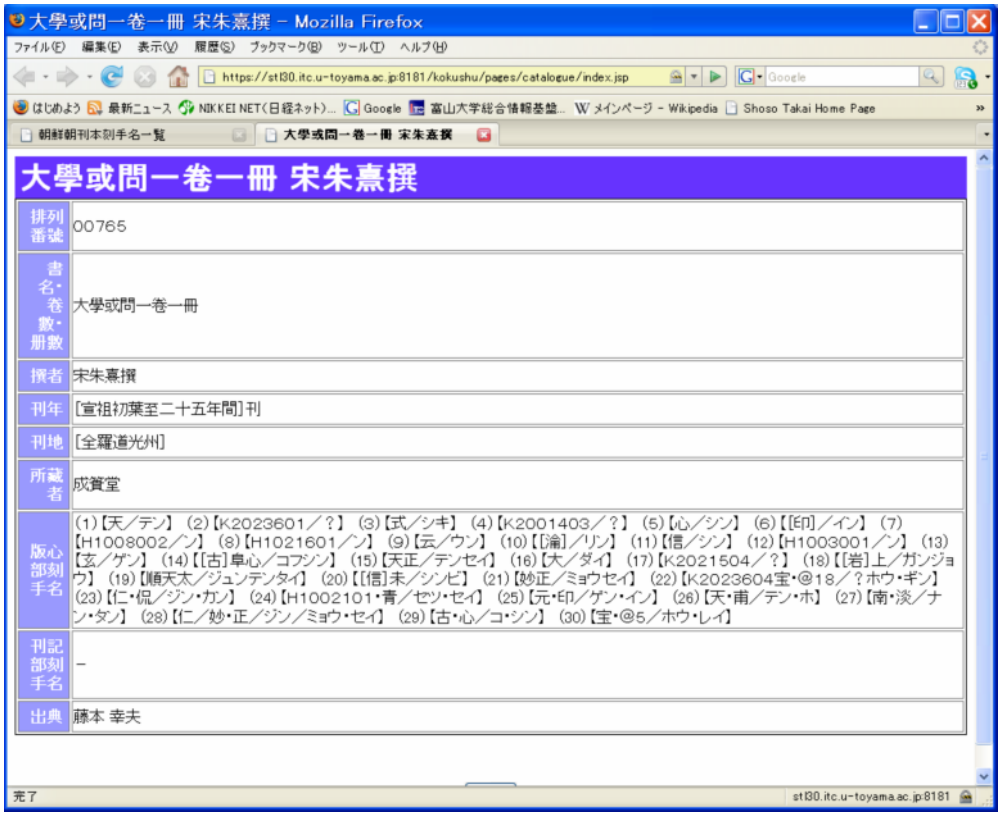
所藏者が「嶺南大學校中央圖」＝嶺南「大學」校中央図書館になっている.

（17）その他の OR 検索「大學」｜「大学」結果一覧画面



書名に「大學」を含むものが大部分である．

（18）書名に「大學」を含む「大學或問一卷一冊」の詳細表示画面



一冊の書物が数多くの刻手によって彫られていることが示されている．＠５は旧字体漢字．

３．３ Ajax 技法による原文画像データベース検索システムの開発

３．３．１　古書高解像度原文画像配信の必要性と可能性

今日まで古書原文画像のインターネットによる配信では，高解像度画像データは大容量のため，容量が限定された状態＝低解像度で，かつ，古書のページ単位で配信されてきていた．しかしながら，印影画像を伴う画像の分析や，角筆加点などの研究を進める上で，高解像度の原文画像の配信サービスが必要不可欠となる．

インターネット上で，このような高解像度画像を配信すると，一度に通信するデータ量が膨大なものとなる（図 3.3.1）．そのため，これまで Gigaview[4] や iPalletnexus[5] などのソフトウェアが開発されてきた．これらは Web Browser の Plug-in や Stand-alone Application であり，ユーザがインストール作業を行う必要があった．

そこで Google Maps[7]などで使用されている Ajax による画像の伝送システムを応用して，DOKB のサブ・システムとして提供するため，古書原文画像閲覧システムを提案することとした．

この Ajax 技法の応用によってユーザのインストール作業がなくなり，また，特定の要件を満たした Web Browser であれば簡単に動作するため，動作環境の制限も緩くなる．

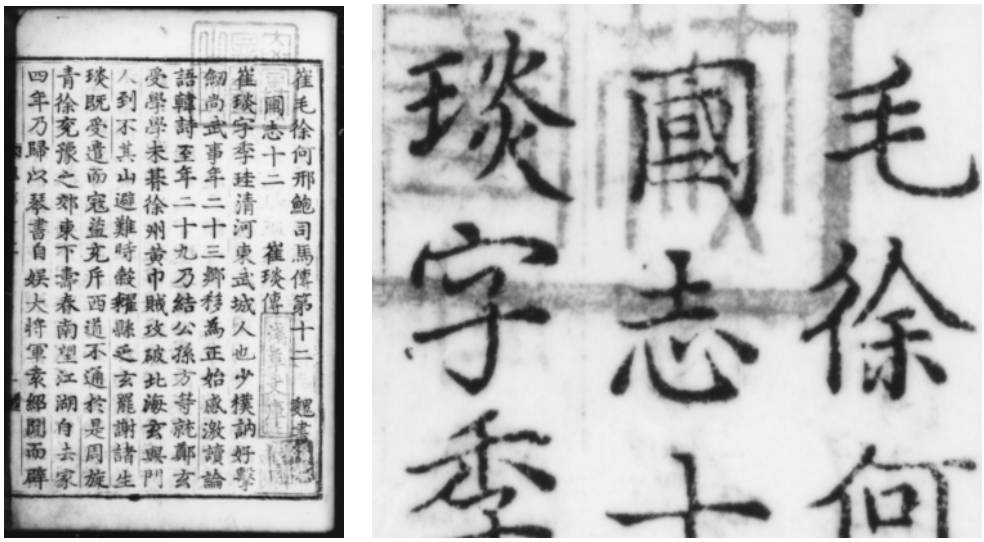


図 3.3.1　左約 100KB（320×500px, PNG）を右側全体で約 12MB 高解像度（約 4500×7300px, PNG）

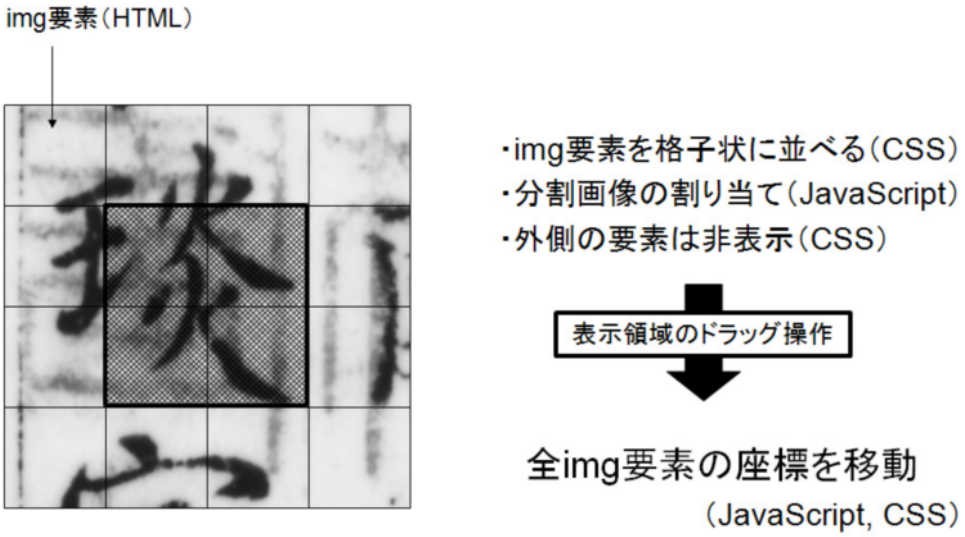


図 3.3.2　img 要素を格子状に配置

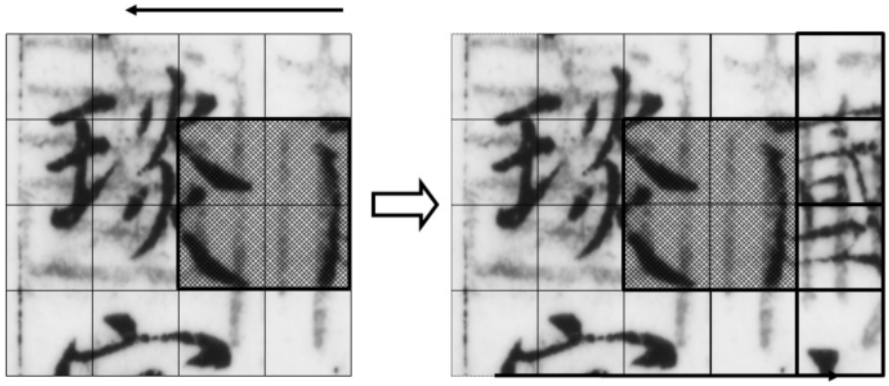


図 3.3.3　全 img 要素の座標を移動

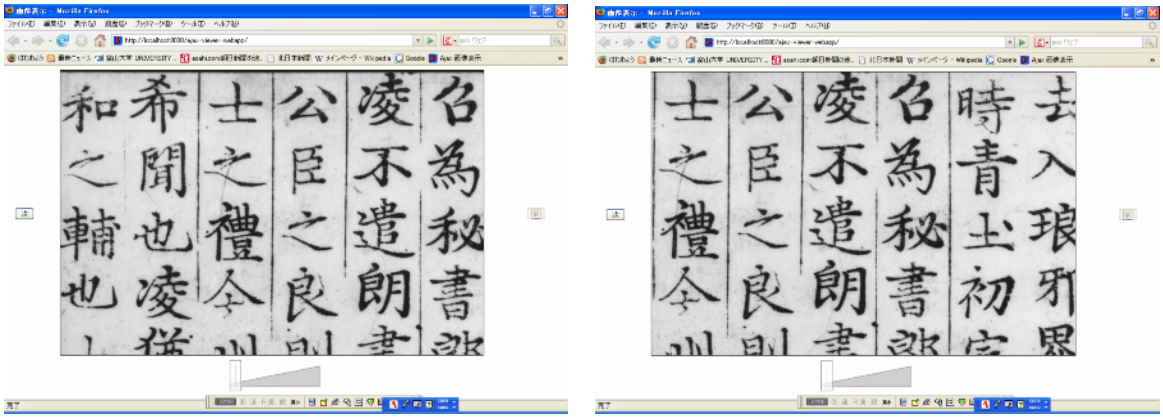


図 3.3.4　画像中央の「禮」を左へドラッグした例

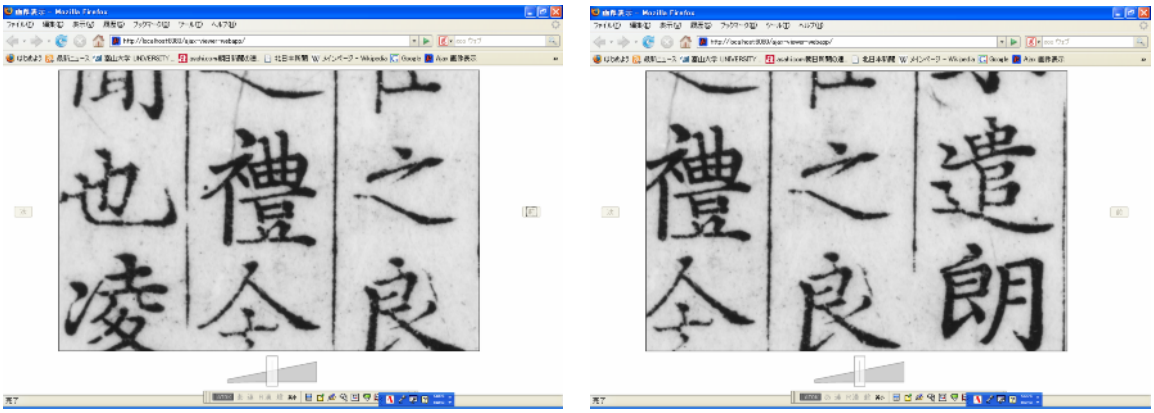


図 3.3.5　画像を拡大し「禮」を左へドラッグした例

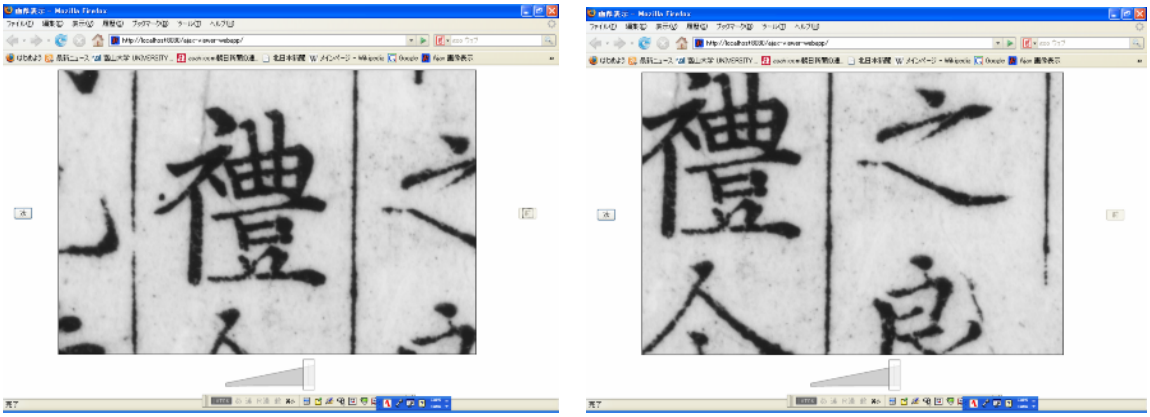


図 3.3.6　画像を更に拡大し「禮」を左上へドラッグした例

３．３．２　Ajax による高解像度画像配信の方法

Ajax では，全体約 12MB の PNG 画像を各 40KB の 256×256px の画像に分割して img 要素を作成し，格子状に並べる（図 3.3.2）．

必要な画像を非同期で配信する方法を採用する．カーソルの移動に応じて受信した画像を Dynamic HTML で組み立て，Web ページを動的に書き換えている（図 3.3.3）．従って，画面の遷移がない状態で画像の拡大縮小や上下左右の移動が，マウスのドラッグだけで可能となる．

３．３．３　実装例

次の図 3.3.4～図 3.3.6 は，原文画像の「禮」という文字を左へドラッグし，順次拡大して画像を移動したときの例である．このように高解像度画像を，画面遷移無しに自在に移動し，その詳細画像を閲覧することができる．

３．３．４　メタ・データの定義

原文画像のメタ・データの項目定義は表 3.3.7 の通りである．ただし，開発中の検索システムではデータベースのテーブル定義がこの表と多少異なっている．今後，エンティティの正規化を含め，定義の見直しが必要であると考えている．

表 3.3.7　原文画像メタ・データの項目定義

No.	項目の内容	Discription
1	文書画像メタ・データ ID 番号	Document Image Metadata ID Number
2	書名	Title
3	撰者	Author
4	版種	Kind of Engraving Cut
5	刊者	Publisher
6	刊年	Year of Publishing
7	刊地	Place of Publishing
8	書名カナ	Title Katakana
9	書名 Pinyin	Title Pinyin
10	書名（別名）	Title（Alias）
11	書名（別名）カナ	Title（Alias）Katakana
12	書名（別名）Pinyin	Title（Alias）Pinyin
13	分類コード	Category Code
14	所蔵者コード	Owner Code
15	リザーブ１	Reserve-1
16	リザーブ２	Reserve-2
17	リザーブ３	Reserve-3

※　ただし，該当項目未詳の場合は，その項目を省略する．

原文画像データの登録項目の定義を表 3.3.8 に示す.

表 3.3.8　原文画像データのテーブルの定義

No.	項目の内容	Discription
1	画像参照番号　（F5）	Reference Number
2	資料ページ番号　（F3）	Image Page Number
3	文書画像メタ・データ ID 番号（F5）	Document Image Metadata ID Number
4	印影サイズ	Seal Impression Image Size
5	備考	Remarks
6	印影画像ファイルへの相対パス	Related Pass to Seal Impression Image File

※　ただし，該当項目未詳の場合は，その項目を省略する.

３．４ Ajax 技法による検索語類推入力支援ツールの開発

３．４．１　検索語類推支援ツールの可能性

Google Suggest [6]では，検索したいキーワードを検索クエリ・ボックスに一字ずつ入力する度に，検索語の類推機構が働き，キーワードの候補が表示され，利用者は該当候補一覧から該当語を選択して入力することができる（図 3.4.1）.

この方法にヒントを得て，DOKB の書名や撰者名の検索語は有限であり，全てのキーワードを DB 化して，利用者が検索語の一文字をインクレメンタルに入力する度にキーワードを類推表示す

図 3.4.1　左は「ai」，右は「えいじゃ」で該当語を表示

れば，無駄な検索語の入力を排除し，検索語入力の迅速化を支援できる．ローマ字入力なら漢字入力の国際化に対応できる．pinyin，Hepburn，McCune Reischauer などを使用できるようにすれば，DOKB の国際化を推進することができる．

だたし，キーワードの任意一致などの課題にも対応が不可欠で，解決すべき課題は多い．

３．４．２　検索語類推支援ツール構築手順

このツールは Ajax 技法を使用し，次の手順で構築する．

１）検索キーワードとして書名，撰者名を抽出.

２）カタカナをローマ字変換.

３）項目別 XML データベースの作成

・書名 XML ファイル

・撰者名 XML ファイル

４）検索語類推支援ツールの Ajax による作成

５）検索システムへのツールの組み込み

当初，DOKB のマスター・データは，索引を作る目的で全ての書名，撰者名，人名などに読みとしてカタカナを付加して入力している．

３．４．３　項目別 XML データベースの作成

今回，ローマ字入力で書名，撰者名の類推支援をするための XML データベース項目を表 3.4.2 の様に定義し，XML ファイルを作成した．

具体的には図 3.4.3 が書名 XML ファイル，図 3.4.4 が撰者名 XML ファイルで定義例である．

表 3.4.2　XML ファイル項目

書名項目	書名（漢字，Unicode）
	書名（カタカナ）
	書名（ローマ字：Hepburn）
	書名インデックス（文書番号，予約）
撰者名項目	撰者名（漢字，Unicode）
	撰者名（カタカナ）
	撰者名（ローマ字，Hepburn）
	撰者名インデックス（文書番号，予約＝将来使用予定）

```
<book_title>
  <title_kanji>一斎先生集績録</title_kanji>
  <title_kana>イッサイセンセイシュウゾクロク</title_kana>
  <title_hepburn>issaisenseishuuzokuroku</title_hepburn>
  <num_idx>3</num_idx>
  <idx>0226</idx>
  <idx>0230</idx>
  <idx>0231</idx>
</book_title>
<book_title>
  <title_kanji>一斎先生績集</title_kanji>
  <title_kana>イッサイセンセイゾクシュウ</title_kana>
  <title_hepburn>issaisenseizokushuu</title_hepburn>
  <num_idx>1</num_idx>
  <idx>0233</idx>
</book_title>
```

図 3.4.3　書名 XML ファイルの定義例

```
<book_author>
  <author_kanji>安宗源</author_kanji>
  <author_kana>アンソウゲン</author_kana>
  <author_hepburn>ansougen</author_hepburn>
  <num_idx>1</num_idx>
  <idx>0697</idx>
</book_author>
<book_author>
  <author_kanji>安英老</author_kanji>
  <author_kana>アンエイロウ</author_kana>
  <author_hepburn>aneirou</author_hepburn>
  <num_idx>2</num_idx>
  <idx>2208</idx>
  <idx>2209</idx>
</book_author>
```

図 3.4.4　撰者名 XML ファイルの定義例

３．４．４　Ajax による検索語類推支援ツールの実装

Ajax による検索語類推支援ツールの組み込みには，XMLHttpRequest オブジェクトを使用するが，IE（Internet Explorer）以外の Browser では使用に問題がないが，IE では図 3.4.5 に示すように Version の違いによって ActiveXObject を使い分ける必要がある．

```
function createHttpRequest()
{
        //Windows Internet Explorerか?
        if(window.ActiveXObject)
        {
                try
                {
                        // MSXML2以降か?
                        return new ActiveXObject("Msxml2.XMLHTTP")
                }
                catch(e)
                {
                        try
                        {
                                // 旧 MSXMLの場合
                                return new ActiveXObject("Microsoft.XMLHTTP")
                        }
                        catch(e2)
                        {
                                return null
                        }
                }
        }
        else if(window.XMLHttpRequest)
        {
                // Windows Internet Explorer以外のブラウザの場合
                return new XMLHttpRequest()
        }
        else
        {
                return null
        }
}
```

図 3.4.5　XMLHttpRequest オブジェクトを使用例

図 3.4.6～7 では書名入力の類推支援ツールの試作品でのインクレメンタルなローマ字入力と表示の違いを示しており，プルダウン・メニューの中から該当の書名をクリックすれば，検索語

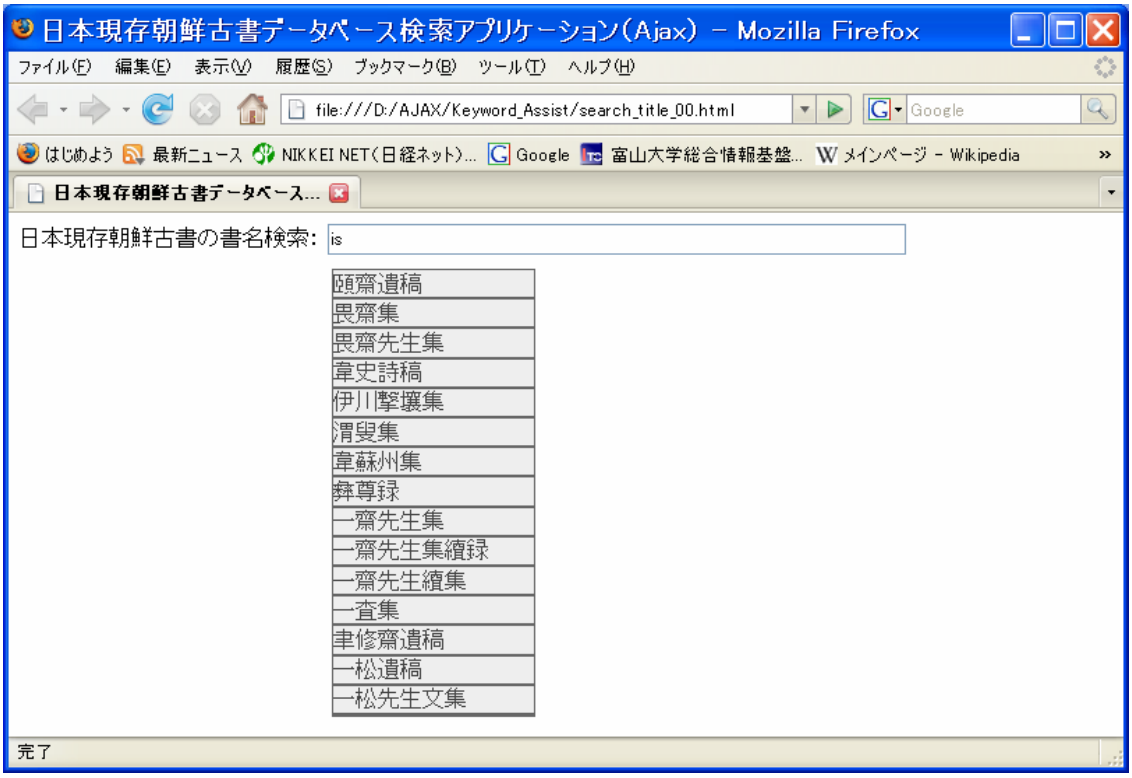


図 3.4.6　書名「is」を入力して候補を表示

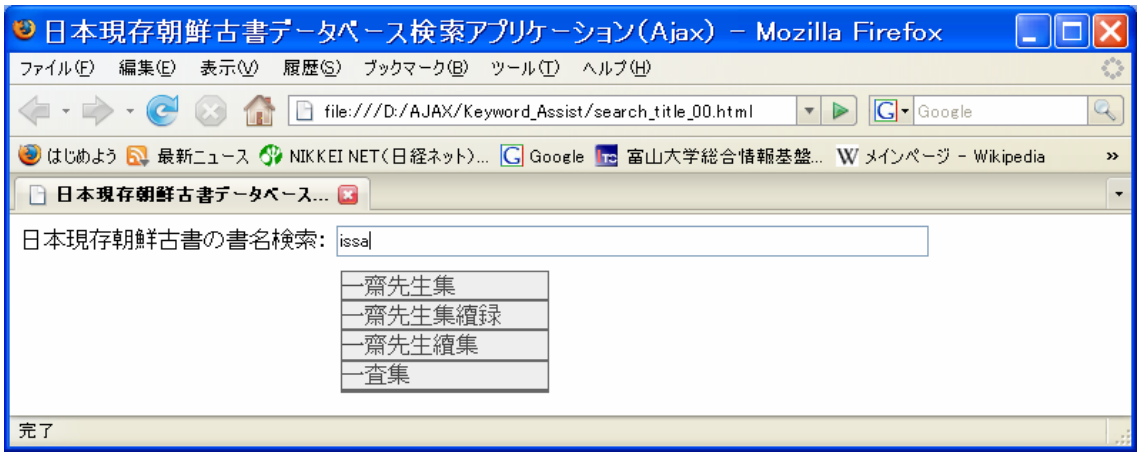


図 3.4.7　書名「issa」を入力して候補を絞り表示

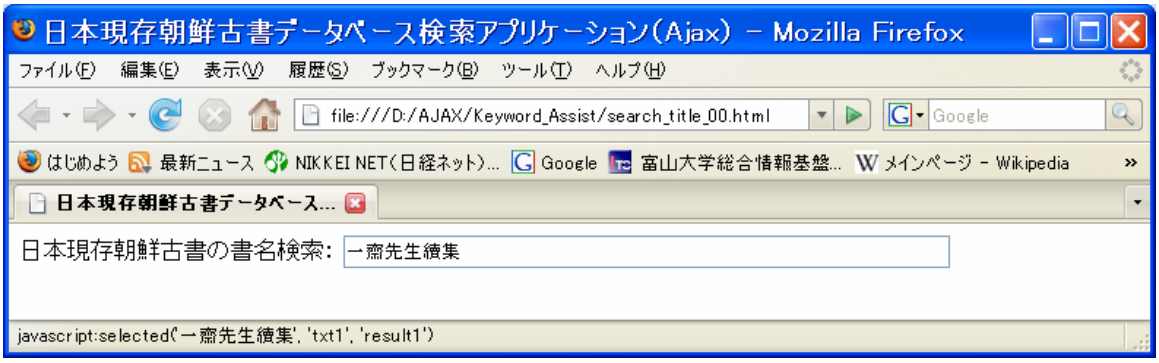


図 3.4.8　「一齋先生續集」をクリックして入力

入力域に該当書名のフル書名が入る（図 3.4.8）．撰者名の場合は図 3.4.9 のとおり同様に絞り込むことができる．

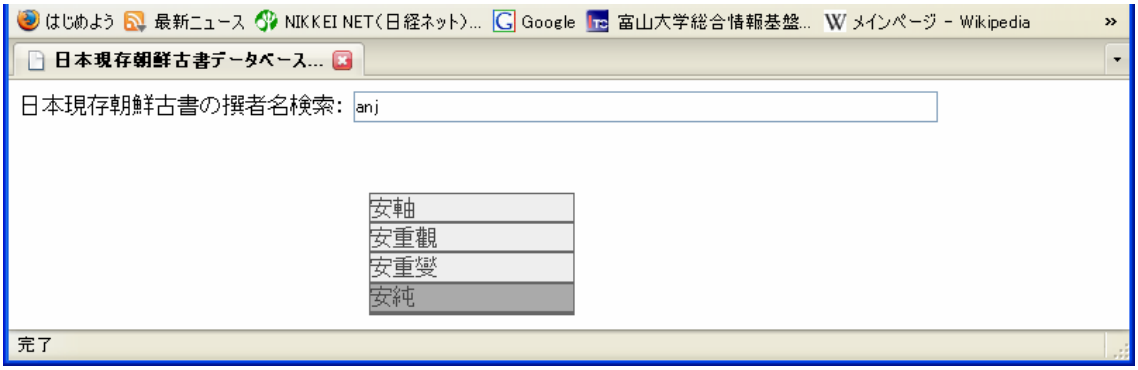


図 3.4.9　「anj」で撰者名「安純」を表示しクリック

３．４．５　ツールの問題点

この検索語類推支援ツールの問題点は，日本語入力における IME の仲介の扱い方とキーワードの任意一致の扱いである．

３．５ Ajax による CJK 用 IME ツールの開発

Ajax による汎用の IME ツールは，次の URL で提供されている．このツールにヒントを得て，検索語類推支援ツールをリンクさせて，CJK 用の IME ツールとして完成させたい．

１）Ajax IME: Web-based Japanese Input Method　http://ajaxime.chasen.org/

図 3.5.1　Ajax による IME ツールの例「Ajax IME: Web-based Japanese Input Method」

２）Ajax を使った 日本語 IME　http://www.chasen.org/~taku/software/ajax/ime/

図 3.5.2　Ajax による日本語 IME ツールの例「Ajax を使った日本語 IME」

3）Ajax を使った 日本語 IME + KWIC

http://www.chasen.org/~taku/software/ajax/imekwic/

4）Ajax を使った手書き文字認識

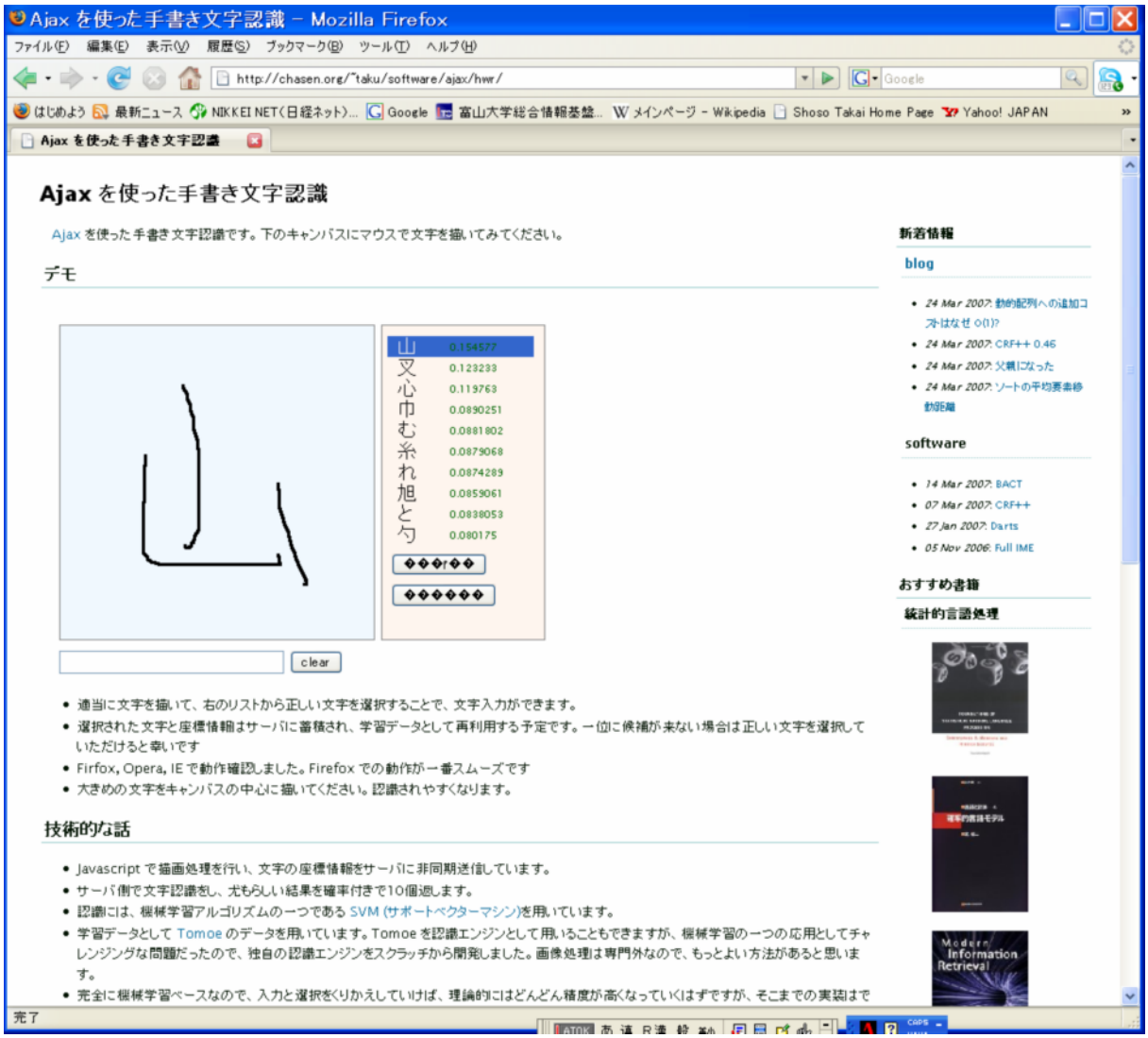


図 3.5.3　Ajax による手書き IME ツールの例「Ajax を使った手書き文字認識」

この IME ツールが完成すれば，CJK 用に，中国に対しては Pinyin 方式，韓国に対しては McCune Reischauer 方式，日本では Hepburn 方式などのローマ字入力により，CJK-Xterm エミュレータのように，１字入力毎に検索語を推測していく機能を有する計画で，各国語専用の IME (Input Method Editor:入力方式編集プログラム＝かな漢字変換などの事前処理プログラム) が不要で，Unicode 漢字が入力できるようになる．

３．６ Ajax 技法による日本現存朝鮮古書印影画像データベース検索システムの開発

３．６．１　原文画像データベース検索システムとの統合開発

印影画像は原文画像データベースの構造とほぼ同じであるため，原文画像データベースへ統合することとして，開発を一本化した．

印影画像の切り出しは絵画の場合は問題がないが，蔵書印の場合は原文文字との重なりが多く，印影を切り出しても，大部分は原文文字と重なり合っているため，正確な画像を切り出すことができない状況である．

次の画像は絵画や掛け軸の画像であり，印影の切り出しは絵画の Image-ID=70001_00.tif と書画 Image-ID=70007_00.tif では印影と絵画や文字との重なりが無く，Image-ID=80062_00.tif の書画では印影と文字が重なりあっているため，正確な著者の印影の切り出しは困難を極める．

Image-ID=70001_00.tif

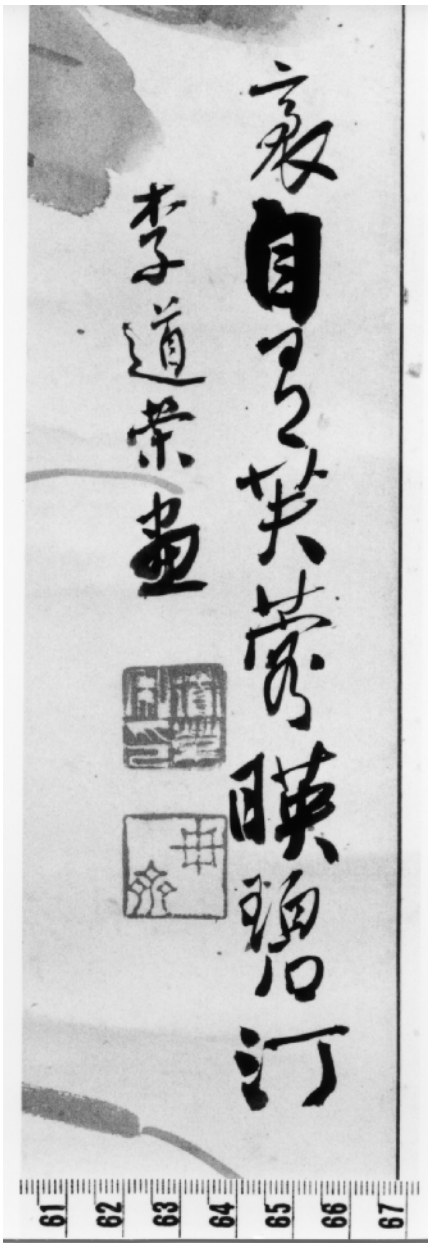


Image-ID=70007_00.tif

Image-ID=80062_00.tif

３．６．２　所蔵印の切り出し対策

所蔵印については，下記の画像のとおり，印影の切り出しは更に困難であることが分かる．

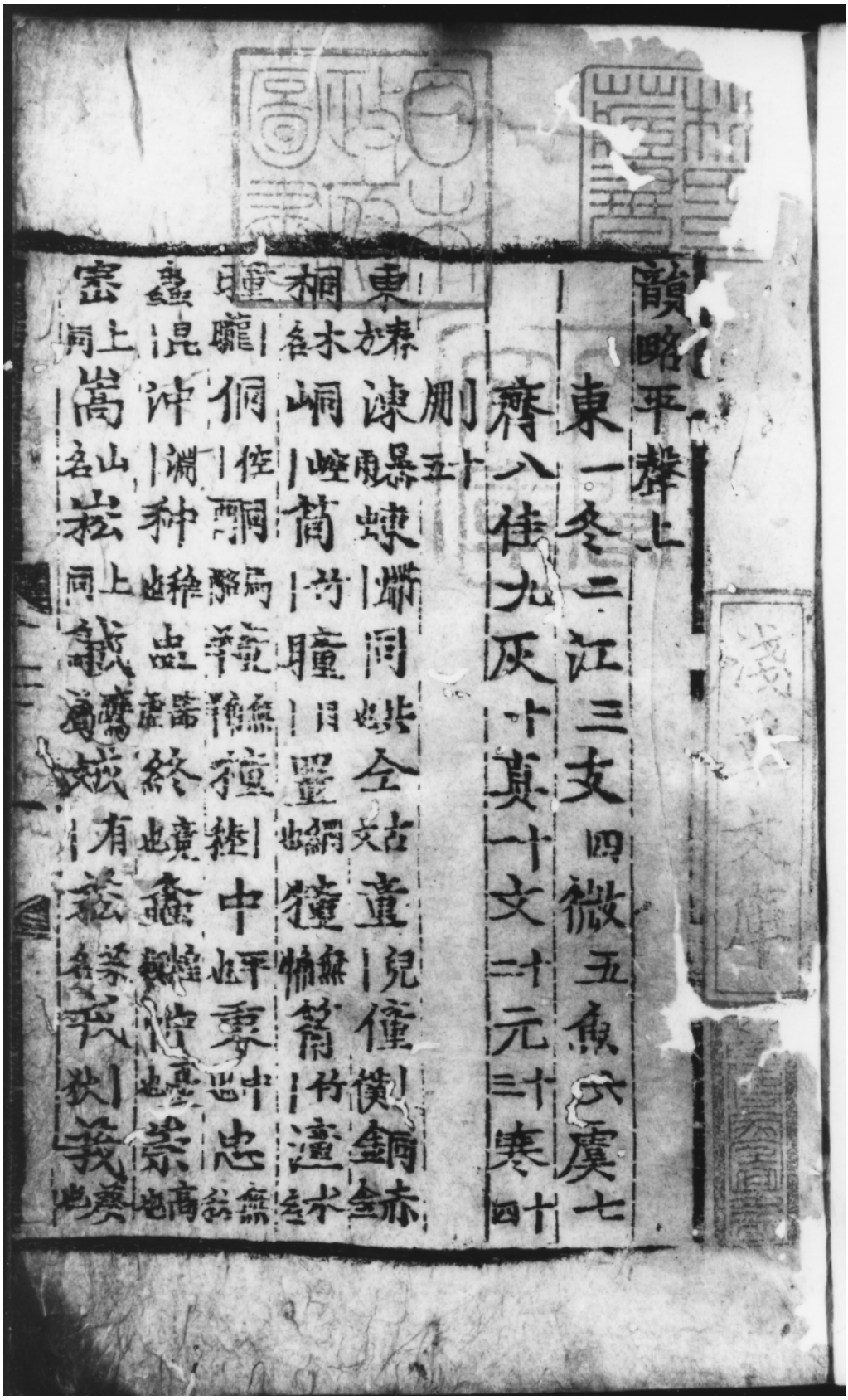


Image-ID=30001_00.png

Image-ID=30002_00.png

以上の様な事情から，所蔵印の印影画像の切り出しは，解決策を模索中であり，対策が見つかるまで開発を休止することとして，他に必要なデータベースの開発を優先することにした.

４．今後課題

今後の課題としては，現在画像毎に別々にある同一古書の数十～数百ページに及ぶ原文画像を格子状に自動分割して連続配置するサブ・システムの構築，現 DOKB 検索システムとのこの古書原文画像システムとの統合，原文画像のメタ・データからの検索システムの開発，そして研究者向けには，必要な画像への付箋の添付と簡単な取り出し方法などの機能が必要と考えられる．

５．まとめ

筆者等は DOKB における User Interface に画面遷移を伴わない Ajax 技法を使用した検索支援ツールである Unicode 漢字入力支援ツール，検索語類推支援ツール，CJK 用 IME ツールと古書原文画像・印影画像閲覧システム及び刻手名データベース検索システムの開発と実装を試みた．いつの日かこのような Ajax 技法による「究極の古書原文画像と書誌情報を対象とする WebDB 情報システム」が登場するものと考えられる．

この中で，Unicode 漢字入力支援ツールは現行 DOKB システムへの実装を完了し，刻手名データベース検索システムは DOKB と独立に KOKUSHU システムとして完成し, 提供するに至った．

検索語類推支援ツール，CJK 用 IME ツールと古書原文画像閲覧システム（印影画像データベースを含む）は開発途上にあり，鋭意開発を継続中である．

今後の課題としては，(1) 原文画像とそのメタ・データの蓄積，DOKB 本体との連携統合，(2) 原文画像の縮小拡大表示スライダーと表示画像切り出しと提供の方法，(3) 書誌情報検索結果の PDF 表示，とりわけマスター・データの窃盗からの防衛方法付加，などである．なお，古書データベースの Unicode 文字における欠字の問題は，Extension C の制定によって，ほぼ 100%が表示できるようになると期待している[8].

終わりに，この朝鮮古書の書誌情報 DB システムの完成によって，公開を待ち望む韓国をはじめ，全世界の朝鮮及び朝鮮本の研究者が，世界に居ながら古書の発掘，同定及び内容に関する研究が可能となることを確信する．

６．謝辞

DOKB データベースの作成及び機能付加，朝鮮古書原文画像のデータベース化に当たって，書誌情報，データの分類方法，記法，索引の採取方法，原文画像フィルム・写真など，本データベースに全体に関する情報の提供と相談を受けた前富山大学人文学部の藤本幸夫教授．これらのデータ入力と修正に献身的な努力をしてくれた越野，洲崎，木戸，竹澤の女性スタッフ．ここに記して感謝の意を表する．

なお，本研究は日本学術振興会科学研究費補助金（基盤研究（C）：課題番号 18500079）を受けて実施した．

７．参考文献

[1]高井正三，日本現存朝鮮古書データベースの国際対応化の方法，富山大学総合情報基盤センター広報，Vol.3，29-38，2006

[2]Justin Gehtland, Ben Galbraith, Dion Almaer 著，宮川達彦監訳，加藤慶彦訳，Pragmatic Ajax：A Web 2.0 Primer（日本語訳名：実践 Ajax―Web2.0 アプリケーション開発への手引き―），オライリー・ジャパン，ISBN4-87311-301-6, 2006

[3]Jesse James Garrett，Ajax: A New Approach to Web Applications,
http://www.adaptivepath.com/ideas/essays/
archives/000385.php

[4]W3C.Document Object Model Specifications.
http://www.w3.org/DOM/DOMTR.

[5] W3C. XHTML 1.1 - Module-based XHTML.
http://www.w3.org/TR/xhtml11/.

[6] Google, Inc. Google Suggest：
http://www.google.co.jp/webhp?complete=1&hl=ja

[7] Google, Inc. Google マップ：
http://maps.google.co.jp/

[8]師茂樹，秋山陽一郎他，多言語情報処理，漢字文献情報処理研究，漢字文献情報処理研究会編集，第６号，157-173，ISBN4-87220-098-5，好文出版，2005

８．発表した論文集，解説集

2006 年～2008 年までに発表した論文リスト及び発表集リストを次に掲げる．

（論文１）p.50-55

・高井正三，喜多啓太，米田恭章，Java フレームワークによる古文書データベース・システムの開発，学術情報処理研究，Vol.10，65-70，2006.

（発表１）論文１と同じ（PowerPoint 資料 p.67-84）

・高井正三，喜多啓太，米田恭章，Java フレームワークによる古文書データベース・システムの開発，学術情報処理研究集会，岩手大学工学部，2006.09.22.

（発表 2）p.56 （PowerPoint 資料 p.85-90）

・喜多啓太，米田恭章，高井正三，Ajax による古文書画像閲覧システムの一提案，平成 18 年度電気関係学会北陸支部連合大会，講演論文集，E-81，金沢工業大学，2006.09.17.

（発表 3）p.57 （PowerPoint 資料 p.91-96）

・米田恭章，喜多啓太，高井正三，Ajax による古文書向け文字検索支援ツールの一提案，平成 18 年度電気関係学会北陸支部連合大会，講演論文集，E-82，金沢工業大学，2006.09.17.

（発表 4）p.58 （PowerPoint 資料 p.97-106）

・高井正三，喜多啓太，米田恭章，Ajax による古文書データベース検索語類推支援ツールの一提案，平成 18 年度電気関係学会北陸支部連合大会，講演論文集，E-83，金沢工業大学，2006.09.17.

（論文２）p.59-66

・高井正三，喜多啓太，米田恭章，Ajax 技法による日本現存朝鮮古書 DB 入力支援と画像 DB システムの開発，富山大学総合情報基盤センター広報，Vol.5，55-62，2008

別刷論文 Page 50 - Page 106 削除

９．資料

資料一覧

［資料９－１］Ajax 技法による Unicode 漢字入力支援ツール開発ソース・コード

１）searchUnicode.js

２）unisearch_cookie.js

３）unisearch_dom.js

４）unisearch_eve.js

５）unisearch_gene.js

６）unisearch_search.js

７）searchcode.pl

なお，Ajax 対応 JavaScript ライブラリ prototype.js を使用している.

［資料９－２］刻手名データベース検索システム開発ソース・コード

１）kokushu-db：刻手データを表すモジュール

２）kokushu-ejb：刻手データを検索・取得するモジュール

３）kokushu-indexer：刻手データベースから全文検索用のインデックスを作成するアプリケーション

４）kokushu-webapp：刻手データを検索するウェブアプリケーション

５）kokushu-webapp-user-db：刻手データを検索するウェブアプリケーションの登録ユーザを表すモジュール

６）koiushu-webapp-user-manager：刻手データを検索するウェブアプリケーションの登録ユーザを管理するアプリケーション

［資料９－３］印影・原文画像データベース検索システム開発ソース・コード（開発中のもの）

［資料９－４］古文書等印影データベースのデータ入力要領（2003.05.13 作成版）

［資料９－１］Ajax 技法による Unicode 漢字入力支援ツール開発ソース・コード

1）searchUnicode.js

searchUnicode.js　　Page 1

```
 1 /**
 2  *各unisearch_xxx.jsとbrowserChecker.js、Selection.jsを統合したもの。
 3  *browserChecker及びSelection.jsは他所よりお借りしたものです。
 4  *システムを動かすにはprototype.jsも必要となる。
 5  **/
 6
 7
 8
 9 // Cross Browser selectionStart/selectionEnd
10 // Version 0.2
11 // Copyright (c) 2005-2007 KOSEKI Kengo
12 //
13 // This script is distributed under the MIT licence.
14 // http://www.opensource.org/licenses/mit-license.php
15
16 function Selection(textareaElement) {
17     this.element = textareaElement;
18 }
19
20 Selection.prototype.create = function() {
21     if (document.selection != null && this.element.selectionStart == null) {
22         return this._ieGetSelection();
23     } else {
24         return this._mozillaGetSelection();
25     }
26 }
27
28 Selection.prototype._mozillaGetSelection = function() {
29     return {
30         start: this.element.selectionStart,
31         end: this.element.selectionEnd
32     };
33 }
34
35 Selection.prototype._ieGetSelection = function() {
36     this.element.focus();
37
38     var range = document.selection.createRange();
39     var bookmark = range.getBookmark();
40
41     var contents = this.element.value;
42     var originalContents = contents;
43     var marker = this._createSelectionMarker();
44     while(contents.indexOf(marker) != -1) {
45         marker = this._createSelectionMarker();
46     }
47
48     var parent = range.parentElement();
49     if (parent == null || parent.type != "textarea") {
50         return { start: 0, end: 0 };
51     }
52     range.text = marker + range.text + marker;
53     contents = this.element.value;
54
55     var result = {};
56     result.start = contents.indexOf(marker);
57     contents = contents.replace(marker, "");
58     result.end = contents.indexOf(marker);
59
60     this.element.value = originalContents;
61     range.moveToBookmark(bookmark);
62     range.select();
```

```
63
64      return result;
65 }
66
67 Selection.prototype._createSelectionMarker = function() {
68      return "##SELECTION_MARKER_" + Math.random() + "##";
69 }
70
71 // 究極の JavaScript クライアント判別 バージョン 3.03
72
73 // (C) Netscape Communications 1999-2001.  再利用と変更を認めます。
74
75 // 修正 17 May 99 :  is_nav5up と is_ie5up を追加(後述).
76
77 // 修正 20 Dec 00 :  is_gecko を追加、 is_nav5up を is_nav6up に変更
78
79 //                   IE5.5 Opera4&5 HotJava3 AOLTV のサポートも追加
80
81 // 修正 22 Feb 01 :  IE 5.x, Opera 4, の JavaScript 判断を修正
82
83 //                   Opera 5 の判断を修正
84
85 //                   winME と win2k のサポートを追加
86
87 //                   browser-type-oo.js と同期
88
89 //          [訳註] 非オブジェクト指向バージョンを修正していったあと最後にオ
90 ブジェクト指向バージョン
91
92 //                   にも変更点を纏めて一気に反映させた、ということではないか
93 と予想します。(^^;
94
95 // 修正 26 Mar 01 :  Opera の判断を修正
96
97 // 修正 02 Oct 01 :  IE6 の判断を追加
98
99
100 // JavaScript クライアントについて知りたいがなかなか聞けないもの全て。
101
102 // "Is" は "is" オブジェクトのコンストラクタ関数で、以下を示すプロパティを持ち
103 ます:
104
105 // (1) ブラウザベンダ:
106
107 //     is.nav, is.ie, is.opera, is.hotjava, is.webtv, is.TVNavigator, is.AOLTV
108
109
110 // (2) ブラウザージョン番号:
111
112 //     is.major (integer indicating major version number: 2, 3, 4 ...)
113
114 //     is.minor (float   indicating full  version number: 2.02, 3.01, 4.04 ...
115 )
116
117 // (3) ブラウザベンダとメジャーバージョン
118
119 //     is.nav2, is.nav3, is.nav4, is.nav4up, is.nav6, is.nav6up, is.gecko, is.
120 ie3,
121
122 //     is.ie4, is.ie4up, is.ie5, is.ie5up, is.ie5_5, is.ie5_5up, is.ie6, is.ie
123 6up, is.hotjava3, is.hotjava3up
124
```

```
125 // (4) JavaScript バージョン番号:
126
127 //     is.js (float indicating full JavaScript version number: 1, 1.1, 1.2 ...
128 )
129
130 // (5) OS プラットフォームとバージョン:
131
132 //     is.win, is.win16, is.win32, is.win31, is.win95, is.winnt, is.win98, is.
133 winme, is.win2k
134
135 //     is.os2
136
137 //     is.mac, is.mac68k, is.macppc
138
139 //     is.unix
140
141 //     is.sun, is.sun4, is.sun5, is.suni86
142
143 //     is.irix, is.irix5, is.irix6
144
145 //     is.hpux, is.hpux9, is.hpux10
146
147 //     is.aix, is.aix1, is.aix2, is.aix3, is.aix4
148
149 //     is.linux, is.sco, is.unixware, is.mpras, is.reliant
150
151 //     is.dec, is.sinix, is.freebsd, is.bsd
152
153 //     is.vms
154
155 //
156
157 // ユーザエージェント文字列の詳細については次を参照してください。
158
159 // http://www.it97.de/JavaScript/JS_tutorial/bstat/navobj.html
160
161 // http://www.it97.de/JavaScript/JS_tutorial/bstat/Browseraol.html
162
163 //
164
165 // 注: Nav5 や IE5 (またはそれ以降) がリリースされてもNav4 や IE4 の
166
167 // コードを"首"にしたりはしたくないでしょうから、条件分岐する際、将来の
168
169 // バージョンでも動作して欲しいコードでは、is.nav4 や is.ie4 ではなく
170
171 // is.nav4up (Nav4以降) や is.ie4up (IE4 以降) を使用してください。
172
173 //[訳註] 原文のまま訳しましたが、この記載は更新忘れているもののようです。
174
175 //        非オブジェクト指向バージョンでは以下のような現状に即した記述になって
176 います。
177
178 //        | 注: 新しいブラウザがリリースされてもNav4 や IE4 のコードを"首"に
179
180 //        | したりはしたくないでしょうから、条件分岐する際、将来のバージョン
181
182 //        | でも動作して欲しいコードでは、is_ie5 や is_opera5 ではなく
183
184 //        | is_ie5up (IE5以降)や is_opera5up (Opera5.0 以降) を使用してくださ
185 い。
186
```

```
187
188
189 function Is ()
190
191 {   // テストを簡単にするために全文字列を小文字に変換
192
193     var agt=navigator.userAgent.toLowerCase();
194
195
196     // *** ブラウザージョン ***
197
198     // 注: IE5 ではこの値で 4 が返されるので IE5 の判断には is_ie5up を使用す
199 る。
200
201     this.major = parseInt(navigator.appVersion);
202
203     this.minor = parseFloat(navigator.appVersion);
204
205
206     // 注: Opera と WebTV は Navigator のマネをしますが、厳密に判断します。
207
208     // マネをするのを認めるなら、opera と webtv のコードを除外してください。
209
210     this.nav  = ((agt.indexOf('mozilla')!=-1) && (agt.indexOf('spoofer')==-1)
211
212                 && (agt.indexOf('compatible') == -1) && (agt.indexOf('opera')=
213 =-1)
214
215                 && (agt.indexOf('webtv')==-1) && (agt.indexOf('hotjava')==-1))
216 ;
217
218     this.nav2 = (this.nav && (this.major == 2));
219
220     this.nav3 = (this.nav && (this.major == 3));
221
222     this.nav4 = (this.nav && (this.major == 4));
223
224     this.nav4up = (this.nav && (this.major >= 4));
225
226     this.navonly      = (this.nav && ((agt.indexOf(";nav") != -1) ||
227
228                           (agt.indexOf("; nav") != -1)) );
229
230     this.nav6 = (this.nav && (this.major == 5));
231
232     this.nav6up = (this.nav && (this.major >= 5));
233
234     this.gecko = (agt.indexOf('gecko') != -1);
235
236
237
238     this.ie     = ((agt.indexOf("msie") != -1) && (agt.indexOf("opera") == -1)
239 );
240
241     this.ie3    = (this.ie && (this.major < 4));
242
243     this.ie4    = (this.ie && (this.major == 4) && (agt.indexOf("msie 4")!=-1)
244  );
245
246     this.ie4up  = (this.ie  && (this.major >= 4));
247
248     this.ie5    = (this.ie && (this.major == 4) && (agt.indexOf("msie 5.0")!=-
```

```
249 1) );
250
251     this.ie5_5  = (this.ie && (this.major == 4) && (agt.indexOf("msie 5.5") !=
252 -1));
253
254     this.ie5up  = (this.ie  && !this.ie3 && !this.ie4);
255
256     this.ie5_5up =(this.ie && !this.ie3 && !this.ie4 && !this.ie5);
257
258     this.ie6    = (this.ie && (this.major == 4) && (agt.indexOf("msie 6.")!=-1
259 ) );
260
261     this.ie6up  = (this.ie  && !this.ie3 && !this.ie4 && !this.ie5 && !this.ie
262 5_5);
263
264
265     // 既知のバグ: AOL4 では IE3 が組み込まれている場合や最初に開かれたブラウ
266 ザ
267
268     // ウィンドウである場合には false を返します。であるから、is_aol, is_aol3,
269
270
271     // is_aol4 は 100% 信頼できるものではありません。
272
273     this.aol    = (agt.indexOf("aol") != -1);
274
275     this.aol3   = (this.aol && this.ie3);
276
277     this.aol4   = (this.aol && this.ie4);
278
279     this.aol5   = (agt.indexOf("aol 5") != -1);
280
281     this.aol6   = (agt.indexOf("aol 6") != -1);
282
283
284     this.opera = (agt.indexOf("opera") != -1);
285
286     this.opera2 = (agt.indexOf("opera 2") != -1 || agt.indexOf("opera/2") != -
287 1);
288
289     this.opera3 = (agt.indexOf("opera 3") != -1 || agt.indexOf("opera/3") != -
290 1);
291
292     this.opera4 = (agt.indexOf("opera 4") != -1 || agt.indexOf("opera/4") != -
293 1);
294
295     this.opera5 = (agt.indexOf("opera 5") != -1 || agt.indexOf("opera/5") != -
296 1);
297
298     this.opera5up = (this.opera && !this.opera2 && !this.opera3 && !this.opera
299 4);
300
301
302     this.webtv = (agt.indexOf("webtv") != -1);
303
304
305     this.TVNavigator = ((agt.indexOf("navio") != -1) || (agt.indexOf("navio_ao
306 ltv") != -1));
307
308     this.AOLTV = this.TVNavigator;
309
310
```

```
311     this.hotjava = (agt.indexOf("hotjava") != -1);
312
313     this.hotjava3 = (this.hotjava && (this.major == 3));
314
315     this.hotjava3up = (this.hotjava && (this.major >= 3));
316
317
318     // *** JAVASCRIPT バージョン ***
319
320     if (this.nav2 || this.ie3) this.js = 1.0;
321
322     else if (this.nav3) this.js = 1.1;
323
324     else if (this.opera5up) this.js = 1.3;
325
326     else if (this.opera) this.js = 1.1;
327
328     else if ((this.nav4 && (this.minor <= 4.05)) || this.ie4) this.js = 1.2;
329
330     else if ((this.nav4 && (this.minor > 4.05)) || this.ie5) this.js = 1.3;
331
332     else if (this.hotjava3up) this.js = 1.4;
333
334     else if (this.nav6 || this.gecko) this.js = 1.5;
335
336     // 注: 将来的には、新しいバージョンのJSが出たらこのコードを更新します。
337
338     // 今のところ、将来のバージョンの Nav や IE が"少なくとも" JS 1.x 互換
339
340     // であることを示します。JS バージョン互換のチェックには常に > や >=
341
342     // を使用するようにしてください。
343
344     else if (this.nav6up) this.js = 1.5;
345
346     // 注: マックでの ie5up は 1.4
347
348     else if (this.ie5up) this.js = 1.3
349
350
351     // HACK: 他のブラウザは分かりません。JS バージョンチェックは > や>= で。
352
353     else this.js = 0.0;
354
355
356     // *** プラットフォーム ***
357
358     this.win   = ( (agt.indexOf("win")!=-1) || (agt.indexOf("16bit")!=-1) );
359
360     // 注: Opera 3.0 では Win32 環境全てでユーザエージェント文字列に "Windows
361 95/NT4"
362
363     // が含まれており、Win95 と WinNT の区別が出来ません。
364
365     this.win95 = ((agt.indexOf("win95")!=-1) || (agt.indexOf("windows 95")!=-1
366 ));
367
368
369     // 16 bit バージョンと思われる。
370
371     this.win16 = ((agt.indexOf("win16")!=-1) ||
372
```

```
373                 (agt.indexOf("16bit")!=-1) || (agt.indexOf("windows 3.1")!=-1)
374 ||
375
376                 (agt.indexOf("windows 16-bit")!=-1) );
377
378
379     this.win31 = ((agt.indexOf("windows 3.1")!=-1) || (agt.indexOf("win16")!=-
380 1) ||
381
382                     (agt.indexOf("windows 16-bit")!=-1));
383
384
385     // 注: Win98 の信頼できる判断法は存在しないようです。次のようだから:
386
387     //      - Nav4.x 以前ではユーザエージェントで "Windows" だけしか得られない
388 。
389
390     //      - Win98 上の Mercury では 32 bit バージョンは "Win98" を返すが、
391
392     //        16 bit バージョンは "Win95" を返す。
393
394     this.win98 = ((agt.indexOf("win98")!=-1) || (agt.indexOf("windows 98")!=-1
395 ));
396
397     this.winnt = ((agt.indexOf("winnt")!=-1) || (agt.indexOf("windows nt")!=-1
398 ));
399
400     this.win32 = (this.win95 || this.winnt || this.win98 ||
401
402                     ((this.major >= 4) && (navigator.platform == "Win32")) ||
403
404                     (agt.indexOf("win32")!=-1) || (agt.indexOf("32bit")!=-1));
405
406
407
408     this.winme = ((agt.indexOf("win 9x 4.90")!=-1));
409
410     this.win2k = ((agt.indexOf("windows nt 5.0")!=-1));
411
412
413     this.os2   = ((agt.indexOf("os/2")!=-1) ||
414
415                     (navigator.appVersion.indexOf("OS/2")!=-1) ||
416
417                     (agt.indexOf("ibm-webexplorer")!=-1));
418
419
420     this.mac    = (agt.indexOf("mac")!=-1);
421
422     // hack: マックでの ie5 の JavaScript バージョン
423
424     if (this.mac && this.ie5up) this.js = 1.4;
425
426     this.mac68k = (this.mac && ((agt.indexOf("68k")!=-1) ||
427
428                                 (agt.indexOf("68000")!=-1)));
429
430     this.macppc = (this.mac && ((agt.indexOf("ppc")!=-1) ||
431
432                                  (agt.indexOf("powerpc")!=-1)));
433
434
```

```
435     this.sun   = (agt.indexOf("sunos")!=-1);
436
437     this.sun4  = (agt.indexOf("sunos 4")!=-1);
438
439     this.sun5  = (agt.indexOf("sunos 5")!=-1);
440
441     this.suni86= (this.sun && (agt.indexOf("i86")!=-1));
442
443     this.irix  = (agt.indexOf("irix") !=-1);    // SGI
444
445     this.irix5 = (agt.indexOf("irix 5") !=-1);
446
447     this.irix6 = ((agt.indexOf("irix 6") !=-1) || (agt.indexOf("irix6") !=-1))
448 ;
449
450     this.hpux  = (agt.indexOf("hp-ux")!=-1);
451
452     this.hpux9 = (this.hpux && (agt.indexOf("09.")!=-1));
453
454     this.hpux10= (this.hpux && (agt.indexOf("10.")!=-1));
455
456     this.aix   = (agt.indexOf("aix") !=-1);      // IBM
457
458     this.aix1  = (agt.indexOf("aix 1") !=-1);
459
460     this.aix2  = (agt.indexOf("aix 2") !=-1);
461
462     this.aix3  = (agt.indexOf("aix 3") !=-1);
463
464     this.aix4  = (agt.indexOf("aix 4") !=-1);
465
466     this.linux = (agt.indexOf("inux")!=-1);
467
468     this.sco   = (agt.indexOf("sco")!=-1) || (agt.indexOf("unix_sv")!=-1);
469
470     this.unixware = (agt.indexOf("unix_system_v")!=-1);
471
472     this.mpras    = (agt.indexOf("ncr")!=-1);
473
474     this.reliant  = (agt.indexOf("reliantunix")!=-1);
475
476     this.dec   = ((agt.indexOf("dec")!=-1) || (agt.indexOf("osf1")!=-1) ||
477
478                   (agt.indexOf("dec_alpha")!=-1) || (agt.indexOf("alphaserver"
479 )!=-1) ||
480
481                   (agt.indexOf("ultrix")!=-1) || (agt.indexOf("alphastation")!
482 =-1));
483
484     this.sinix = (agt.indexOf("sinix")!=-1);
485
486     this.freebsd = (agt.indexOf("freebsd")!=-1);
487
488     this.bsd = (agt.indexOf("bsd")!=-1);
489
490     this.unix  = ((agt.indexOf("x11")!=-1) || this.sun || this.irix || this.hp
491 ux ||
492
493                   this.sco ||this.unixware || this.mpras || this.reliant ||
494
495                   this.dec || this.sinix || this.aix || this.linux || this.bsd
496 || this.freebsd);
```

```
497
498
499     this.vms    = ((agt.indexOf("vax")!=-1) || (agt.indexOf("openvms")!=-1));
500
501 }
502
503
504 var is;
505
506 var isIE3Mac = false;
507
508 // この節は Mac の IE3 の為に特別に用意したものです。
509
510
511 if ((navigator.appVersion.indexOf("Mac")!=-1) && (navigator.userAgent.indexOf(
512 "MSIE")!=-1) &&
513
514
515 (parseInt(navigator.appVersion)==3))
516
517        isIE3Mac = true;
518
519 else   is = new Is();
520
521
522 // JavaScript隠蔽終了
523
524
525 //システムのユーザー操作に対する非同期通信及びDOM操作に関する処理
526
527 //部首番号
528 var radicalNumber = -1;
529 var nodeList = new Array();
530 //コードポイントで表示するか
531 var dispCode = false;
532 //コードポイントの位置
533 var cpListNum = 0;
534 //一度に表示する漢字数
535 var dispNumber = 10;
536 //表示している漢字を格納
537 var cpTable = new Array(dispNumber);
538 var cpListTable = new Array();
539 //最後にフォーカスされたtextAreaを格納
540 var focusedArea;
541
542 //フォルダの位置
543 var cgi = "http://stl30.itc.u-toyama.ac.jp/cgi-bin/";
544 var uri = "http://stl30.itc.u-toyama.ac.jp/dokb/ime/";
545
546 //ファイル要求
547 //部首ナンバーからその部首に属している文字のストロークリストを取得
548 function getStroke(){
549        var req = new Ajax.Request( cgi+"searchcode.pl?"+radicalNumber, { meth
550 od: 'GET', onComplete: gene_stroke});//, onException: function(request,excepti
551 on) { document.write(exception)} });
552 }
553
554 //部首ナンバーとストロークナンバーから属している文字のコードポイントリストを取
555 得
556 function getCodepoint(strokeNumber){
557        var req = new Ajax.Request( cgi+"searchcode.pl?"+radicalNumber+"&"+str
558 okeNumber, { method: 'GET', onComplete: gene_code});//, onException: function(
```

```
559 request,exception) { document.write(exception)} });
560 }
561
562 //部首が選択された場合
563 function radical_function(){
564         var selectedElement = $('radical');
565
566         radicalNumber = selectedElement.selectedIndex+1;
567         if(radicalNumber > 0){
568                 getStroke();
569         }else {
570                 recreateSelectDummy('null');
571         }
572 }
573
574 //ストロークリストのXMLを取得し、画数のセレクトボックスを生成
575 function gene_stroke(request){
576         var tempXML = request.responseXML;
577
578         var root = tempXML.documentElement;
579         var strokeList = new Array();
580
581         strokeList = root.getElementsByTagName('stroke_num');
582         recreateSelectDummy(strokeList);
583 }
584
585 //コードポイントリストのXMLを取得し、コード表を生成
586 function gene_code(request){
587         var tempXML = request.responseXML;
588         var root = tempXML.documentElement;
589
590         cpListTable = root.getElementsByTagName('codepoint');
591         setCodeTable();
592 }
593
594 //コードのtableを生成する
595 function setCodeTable(){
596                 var addArea = $('imgDummy');
597                 var tableNode = document.createElement('table');
598                 var tbodyNode = document.createElement('tbody');
599                 var trNode = document.createElement('tr');
600                 tableNode.id = "table";
601                 tableNode.border = "1";
602                 tableNode.cellspacing = "0";
603                 tbodyNode.appendChild(trNode);
604                 tableNode.appendChild(tbodyNode);
605                 addArea.appendChild(tableNode);
606
607                 //cpListTable = codeList;
608                 var i;
609                 for(i=cpListNum;i<dispNumber && i<cpListTable.length;i++){
610                         codepoint = cpListTable[i].firstChild.nodeValue;
611                         cpTable[i] = codepoint;
612                         var tdNode = gene_imgNode(codepoint,trNode);
613                         setInsEvent(i,tdNode);
614                 }
615                 cpListNum = i;
616
617
618                 if(cpListNum<cpListTable.length){
619                         var tw = $('nextB');
620
```

```
621                         if(is.gecko || is.opera){
622                                 tw.addEventListener('click',next_code,false);
623                         }
624                         else{
625                                 tw.attachEvent('onclick',next_code);
626                         }
627                         tw.src = uri+"arrowR.png"
628                         tw.className = "onCursor";
629                 };
630 }
631
632 //画数が選択された場合
633 function stroke_function(){
634
635         //コードポイント抜き出し
636         var codeList;
637         var selectElement = $('selectDummy');
638         var indexNumber = selectElement.selectedIndex;
639         recreateImgDummy();
640
641         var addArea = $('imgDummy');
642
643         var loc = document.createElement('left');
644         addArea.appendChild(loc);
645
646         //back矢印画像を設置
647         var creBw = document.createElement('img');
648         creBw.src = uri+"narrowL.png";
649         creBw.width = 25;
650         creBw.height = 25;
651         creBw.id = "backB";
652         creBw.hspace = 10;
653         loc.appendChild(creBw);
654
655         //next矢印画像を設置
656         var creTw = document.createElement('img');
657         creTw.src = uri+"narrowR.png"
658         creTw.width = 25;
659         creTw.height = 25;
660         creTw.id = "nextB";
661         creTw.hspace = 10;
662         loc.appendChild(creTw);
663
664         var strokeNumber = -1;
665
666         //選択された画数がALL(0)かそれ以外(>0)か
667         if(indexNumber >0){
668                 strokeNumber = selectElement[indexNumber].firstChild.nodeValue
669 ;
670         }
671         getCodepoint(strokeNumber);
672 }
673
674 //右矢印がクリックされた場合
675 function next_code(){
676         var selectElement = $('selectDummy');
677         var indexNumber = selectElement.selectedIndex-1;
678
679         var tableNode = $('table');
680
681         var children = tableNode.childNodes;
682
```

```
683         for(var j=0;j<children.length;j++){
684                 tableNode.removeChild(children[j]);
685         }
686
687         var tbodyNode = document.createElement('tbody');
688         var trNode = document.createElement('tr');
689         tableNode.appendChild(tbodyNode);
690         tbodyNode.appendChild(trNode);
691
692         var i;
693         for(i=cpListNum;i<cpListNum+dispNumber && i<cpListTable.length;i++){
694                 codepoint = cpListTable[i].firstChild.nodeValue;
695                 cpTable[i-cpListNum] = codepoint;
696                 var tdNode = gene_imgNode(codepoint,trNode);
697                 setInsEvent(i-cpListNum,tdNode);
698         }
699
700         cpListNum = i;
701         if(cpListNum>=dispNumber){
702                 var bw = $('backB');
703                 if(is.gecko || is.opera){
704                         bw.addEventListener('click',back_code,false);
705                 }
706                 else{
707                         bw.detachEvent('onclick',back_code);
708                         bw.attachEvent('onclick',back_code);
709                 }
710                 bw.src = uri+"arrowL.png"
711                 bw.className = "onCursor";
712         }
713         if(cpListNum>=cpListTable.length){
714                 var tw = $('nextB');
715                 if(is.gecko || is.opera){
716                         tw.removeEventListener('click',next_code,false);
717                 }
718                 else{
719                         tw.detachEvent('onclick',next_code);
720                 }
721                 tw.src = uri+"narrowR.png"
722                 tw.className = "offCursor";
723         }
724         write_cookie();
725 }
726
727 //左矢印がクリックされた場合
728 function back_code(){
729         var selectElement = $('selectDummy');
730         var indexNumber = selectElement.selectedIndex-1;
731
732         var tableNode = $('table');
733
734         var children = tableNode.childNodes;
735
736         for(var j=0;j<children.length;j++){
737                 tableNode.removeChild(children[j]);
738         }
739
740         var tbodyNode = document.createElement('tbody');
741         var trNode = document.createElement('tr');
742         tableNode.appendChild(tbodyNode);
743         tbodyNode.appendChild(trNode);
744
```

```
745         var i;
746
747         //微調整
748         if(cpListNum==cpListTable.length&&cpListNum%dispNumber>0){
749                 cpListNum = cpListNum + dispNumber - cpListNum%dispNumber;
750         }
751
752         var tempBack = cpListNum-dispNumber*2;
753         if(tempBack<0){
754                 tempBack=0;
755         }
756
757         for(i=tempBack;i<tempBack+dispNumber && i<cpListTable.length;i++){
758                 codepoint = cpListTable[i].firstChild.nodeValue;
759                 cpTable[i-tempBack] = codepoint;
760                 var tdNode = gene_imgNode(codepoint,trNode);
761                 setInsEvent(i-tempBack,tdNode);
762         }
763
764         cpListNum = i;
765         if(cpListNum<=cpListTable.length){
766                 var tw = $('nextB');
767                 if(is.gecko || is.opera){
768                         tw.addEventListener('click', next_code,false);
769                 }
770                 else{
771                         tw.detachEvent('onclick', next_code);
772                         tw.attachEvent('onclick', next_code);
773                 }
774                 tw.src = uri+"arrowR.png"
775                 tw.className = "onCursor";
776         }
777
778         if(cpListNum<=dispNumber){
779                 var bw = $('backB');
780                 if(is.gecko || is.opera){
781                         bw.removeEventListener('click',back_code,false);
782                 }
783                 else{
784                         bw.detachEvent('onclick',back_code);
785                 }
786                 bw.src = uri+"narrowL.png"
787                 bw.className = "offCursor";
788         }
789
790         write_cookie();
791 }
792
793 //コードポイント表記のチェックボックスがクリックされた場合
794 function checkBox(){
795         write_cookie();
796 }
797
798 //個々のコードポイントの挿入するHTMLを生成
799 function gene_imgNode(codepoint,trNode){
800         var url;
801         url = uri+ "img/"+codepoint+".gif";
802         var img ="<img src=¥""+url+"¥">";
803         var tdNode = document.createElement('td');
804         var imgNode = document.createElement('img');
805         imgNode.src = url;
806         trNode.appendChild(tdNode);
```

```
807         tdNode.appendChild(imgNode);
808
809         var text = "¥&#x"+codepoint+";";
810         disp = $('dispWay');
811         if(disp.checked){
812                 var textNode = document.createTextNode(codepoint);
813         }
814         else{
815                 var textNode = document.createTextNode(text.unescapeHTML());
816         }
817         tdNode.appendChild(textNode);
818         tdNode.className = "onCursor";
819         return(tdNode);
820 }
821
822 //テキストエリアに文字を挿入
823 function setStr2TxtBox(codepoint){
824         var insCode = '&#x'+codepoint+';';
825         var top = focusedArea.scrollTop;
826         var selection = new Selection(focusedArea);
827         var s = selection.create();
828         var n = s.start;
829         var before = focusedArea.value.substring(0, n);
830         var after = focusedArea.value.substring(n, focusedArea.value.length);
831         focusedArea.value = before + insCode.unescapeHTML() + after;
832
833         if (is.ie) { // IEの場合
834                 var e = focusedArea.createTextRange();
835                 var tx = focusedArea.value.substr(0, n+insCode.unescapeHTML().
836 length.length);
837                 var pl = tx.split(/¥n/);
838                 e.collapse(true);
839                 e.moveStart('character',n+insCode.unescapeHTML().length-pl.len
840 gth+1);
841                 e.text = e.text+'';
842                 e.collapse(false);
843                 e.select();
844         } else {
845                 focusedArea.scrollTop = top;
846                 focusedArea.setSelectionRange(n+insCode.unescapeHTML().length,
847  n+insCode.unescapeHTML().length);
848         }
849         focusedArea.focus();
850 }
851
852 //画数のセレクトボックスのダミーノードを破壊して再生成
853 function recreateSelectDummy(num){
854         var addArea = $('strokeMenu');
855         var remDmy = $('selectDummy');
856         addArea.removeChild(remDmy);
857         var addNode = document.createElement('select');
858         addNode.id = 'selectDummy';
859
860         if(is.gecko || is.opera){
861                 addNode.addEventListener('change',stroke_function,false);
862         }
863         else{
864                 addNode.attachEvent('onchange',stroke_function)
865         }
866         addOption(addNode,null);
867         if(radicalNumber > 0){
868                 for(var i=0;i<num.length;i++){
```

```
869                             addOption(addNode,num[i]);
870                     }
871             }
872
873             addArea.appendChild(addNode);
874 }
875
876 //選択された部首の画数をセレクトボックスに格納する
877 function addOption(node,num){
878             var addNode = document.createElement('option');
879             if(num != null){
880                     addNode.appendChild(document.createTextNode(num.firstChild.nod
881 eValue));
882                     node.appendChild(addNode);
883             }
884             else{
885                     addNode.appendChild(document.createTextNode("All"));
886                     node.appendChild(addNode);
887             }
888 }
889
890 //部首画数選択がクリックされた場合
891 function jumpRad(){
892             var selectElement = $('jumpRadicalStroke');
893             var indexNumber = selectElement.selectedIndex+1;
894             var jump = toJumpRadNum(indexNumber);
895             var radicalSelect = $('radical');
896             radicalSelect.selectedIndex = jump-1;
897             radical_function();
898 }
899
900 //検索結果の漢字のダミーノードを破壊して再生成
901 function recreateImgDummy(){
902             var addArea = document.getElementById('imgArea');
903             var remDmy = document.getElementById('imgDummy');
904             addArea.removeChild(remDmy);
905
906             var addNode = document.createElement('div');
907             addNode.id = 'imgDummy';
908             addArea.appendChild(addNode);
909             /*var range = document.createRange();
910             range.selectNodeContents(remDmy);
911             range.deleteContents();*/
912             cpListNum = 0;
913 }
914
915 //部首画数選択のセレクトボックスの中身を生成
916 function gene_JumpRadicalOption(){
917             var selectBox = $('jumpRadicalStroke');
918             var opt = new Array(17);
919             for(var i=0;i<17;i++){
920                     opt[i] = i+1;
921                     var addNode = document.createElement('option');
922                     addNode.appendChild(document.createTextNode(opt[i]));
923                     selectBox.appendChild(addNode);
924             }
925 }
926 //セレクトボックスの中身を生成
927 function start_func(){
928             gene_strokeOption();
929             gene_JumpRadicalOption();
930             setSearchFunc();
```

```
931         focusedArea = $('search_form:search_word1');
932 }
933
934 var saveArea = $('writeArea');
935
936 //子ノードを隠す
937 function hide(){
938         var url = "http://st130.itc.u-toyama.ac.jp/dokb/ime/";
939         var root = $('writeDummy');
940         var checkHide = $('hide');
941         if(checkHide.src == url+"stor.png"){
942                 checkHide.src = url+"evol.png";
943                 write_cookie();
944                 saveArea = $('writeArea');
945                 root.removeChild(saveArea);
946         }
947         else{
948                 checkHide.src = url+"stor.png";
949                 root.appendChild(saveArea);
950                 write_cookie();
951         }
952 }
953
954 //DOM操作により、予め設置してあるdivノードにシステムを生成する為の処理
955
956 function generate_dummy(){
957         var root = $('writeDummy');
958         var onCheck = document.createElement('img');
959         onCheck.id = "hide";
960         onCheck.className = "onCursor";
961         onCheck.width = 10;
962         onCheck.height = 10;
963         var url = "http://st130.itc.u-toyama.ac.jp/dokb/ime/";
964         onCheck.src = url+"stor.png";
965         root.appendChild(onCheck);
966         root.appendChild(document.createElement('br'));
967         var attention = document.createElement('span');
968         attention.appendChild(document.createTextNode("漢字検索システム(β) -U
969 nicodeにおけるCJK統合漢字拡張A及びBに収録されている漢字を部首ベースによる検索
970 ができます。"));
971         attention.id = "attention";
972         var suggest = document.createElement('span');
973         var font = document.createElement('a');
974         font.href = "http://wiki.livedoor.jp/qvarie/d/%b3%c8%c4%a5%b4%c1%bb%fa
975 %a5%d5%a5%a9%a5%f3%a5%c8%b4%d8%cf%a2";
976         font.appendChild(document.createTextNode("フォント"));
977         suggest.appendChild(document.createTextNode("IE等では表示に的確な"));
978         suggest.appendChild(font);
979         suggest.appendChild(document.createTextNode("を指定する必要があるため
980 、Mozilla Firefoxによる使用を推奨します。"));
981         suggest.id = "suggest";
982
983         var writeArea = document.createElement('div');
984         writeArea.id = "writeArea";
985         root.appendChild(writeArea);
986         writeArea.appendChild(attention);
987         writeArea.appendChild(document.createElement('br'));
988         writeArea.appendChild(suggest);
989         var selectTable = document.createElement('table');
990         var selectTbody = document.createElement('tbody');
991
992         var selectTr = document.createElement('tr');
```

```
 993         var radLabelTd = document.createElement('td');
 994         var strokeLabelTd = document.createElement('td');
 995         var jumpLabelTd = document.createElement('td');
 996
 997         var boxTr = document.createElement('tr');
 998         var radicalTd = document.createElement('td');
 999         var strokeTd = document.createElement('td');
1000         var jumpTd = document.createElement('td');
1001
1002         selectTable.border = 0;
1003         radicalTd.rowSpan = 3;
1004
1005         selectTr.appendChild(jumpLabelTd);
1006         selectTr.appendChild(radLabelTd);
1007         selectTr.appendChild(strokeLabelTd);
1008         selectTbody.appendChild(selectTr);
1009
1010         boxTr.appendChild(jumpTd);
1011         boxTr.appendChild(radicalTd);
1012         boxTr.appendChild(strokeTd);
1013         selectTbody.appendChild(boxTr);
1014         selectTable.appendChild(selectTbody);
1015
1016         var radicalForm = document.createElement('form');
1017         var selectRad = document.createElement('select');
1018
1019         selectRad.size = 6;
1020         selectRad.id = "radical";
1021         selectRad.multiple = "multiple";
1022
1023         var selectRadStroke = document.createElement('select');
1024         selectRadStroke.size = 3;
1025         selectRadStroke.id = "jumpRadicalStroke";
1026         selectRadStroke.multiple = "multiple";
1027
1028         jumpLabelTd.appendChild(document.createTextNode("部首画数選択"));
1029         jumpLabelTd.appendChild(document.createElement('br'));
1030         jumpLabelTd.appendChild(document.createTextNode("(select strokes in ra
1031 dical)"));
1032         jumpTd.appendChild(selectRadStroke);
1033
1034         var childLabelTr = document.createElement('tr');
1035         var childBoxTr = document.createElement('tr');
1036         var childLabelTd = document.createElement('td');
1037         var childBoxTd = document.createElement('td');
1038         var td2span = document.createElement('span');
1039
1040         selectTbody.appendChild(childLabelTr);
1041         selectTbody.appendChild(childBoxTr);
1042
1043         td2span.id = "searchFromChar";
1044
1045         var inputChar = document.createElement('input');
1046         inputChar.size = 4;
1047         inputChar.type = "text";
1048         inputChar.id = "searchRadFromChar";
1049
1050         var inputButton = document.createElement('input');
1051         inputButton.type = "button";
1052         inputButton.id = "searchRadFromButton";
1053         inputButton.value = "search";
1054
```

```
1055          childLabelTr.appendChild(childLabelTd);
1056          td2span.appendChild(document.createTextNode("文字から部首を検索"));
1057          td2span.appendChild(document.createElement('br'));
1058          td2span.appendChild(document.createTextNode("(search Radical from char
1059 acter)"));
1060          childLabelTd.appendChild(td2span);
1061
1062          var dispType = document.createElement('input');
1063          var dispForm = document.createElement('form');
1064          var dispBoxTd = document.createElement('td');
1065          dispType.type = "checkbox";
1066          dispType.id = "dispWay";
1067          dispForm.appendChild(dispType);
1068          dispForm.align = "center";
1069          dispBoxTd.appendChild(dispForm);
1070
1071          var dispLabel = document.createTextNode("コードポイント表記");
1072          var checkLabelTd = document.createElement('td');
1073          checkLabelTd.appendChild(dispLabel);
1074          childLabelTr.appendChild(checkLabelTd);
1075
1076          childBoxTd.appendChild(inputChar);
1077          childBoxTd.appendChild(inputButton);
1078          childBoxTr.appendChild(childBoxTd);
1079          childBoxTr.appendChild(dispBoxTd);
1080
1081          radLabelTd.appendChild(document.createTextNode("部首選択"));
1082          radLabelTd.appendChild(document.createElement('br'));
1083          radLabelTd.appendChild(document.createTextNode("(select radical)"));
1084          radicalTd.appendChild(radicalForm);
1085          radicalForm.appendChild(selectRad);
1086
1087          var menuDiv = document.createElement('div');
1088          var selectStroke = document.createElement('select');
1089
1090          menuDiv.id = "strokeMenu";
1091          menuDiv.align = "center";
1092
1093          selectStroke.id = "selectDummy";
1094
1095          strokeLabelTd.appendChild(document.createTextNode("画数選択"));
1096          strokeLabelTd.appendChild(document.createElement('br'));
1097          strokeLabelTd.appendChild(document.createTextNode("(select strokes)"))
1098 ;
1099          menuDiv.appendChild(selectStroke);
1100          strokeTd.appendChild(menuDiv);
1101
1102          var imgArea = document.createElement('div');
1103          var imgDummy = document.createElement('div');
1104          imgArea.id = "imgArea";
1105          imgDummy.id = "imgDummy";
1106
1107
1108          writeArea.appendChild(selectTable);
1109          imgArea.appendChild(imgDummy);
1110          writeArea.appendChild(imgArea);
1111
1112          //セレクトボックスの中身を生成
1113          start_func();
1114
1115          //各イベントリスナーを設定
1116          if(is.gecko || is.opera){
```

```
1117                    onCheck.addEventListener('click',hide,false);
1118                    dispType.addEventListener('change',checkBox,false);
1119                    selectStroke.addEventListener('change',stroke_function,false);
1120
1121                    selectRad.addEventListener('change',radical_function,false);
1122                    selectRadStroke.addEventListener('click',jumpRad,false);
1123                    inputChar.addEventListener('keypress',char2radnum,false);
1124                    inputButton.addEventListener('click',char2radnumB,false);
1125            }
1126            else{
1127                    onCheck.attachEvent('onclick',hide);
1128                    dispType.attachEvent('onchange',checkBox);
1129                    selectStroke.attachEvent('onchange',stroke_function);
1130                    selectRad.attachEvent('onchange',radical_function);
1131                    selectRadStroke.attachEvent('onclick',jumpRad);
1132                    inputChar.attachEvent('onkeydown',char2radnum);
1133                    inputButton.attachEvent('onclick',char2radnumB);
1134            }
1135
1136            //cookieから前の状態を復元
1137            decomp();
1138 }
1139
1140 //cookie関連の処理
1141
1142 //表示するかしないかをcookieに格納する値
1143 var onDisp
1144
1145 //cookieへの書き込み
1146 function write_cookie(){
1147            var hide = $('hide');
1148            var src = hide.src;
1149            if(src==uri+"stor.png"){
1150                    onDisplay=1;
1151            }
1152            else{
1153                    onDisplay=0;
1154            }
1155            var times = new Date();
1156            //cookieの有効期限24時間
1157            times.setTime(times.getTime()+1000*60*60*24);
1158
1159            var selectElement = $('selectDummy');
1160            var indexNumber = selectElement.selectedIndex;
1161            var cplist=cpListNum;
1162            if(cplist==cpListTable.length&&cplist%dispNumber>0){
1163                    cplist = cplist + dispNumber - cplist%dispNumber;
1164            }
1165            var dispCode=0;
1166            if($('dispWay').checked){
1167                    dispCode=1;
1168            }
1169
1170            cplist = cplist-dispNumber;
1171            if(cplist<0){
1172                    cplist=0;
1173            }
1174            document.cookie = "dokb_dsp="+onDisplay+":dokb_rad="+radicalNumber+":d
1175 okb_str="+indexNumber+":dokb_cpL="+cplist+":dokb_cdp="+dispCode+"; path=/dokb/
1176  ; expires="+times.toGMTString();
1177 }
1178
```

```
1179 //cookieから前の状態を復元
1180 function decomp(){
1181         var thisCookie = document.cookie.split("; ")[0];
1182         var cookieList = new Array();
1183         cookieList = thisCookie.split(":");
1184         var disp;
1185         var rad;
1186         var strk;
1187         var cplis;
1188         var cdp;
1189         if(cookieList[0].substr(0,9) == "dokb_dsp="){
1190                 disp=cookieList[0].substr(9,cookieList[0].length);
1191         }
1192         if(cookieList[1].substr(0,9) == "dokb_rad="){
1193                 rad=cookieList[1].substr(9,cookieList[1].length);
1194         }
1195         if(cookieList[2].substr(0,9) == "dokb_str="){
1196                 strk=cookieList[2].substr(9,cookieList[2].length);
1197         }
1198         if(cookieList[3].substr(0,9) == "dokb_cpL="){
1199                 cplis=cookieList[3].substr(9,cookieList[3].length);
1200         }
1201         if(cookieList[4].substr(0,9) == "dokb_cdp="){
1202                 cdp=cookieList[4].substr(9,cookieList[4].length);
1203         }
1204         if(cdp==1){
1205                 $('dispWay').checked=1;
1206         }
1207
1208         if(rad > -1){
1209                 var selectedElement = $('radical');
1210                 radicalNumber = rad;
1211                 selectedElement.selectedIndex=radicalNumber-1;
1212
1213                 var req = new Ajax.Request( cgi+"searchcode.pl?" + radicalNumb
1214 er , { method: 'GET', onComplete: function(request){
1215                         var tempXML = request.responseXML;
1216
1217                         var root = tempXML.documentElement;
1218                         var strokeList = new Array();
1219
1220                         strokeList = root.getElementsByTagName('stroke_num');
1221                         recreateSelectDummy(strokeList);
1222
1223
1224                         if(strk>-1){
1225                                 strk=strk-1;
1226                                 var selectedStroke = $('selectDummy');
1227                                 selectedStroke.selectedIndex=strk+1;
1228                                 cpListNum=cplis;
1229                                 stroke_function();
1230                         }
1231                         if(disp!=1){
1232                                 hide();
1233                         }
1234                 },onFailure: function(request){
1235                         if(disp!=1){
1236                                 hide();
1237                         }
1238                 }});
1239         }
1240 }
```

```
1241
1242 //文字から属する部首を検索する処理
1243
1244 //テキストエリア内のkeypressイベント(onkeydown)の処理 enterが押されたか判定
1245 function char2radnum(e){
1246         if(e.keyCode == 13) {
1247                 char2radnumB();
1248         }
1249 }
1250
1251 //searchボタンを押した時の処理
1252 function char2radnumB(){
1253         var text = $('searchRadFromChar');
1254         //先頭文字の文字コードを得る
1255         var code = char2codepoint(text.value);
1256         //文字コードから属する部首を得る
1257         getRadnumFromCode(code);
1258 }
1259
1260 //テキストエリア内の先頭の文字の16進文字コード(UTF8)を得る
1261 function char2codepoint(str){
1262         var surrogate_1st = 0;
1263         var code = null;
1264         var i = 0;
1265         do {
1266                 var utf16_code = str.charCodeAt(i);
1267                 if (surrogate_1st != 0) {
1268                         if (utf16_code >= 0xDC00 && utf16_code <= 0xDFFF) {
1269                                 var surrogate_2nd = utf16_code;
1270                                 var unicode_code = (surrogate_1st - 0xD800) *
1271 (1 << 10) + (1 << 16) +
1272                                                                         (surro
1273 gate_2nd - 0xDC00);
1274                                 code = unicode_code;
1275                         } else {
1276                         // Malformed surrogate pair ignored.
1277                                 code = 4E00;
1278                         }
1279                                 surrogate_1st = 0;
1280                 } else if (utf16_code >= 0xD800 && utf16_code <= 0xDBFF) {
1281                         surrogate_1st = utf16_code;
1282                 } else {
1283                         code = utf16_code;
1284                 }
1285                 i++;
1286         } while(code == null);
1287         return(code.toString(16).toUpperCase());
1288 }
1289
1290 //データベースへの要求
1291 function getRadnumFromCode(code){
1292         var req = new Ajax.Request( cgi+"searchcode.pl?"+code, { method: 'GET'
1293 , onComplete: moveRadNum});
1294 }
1295
1296 //得た部首ナンバーで部首選択のセレクトボックスを移動
1297 function moveRadNum(request){
1298         var tempXML = request.responseXML;
1299         var root = tempXML.documentElement;
1300         var jump = root.getElementsByTagName('radical_num');
1301         var radnum = jump[0].firstChild.nodeValue;
1302
```

```
1303          var radicalSelect = $('radical');
1304          radicalSelect.selectedIndex = radnum-1;
1305          radical_function();
1306 }
1307
1308 //各種イベント等の為の設定
1309
1310 //部首選択のセレクトボックスに入る項目の設定
1311 function gene_strokeOption(){
1312          var selectBox = $('radical');
1313          var opt = new Array(215);
1314          opt[0] = "";
1315          opt[1] = "一(one)";
1316          opt[2] = "丨(line)";
1317          opt[3] = "丶(dot)";
1318          opt[4] = "丿(slash)";
1319          opt[5] = "乙(second)";
1320          opt[6] = "亅(hook)";
1321          opt[7] = "二(two)";
1322          opt[8] = "亠(lid)";
1323          opt[9] = "人(man)";
1324          opt[10] = "儿(legs)";
1325          opt[11] = "入(enter)";
1326          opt[12] = "八(eight)";
1327          opt[13] = "冂(down box)";
1328          opt[14] = "冖(cover)";
1329          opt[15] = "冫(ice)";
1330          opt[16] = "几(table)";
1331          opt[17] = "凵(open box)";
1332          opt[18] = "刀(knife)";
1333          opt[19] = "力(power)";
1334          opt[20] = "勹(wrap)";
1335          opt[21] = "匕(spoon)";
1336          opt[22] = "匚(right open)";
1337          opt[23] = "匸(hiding)";
1338          opt[24] = "十(ten)";
1339          opt[25] = "卜(divination)";
1340          opt[26] = "卩(seal)";
1341          opt[27] = "厂(cliff)";
1342          opt[28] = "厶(private)";
1343          opt[29] = "又(again)";
1344          opt[30] = "口(mouth)";
1345          opt[31] = "囗(enclosure)";
1346          opt[32] = "土(earth)";
1347          opt[33] = "士(scholar)";
1348          opt[34] = "夂(go)";
1349          opt[35] = "夊(go slowly)";
1350          opt[36] = "夕(evening)";
1351          opt[37] = "大(big)";
1352          opt[38] = "女(woman)";
1353          opt[39] = "子(child)";
1354          opt[40] = "宀(roof)";
1355          opt[41] = "寸(inch)";
1356          opt[42] = "小(small)";
1357          opt[43] = "尢(lame)";
1358          opt[44] = "尸(corpse)";
1359          opt[45] = "屮(sprout)";
1360          opt[46] = "山(mountain)";
1361          opt[47] = "巛(river)";
1362          opt[48] = "工(work)";
1363          opt[49] = "己(oneself)";
1364          opt[50] = "巾(turban)";
```

```
1365        opt[51] = "干(dry)";
1366        opt[52] = "幺(short thread)";
1367        opt[53] = "广(dotted cliff)";
1368        opt[54] = "廴(long stride)";
1369        opt[55] = "廾(two hands)";
1370        opt[56] = "弋(shoot)";
1371        opt[57] = "弓(bow)";
1372        opt[58] = "彐(snout)";
1373        opt[59] = "彡(bristle)";
1374        opt[60] = "彳(step)";
1375        opt[61] = "心(heart)";
1376        opt[62] = "戈(halberd)";
1377        opt[63] = "戶(door)";
1378        opt[64] = "手(hand)";
1379        opt[65] = "支(branch)";
1380        opt[66] = "攴(rap)";
1381        opt[67] = "文(script)";
1382        opt[68] = "斗(dipper)";
1383        opt[69] = "斤(axe)";
1384        opt[70] = "方(square)";
1385        opt[71] = "无(not)";
1386        opt[72] = "日(sun)";
1387        opt[73] = "曰(say)";
1388        opt[74] = "月(moon)";
1389        opt[75] = "木(tree)";
1390        opt[76] = "欠(lack)";
1391        opt[77] = "止(stop)";
1392        opt[78] = "歹(death)";
1393        opt[79] = "殳(weapon)";
1394        opt[80] = "毋(do not)";
1395        opt[81] = "比(compare)";
1396        opt[82] = "毛(fur)";
1397        opt[83] = "氏(clan)";
1398        opt[84] = "气(stream)";
1399        opt[85] = "水(water)";
1400        opt[86] = "火(fire)";
1401        opt[87] = "爪(claw)";
1402        opt[88] = "父(father)";
1403        opt[89] = "爻(double x)";
1404        opt[90] = "爿(half tree trunk)";
1405        opt[91] = "片(slice)";
1406        opt[92] = "牙(fang)";
1407        opt[93] = "牛(cow)";
1408        opt[94] = "犬(dog)";
1409        opt[95] = "玄(profound)";
1410        opt[96] = "玉(jade)";
1411        opt[97] = "瓜(melon)";
1412        opt[98] = "瓦(tile)";
1413        opt[99] = "甘(sweet)";
1414        opt[100] = "生(life)";
1415        opt[101] = "用(use)";
1416        opt[102] = "田(field)";
1417        opt[103] = "疋(bolt of cloth)";
1418        opt[104] = "疒(sickness)";
1419        opt[105] = "癶(dotted tent)";
1420        opt[106] = "白(white)";
1421        opt[107] = "皮(skin)";
1422        opt[108] = "皿(dish)";
1423        opt[109] = "目(eye)";
1424        opt[110] = "矛(spear)";
1425        opt[111] = "矢(arrow)";
1426        opt[112] = "石(stone)";
```

```
1427        opt[113] = "示(spirit)";
1428        opt[114] = "内(track)";
1429        opt[115] = "禾(grain)";
1430        opt[116] = "穴(cave)";
1431        opt[117] = "立(stand)";
1432        opt[118] = "竹(bamboo)";
1433        opt[119] = "米(rice)";
1434        opt[120] = "糸(silk)";
1435        opt[121] = "缶(jar)";
1436        opt[122] = "网(net)";
1437        opt[123] = "羊(sheep)";
1438        opt[124] = "羽(feaher)";
1439        opt[125] = "老(old)";
1440        opt[126] = "而(and)";
1441        opt[127] = "耒(plow)";
1442        opt[128] = "耳(ear)";
1443        opt[129] = "聿(brush)";
1444        opt[130] = "肉(meat)";
1445        opt[131] = "臣(minister)";
1446        opt[132] = "自(self)";
1447        opt[133] = "至(arrive)";
1448        opt[134] = "臼(mortar)";
1449        opt[135] = "舌(tongue)";
1450        opt[136] = "舛(oppose)";
1451        opt[137] = "舟(boat)";
1452        opt[138] = "艮(stopping)";
1453        opt[139] = "色(color)";
1454        opt[140] = "艸(grass)";
1455        opt[141] = "虍(tiger)";
1456        opt[142] = "虫(insect)";
1457        opt[143] = "血(blood)";
1458        opt[144] = "行(walk enclosure)";
1459        opt[145] = "衣(clothes)";
1460        opt[146] = "襾(west)";
1461        opt[147] = "見(see)";
1462        opt[148] = "角(horn)";
1463        opt[149] = "言(speech)";
1464        opt[150] = "谷(valley)";
1465        opt[151] = "豆(bean)";
1466        opt[152] = "豕(pig)";
1467        opt[153] = "豸(badger)";
1468        opt[154] = "貝(shell)";
1469        opt[155] = "赤(red)";
1470        opt[156] = "走(run)";
1471        opt[157] = "足(foot)";
1472        opt[158] = "身(body)";
1473        opt[159] = "車(cart)";
1474        opt[160] = "辛(bitter)";
1475        opt[161] = "辰(morning)";
1476        opt[162] = "辵(walk)";
1477        opt[163] = "邑(city)";
1478        opt[164] = "酉(wine)";
1479        opt[165] = "釆(distinguish)";
1480        opt[166] = "里(village)";
1481        opt[167] = "金(gold)";
1482        opt[168] = "長(long)";
1483        opt[169] = "門(gate)";
1484        opt[170] = "阜(mound)";
1485        opt[171] = "隶(slave)";
1486        opt[172] = "隹(short tailed bird)";
1487        opt[173] = "雨(rain)";
1488        opt[174] = "青(blue)";
```

```
1489          opt[175] = "非(wrong)";
1490          opt[176] = "面(face)";
1491          opt[177] = "革(leather)";
1492          opt[178] = "韋(tanned leather)";
1493          opt[179] = "韭(leek)";
1494          opt[180] = "音(sound)";
1495          opt[181] = "頁(leaf)";
1496          opt[182] = "風(wind)";
1497          opt[183] = "飛(fly)";
1498          opt[184] = "食(eat)";
1499          opt[185] = "首(head)";
1500          opt[186] = "香(fragrant)";
1501          opt[187] = "馬(horse)";
1502          opt[188] = "骨(bone)";
1503          opt[189] = "高(tall)";
1504          opt[190] = "髟(hair)";
1505          opt[191] = "鬥(fight)";
1506          opt[192] = "鬯(sacrificial wine)";
1507          opt[193] = "鬲(cauldron)";
1508          opt[194] = "鬼(ghost)";
1509          opt[195] = "魚(fish)";
1510          opt[196] = "鳥(bird)";
1511          opt[197] = "鹵(salt)";
1512          opt[198] = "鹿(deer)";
1513          opt[199] = "麥(wheat)";
1514          opt[200] = "麻(hemp)";
1515          opt[201] = "黃(yellow)";
1516          opt[202] = "黍(millet)";
1517          opt[203] = "黑(black)";
1518          opt[204] = "黹(embroidery)";
1519          opt[205] = "黽(frog)";
1520          opt[206] = "鼎(tripod)";
1521          opt[207] = "鼓(drum)";
1522          opt[208] = "鼠(rat)";
1523          opt[209] = "鼻(bise)";
1524          opt[210] = "齊(even)";
1525          opt[211] = "齒(tooth)";
1526          opt[212] = "龍(dragon)";
1527          opt[213] = "龜(turtle)";
1528          opt[214] = "龠(flute)";
1529
1530          for(var i=1;i<opt.length;i++){
1531                  var addNode = document.createElement('option');
1532                  addNode.appendChild(document.createTextNode(opt[i]));
1533                  selectBox.appendChild(addNode);
1534          }
1535  }
1536
1537  //テキストエリアにイベントを設置する準備
1538  function setInsEvent(no,node){
1539          switch(no){
1540                  case 0:
1541                          if(is.gecko || is.opera){
1542                                  node.addEventListener('click',setStr0,false);
1543                          }
1544                          else{
1545                                  node.attachEvent('onclick',setStr0);
1546                          }
1547                          break;
1548                  case 1:
1549                          if(is.gecko || is.opera){
1550                                  node.addEventListener('click',setStr1,false);
```

```
1551                    }
1552                    else{
1553                            node.attachEvent('onclick',setStr1);
1554                    }
1555                    break;
1556            case 2:
1557                    if(is.gecko || is.opera){
1558                            node.addEventListener('click',setStr2,false);
1559                    }
1560                    else{
1561                            node.attachEvent('onclick',setStr2);
1562                    }
1563                    break;
1564            case 3:
1565                    if(is.gecko || is.opera){
1566                            node.addEventListener('click',setStr3,false);
1567                    }
1568                    else{
1569                            node.attachEvent('onclick',setStr3);
1570                    }
1571                    break;
1572            case 4:
1573                    if(is.gecko || is.opera){
1574                            node.addEventListener('click',setStr4,false);
1575                    }
1576                    else{
1577                            node.attachEvent('onclick',setStr4);
1578                    }
1579                    break;
1580            case 5:
1581                    if(is.gecko || is.opera){
1582                            node.addEventListener('click',setStr5,false);
1583                    }
1584                    else{
1585                            node.attachEvent('onclick',setStr5);
1586                    }
1587                    break;
1588            case 6:
1589                    if(is.gecko || is.opera){
1590                            node.addEventListener('click',setStr6,false);
1591                    }
1592                    else{
1593                            node.attachEvent('onclick',setStr6);
1594                    }
1595                    break;
1596            case 7:
1597                    if(is.gecko || is.opera){
1598                            node.addEventListener('click',setStr7,false);
1599                    }
1600                    else{
1601                            node.attachEvent('onclick',setStr7);
1602                    }
1603                    break;
1604            case 8:
1605                    if(is.gecko || is.opera){
1606                            node.addEventListener('click',setStr8,false);
1607                    }
1608                    else{
1609                            node.attachEvent('onclick',setStr8);
1610                    }
1611                    break;
1612            case 9:
```

```
1613                            if(is.gecko || is.opera){
1614                                    node.addEventListener('click',setStr9,false);
1615                            }
1616                            else{
1617                                    node.attachEvent('onclick',setStr9);
1618                            }
1619                            break;
1620                    default:
1621                            break;
1622            }
1623 }
1624
1625 //テキストエリアにイベントを設置
1626 function setSearchFunc(){
1627            for(var i=1;i<=4;i++){
1628                    var searchArea = $('search_form:search_word'+i);
1629
1630                    switch(i){
1631                    case 1:
1632                            if(is.gecko || is.opera){
1633                                    searchArea.addEventListener('click',setTxtBox1
1634 ,false);
1635                            }
1636                            else{
1637                                    searchArea.attachEvent('onclick',setTxtBox1);
1638                            }
1639                            break;
1640                    case 2:
1641                            if(is.gecko || is.opera){
1642                                    searchArea.addEventListener('click',setTxtBox2
1643 ,false);
1644                            }
1645                            else{
1646                                    searchArea.attachEvent('onclick',setTxtBox2);
1647                            }
1648                            break;
1649                    case 3:
1650                            if(is.gecko || is.opera){
1651                                    searchArea.addEventListener('click',setTxtBox3
1652 ,false);
1653                            }
1654                            else{
1655                                    searchArea.attachEvent('onclick',setTxtBox3);
1656                            }
1657                            break;
1658                    case 4:
1659                            if(is.gecko || is.opera){
1660                                    searchArea.addEventListener('click',setTxtBox4
1661 ,false);
1662                            }
1663                            else{
1664                                    searchArea.attachEvent('onclick',setTxtBox4);
1665                            }
1666                            break;
1667                    default:
1668                            break;
1669                    }
1670            }
1671 }
1672
1673 //各テキストエリアに対応した漢字をcpTableより挿入
1674 function setStr0(){
```

```
1675            setStr2TxtBox(cpTable[0]);
1676    }
1677    function setStr1(){
1678            setStr2TxtBox(cpTable[1]);
1679    }
1680    function setStr2(){
1681            setStr2TxtBox(cpTable[2]);
1682    }
1683    function setStr3(){
1684            setStr2TxtBox(cpTable[3]);
1685    }
1686    function setStr4(){
1687            setStr2TxtBox(cpTable[4]);
1688    }
1689    function setStr5(){
1690            setStr2TxtBox(cpTable[5]);
1691    }
1692    function setStr6(){
1693            setStr2TxtBox(cpTable[6]);
1694    }
1695    function setStr7(){
1696            setStr2TxtBox(cpTable[7]);
1697    }
1698    function setStr8(){
1699            setStr2TxtBox(cpTable[8]);
1700    }
1701    function setStr9(){
1702            setStr2TxtBox(cpTable[9]);
1703    }
1704
1705    //書き込むテキストエリアのid属性を指定
1706    function setTxtBox1(){
1707            focusedArea = $('search_form:search_word1');
1708    }
1709    function setTxtBox2(){
1710            focusedArea = $('search_form:search_word2');
1711    }
1712    function setTxtBox3(){
1713            focusedArea = $('search_form:search_word3');
1714    }
1715    function setTxtBox4(){
1716            focusedArea = $('search_form:search_word4');
1717    }
1718
1719    //部首画数選択で跳ぶためのインデックスを返す
1720    function toJumpRadNum(index){
1721            switch(index){
1722                    case 1:
1723                            return 1;
1724                            break;
1725                    case 2:
1726                            return 7;
1727                            break;
1728                    case 3:
1729                            return 30;
1730                            break;
1731                    case 4:
1732                            return 61;
1733                            break;
1734                    case 5:
1735                            return 95;
1736                            break;
```

```
1737                    case 6:
1738                            return 118;
1739                            break;
1740                    case 7:
1741                            return 147;
1742                            break;
1743                    case 8:
1744                            return 167;
1745                            break;
1746                    case 9:
1747                            return 176;
1748                            break;
1749                    case 10:
1750                            return 187;
1751                            break;
1752                    case 11:
1753                            return 195;
1754                            break;
1755                    case 12:
1756                            return 201;
1757                            break;
1758                    case 13:
1759                            return 205;
1760                            break;
1761                    case 14:
1762                            return 209;
1763                            break;
1764                    case 15:
1765                            return 211;
1766                            break;
1767                    case 16:
1768                            return 212;
1769                            break;
1770                    case 17:
1771                            return 214;
1772                            break;
1773                    default:
1774                            break;
1775            }
1776            return 1;
1777 }
```

2）unisearch_cookie.js

unisearch_cookie.js　　Page 1

```
 1 //cookie関連の処理
 2
 3 //表示するかしないかをcookieに格納する値
 4 var onDisp
 5
 6 //cookieへの書き込み
 7 function write_cookie(){
 8         var hide = $('hide');
 9         var src = hide.src;
10         if(src==uri+"stor.png"){
11                 onDisplay=1;
12         }
13         else{
14                 onDisplay=0;
15         }
16         var times = new Date();
17         //cookieの有効期限24時間
18         times.setTime(times.getTime()+1000*60*60*24);
19
20         var selectElement = $('selectDummy');
21         var indexNumber = selectElement.selectedIndex;
22         var cplist=cpListNum;
23         if(cplist==cpListTable.length&&cplist%dispNumber>0){
24                 cplist = cplist + dispNumber - cplist%dispNumber;
25         }
26         var dispCode=0;
27         if($('dispWay').checked){
28                 dispCode=1;
29         }
30
31         cplist = cplist-dispNumber;
32         if(cplist<0){
33                 cplist=0;
34         }
35         document.cookie = "dokb_dsp="+onDisplay+":dokb_rad="+radicalNumber+":d
36 okb_str="+indexNumber+":dokb_cpL="+cplist+":dokb_cdp="+dispCode+"; path=/dokb/
37  ; expires="+times.toGMTString();
38 }
39
40 //cookieから前の状態を復元
41 function decomp(){
42         var thisCookie = document.cookie.split("; ")[0];
43         var cookieList = new Array();
44         cookieList = thisCookie.split(":");
45         var disp;
46         var rad;
47         var strk;
48         var cplis;
49         var cdp;
50         if(cookieList[0].substr(0,9) == "dokb_dsp="){
51                 disp=cookieList[0].substr(9,cookieList[0].length);
52         }
53         if(cookieList[1].substr(0,9) == "dokb_rad="){
54                 rad=cookieList[1].substr(9,cookieList[1].length);
55         }
56         if(cookieList[2].substr(0,9) == "dokb_str="){
57                 strk=cookieList[2].substr(9,cookieList[2].length);
58         }
59         if(cookieList[3].substr(0,9) == "dokb_cpL="){
60                 cplis=cookieList[3].substr(9,cookieList[3].length);
61         }
62         if(cookieList[4].substr(0,9) == "dokb_cdp="){
```

```
63                  cdp=cookieList[4].substr(9,cookieList[4].length);
64          }
65          if(cdp==1){
66                  $('dispWay').checked=1;
67          }
68
69          if(rad > -1){
70                  var selectedElement = $('radical');
71                  radicalNumber = rad;
72                  selectedElement.selectedIndex=radicalNumber-1;
73
74                  var req = new Ajax.Request( cgi+"searchcode.pl?" + radicalNumb
75 er , { method: 'GET', onComplete: function(request){
76                          var tempXML = request.responseXML;
77
78                          var root = tempXML.documentElement;
79                          var strokeList = new Array();
80
81                          strokeList = root.getElementsByTagName('stroke_num');
82                          recreateSelectDummy(strokeList);
83
84
85                          if(strk>-1){
86                                  strk=strk-1;
87                                  var selectedStroke = $('selectDummy');
88                                  selectedStroke.selectedIndex=strk+1;
89                                  cpListNum=cplis;
90                                  stroke_function();
91                          }
92                          if(disp!=1){
93                                  hide();
94                          }
95                  },onFailure: function(request){
96                          if(disp!=1){
97                                  hide();
98                          }
99                  }});
100         }
101 }
```

（3）unisearch_dom.js

unisearch_dom.js　　Page 1

```
 1 //システムのユーザー操作に対する非同期通信及びDOM操作に関する処理
 2
 3 //部首番号
 4 var radicalNumber = -1;
 5 var nodeList = new Array();
 6 //コードポイントで表示するか
 7 var dispCode = false;
 8 //コードポイントの位置
 9 var cpListNum = 0;
10 //一度に表示する漢字数
11 var dispNumber = 10;
12 //表示している漢字を格納
13 var cpTable = new Array(dispNumber);
14 var cpListTable = new Array();
15 //最後にフォーカスされたtextAreaを格納
16 var focusedArea;
17
18 //フォルダの位置
19 var cgi = "http://stl30.itc.u-toyama.ac.jp/cgi-bin/";
20 var uri = "http://stl30.itc.u-toyama.ac.jp/dokb/ime/";
21
22 //ファイル要求
23 //部首ナンバーからその部首に属している文字のストロークリストを取得
24 function getStroke(){
25         var req = new Ajax.Request( cgi+"searchcode.pl?"+radicalNumber, { meth
26 od: 'GET', onComplete: gene_stroke});//, onException: function(request,excepti
27 on) { document.write(exception)} });
28 }
29
30 //部首ナンバーとストロークナンバーから属している文字のコードポイントリストを取
31 得
32 function getCodepoint(strokeNumber){
33         var req = new Ajax.Request( cgi+"searchcode.pl?"+radicalNumber+"&"+str
34 okeNumber, { method: 'GET', onComplete: gene_code});//, onException: function(
35 request,exception) { document.write(exception)} });
36 }
37
38 //部首が選択された場合
39 function radical_function(){
40         var selectedElement = $('radical');
41
42         radicalNumber = selectedElement.selectedIndex+1;
43         if(radicalNumber > 0){
44                 getStroke();
45         }else {
46                 recreateSelectDummy('null');
47         }
48 }
49
50 //ストロークリストのXMLを取得し、画数のセレクトボックスを生成
51 function gene_stroke(request){
52         var tempXML = request.responseXML;
53
54         var root = tempXML.documentElement;
55         var strokeList = new Array();
56
57         strokeList = root.getElementsByTagName('stroke_num');
58         recreateSelectDummy(strokeList);
59 }
60
61 //コードポイントリストのXMLを取得し、コード表を生成
62 function gene_code(request){
```

```
63          var tempXML = request.responseXML;
64          var root = tempXML.documentElement;
65
66          cpListTable = root.getElementsByTagName('codepoint');
67          setCodeTable();
68  }
69
70  //コードのtableを生成する
71  function setCodeTable(){
72                  var addArea = $('imgDummy');
73                  var tableNode = document.createElement('table');
74                  var tbodyNode = document.createElement('tbody');
75                  var trNode = document.createElement('tr');
76                  tableNode.id = "table";
77                  tableNode.border = "1";
78                  tableNode.cellspacing = "0";
79                  tbodyNode.appendChild(trNode);
80                  tableNode.appendChild(tbodyNode);
81                  addArea.appendChild(tableNode);
82
83                  //cpListTable = codeList;
84                  var i;
85                  for(i=cpListNum;i<dispNumber && i<cpListTable.length;i++){
86                          codepoint = cpListTable[i].firstChild.nodeValue;
87                          cpTable[i] = codepoint;
88                          var tdNode = gene_imgNode(codepoint,trNode);
89                          setInsEvent(i,tdNode);
90                  }
91                  cpListNum = i;
92
93
94                  if(cpListNum<cpListTable.length){
95                          var tw = $('nextB');
96
97                          if(is.gecko || is.opera){
98                                  tw.addEventListener('click',next_code,false);
99                          }
100                         else{
101                                 tw.attachEvent('onclick',next_code);
102                         }
103                         tw.src = uri+"arrowR.png"
104                         tw.className = "onCursor";
105                 };
106 }
107
108 //画数が選択された場合
109 function stroke_function(){
110
111         //コードポイント抜き出し
112         var codeList;
113         var selectElement = $('selectDummy');
114         var indexNumber = selectElement.selectedIndex;
115         recreateImgDummy();
116
117         var addArea = $('imgDummy');
118
119         var loc = document.createElement('left');
120         addArea.appendChild(loc);
121
122         //back矢印画像を設置
123         var creBw = document.createElement('img');
124         creBw.src = uri+"narrowL.png";
```

```
125         creBw.width = 25;
126         creBw.height = 25;
127         creBw.id = "backB";
128         creBw.hspace = 10;
129         loc.appendChild(creBw);
130
131         //next矢印画像を設置
132         var creTw = document.createElement('img');
133         creTw.src = uri+"narrowR.png"
134         creTw.width = 25;
135         creTw.height = 25;
136         creTw.id = "nextB";
137         creTw.hspace = 10;
138         loc.appendChild(creTw);
139
140         var strokeNumber = -1;
141
142         //選択された画数がALL(0)かそれ以外(>0)か
143         if(indexNumber >0){
144                 strokeNumber = selectElement[indexNumber].firstChild.nodeValue
145 ;
146         }
147         getCodepoint(strokeNumber);
148 }
149
150 //右矢印がクリックされた場合
151 function next_code(){
152         var selectElement = $('selectDummy');
153         var indexNumber = selectElement.selectedIndex-1;
154
155         var tableNode = $('table');
156
157         var children = tableNode.childNodes;
158
159         for(var j=0;j<children.length;j++){
160                 tableNode.removeChild(children[j]);
161         }
162
163         var tbodyNode = document.createElement('tbody');
164         var trNode = document.createElement('tr');
165         tableNode.appendChild(tbodyNode);
166         tbodyNode.appendChild(trNode);
167
168         var i;
169         for(i=cpListNum;i<cpListNum+dispNumber && i<cpListTable.length;i++){
170                 codepoint = cpListTable[i].firstChild.nodeValue;
171                 cpTable[i-cpListNum] = codepoint;
172                 var tdNode = gene_imgNode(codepoint,trNode);
173                 setInsEvent(i-cpListNum,tdNode);
174         }
175
176         cpListNum = i;
177         if(cpListNum>=dispNumber){
178                 var bw = $('backB');
179                 if(is.gecko || is.opera){
180                         bw.addEventListener('click',back_code,false);
181                 }
182                 else{
183                         bw.detachEvent('onclick',back_code);
184                         bw.attachEvent('onclick',back_code);
185                 }
186                 bw.src = uri+"arrowL.png"
```

```
187                     bw.className = "onCursor";
188             }
189             if(cpListNum>=cpListTable.length){
190                     var tw = $('nextB');
191                     if(is.gecko || is.opera){
192                             tw.removeEventListener('click',next_code,false);
193                     }
194                     else{
195                             tw.detachEvent('onclick',next_code);
196                     }
197                     tw.src = uri+"narrowR.png"
198                     tw.className = "offCursor";
199             }
200             write_cookie();
201 }
202
203 //左矢印がクリックされた場合
204 function back_code(){
205             var selectElement = $('selectDummy');
206             var indexNumber = selectElement.selectedIndex-1;
207
208             var tableNode = $('table');
209
210             var children = tableNode.childNodes;
211
212             for(var j=0;j<children.length;j++){
213                     tableNode.removeChild(children[j]);
214             }
215
216             var tbodyNode = document.createElement('tbody');
217             var trNode = document.createElement('tr');
218             tableNode.appendChild(tbodyNode);
219             tbodyNode.appendChild(trNode);
220
221             var i;
222
223             //微調整
224             if(cpListNum==cpListTable.length&&cpListNum%dispNumber>0){
225                     cpListNum = cpListNum + dispNumber - cpListNum%dispNumber;
226             }
227
228             var tempBack = cpListNum-dispNumber*2;
229             if(tempBack<0){
230                     tempBack=0;
231             }
232
233             for(i=tempBack;i<tempBack+dispNumber && i<cpListTable.length;i++){
234                     codepoint = cpListTable[i].firstChild.nodeValue;
235                     cpTable[i-tempBack] = codepoint;
236                     var tdNode = gene_imgNode(codepoint,trNode);
237                     setInsEvent(i-tempBack,tdNode);
238             }
239
240             cpListNum = i;
241             if(cpListNum<=cpListTable.length){
242                     var tw = $('nextB');
243                     if(is.gecko || is.opera){
244                             tw.addEventListener('click', next_code,false);
245                     }
246                     else{
247                             tw.detachEvent('onclick', next_code);
248                             tw.attachEvent('onclick', next_code);
```

```
249                     }
250                     tw.src = uri+"arrowR.png"
251                     tw.className = "onCursor";
252             }
253
254             if(cpListNum<=dispNumber){
255                     var bw = $('backB');
256                     if(is.gecko || is.opera){
257                             bw.removeEventListener('click',back_code,false);
258                     }
259                     else{
260                             bw.detachEvent('onclick',back_code);
261                     }
262                     bw.src = uri+"narrowL.png"
263                     bw.className = "offCursor";
264             }
265
266             write_cookie();
267 }
268
269 //コードポイント表記のチェックボックスがクリックされた場合
270 function checkBox(){
271             write_cookie();
272 }
273
274 //個々のコードポイントの挿入するHTMLを生成
275 function gene_imgNode(codepoint,trNode){
276             var url;
277             url = uri+ "img/"+codepoint+".gif";
278             var img ="<img src=¥""+url+"¥">";
279             var tdNode = document.createElement('td');
280             var imgNode = document.createElement('img');
281             imgNode.src = url;
282             trNode.appendChild(tdNode);
283             tdNode.appendChild(imgNode);
284
285             var text = "¥&#x"+codepoint+";";
286             disp = $('dispWay');
287             if(disp.checked){
288                     var textNode = document.createTextNode(codepoint);
289             }
290             else{
291                     var textNode = document.createTextNode(text.unescapeHTML());
292             }
293             tdNode.appendChild(textNode);
294             tdNode.className = "onCursor";
295             return(tdNode);
296 }
297
298 //テキストエリアに文字を挿入
299 function setStr2TxtBox(codepoint){
300             var insCode = '&#x'+codepoint+';';
301             var top = focusedArea.scrollTop;
302             var selection = new Selection(focusedArea);
303             var s = selection.create();
304             var n = s.start;
305             var before = focusedArea.value.substring(0, n);
306             var after = focusedArea.value.substring(n, focusedArea.value.length);
307             focusedArea.value = before + insCode.unescapeHTML() + after;
308
309             if (is.ie) { // IEの場合
310                     var e = focusedArea.createTextRange();
```

```
311                     var tx = focusedArea.value.substr(0, n+insCode.unescapeHTML().
312 length.length);
313                     var pl = tx.split(/¥n/);
314                     e.collapse(true);
315                     e.moveStart('character',n+insCode.unescapeHTML().length-pl.len
316 gth+1);
317                     e.text = e.text+'';
318                     e.collapse(false);
319                     e.select();
320             } else {
321                     focusedArea.scrollTop = top;
322                     focusedArea.setSelectionRange(n+insCode.unescapeHTML().length,
323  n+insCode.unescapeHTML().length);
324             }
325             focusedArea.focus();
326 }
327
328 //画数のセレクトボックスのダミーノードを破壊して再生成
329 function recreateSelectDummy(num){
330             var addArea = $('strokeMenu');
331             var remDmy = $('selectDummy');
332             addArea.removeChild(remDmy);
333             var addNode = document.createElement('select');
334             addNode.id = 'selectDummy';
335
336             if(is.gecko || is.opera){
337                     addNode.addEventListener('change',stroke_function,false);
338             }
339             else{
340                     addNode.attachEvent('onchange',stroke_function)
341             }
342             addOption(addNode,null);
343             if(radicalNumber > 0){
344                     for(var i=0;i<num.length;i++){
345                             addOption(addNode,num[i]);
346                     }
347             }
348
349             addArea.appendChild(addNode);
350 }
351
352 //選択された部首の画数をセレクトボックスに格納する
353 function addOption(node,num){
354             var addNode = document.createElement('option');
355             if(num != null){
356                     addNode.appendChild(document.createTextNode(num.firstChild.nod
357 eValue));
358                     node.appendChild(addNode);
359             }
360             else{
361                     addNode.appendChild(document.createTextNode("All"));
362                     node.appendChild(addNode);
363             }
364 }
365
366 //部首画数選択がクリックされた場合
367 function jumpRad(){
368             var selectElement = $('jumpRadicalStroke');
369             var indexNumber = selectElement.selectedIndex+1;
370             var jump = toJumpRadNum(indexNumber);
371             var radicalSelect = $('radical');
372             radicalSelect.selectedIndex = jump-1;
```

```
373         radical_function();
374 }
375
376 //検索結果の漢字のダミーノードを破壊して再生成
377 function recreateImgDummy(){
378         var addArea = document.getElementById('imgArea');
379         var remDmy = document.getElementById('imgDummy');
380         addArea.removeChild(remDmy);
381
382         var addNode = document.createElement('div');
383         addNode.id = 'imgDummy';
384         addArea.appendChild(addNode);
385         /*var range = document.createRange();
386         range.selectNodeContents(remDmy);
387         range.deleteContents();*/
388         cpListNum = 0;
389 }
390
391 //部首画数選択のセレクトボックスの中身を生成
392 function gene_JumpRadicalOption(){
393         var selectBox = $('jumpRadicalStroke');
394         var opt = new Array(17);
395         for(var i=0;i<17;i++){
396                 opt[i] = i+1;
397                 var addNode = document.createElement('option');
398                 addNode.appendChild(document.createTextNode(opt[i]));
399                 selectBox.appendChild(addNode);
400         }
401 }
402 //セレクトボックスの中身を生成
403 function start_func(){
404         gene_strokeOption();
405         gene_JumpRadicalOption();
406         setSearchFunc();
407         focusedArea = $('search_form:search_word1');
408 }
409
410 var saveArea = $('writeArea');
411
412 //子ノードを隠す
413 function hide(){
414         var url = "http://st130.itc.u-toyama.ac.jp/dokb/ime/";
415         var root = $('writeDummy');
416         var checkHide = $('hide');
417         if(checkHide.src == url+"stor.png"){
418                 checkHide.src = url+"evol.png";
419                 write_cookie();
420                 saveArea = $('writeArea');
421                 root.removeChild(saveArea);
422         }
423         else{
424                 checkHide.src = url+"stor.png";
425                 root.appendChild(saveArea);
426                 write_cookie();
427         }
428 }
```

（4）unisearch_eve.js

unisearch_eve.js　　Page 1

```
 1 //各種イベント等の為の設定
 2
 3 //部首選択のセレクトボックスに入る項目の設定
 4 function gene_strokeOption(){
 5         var selectBox = $('radical');
 6         var opt = new Array(215);
 7         opt[0] = "";
 8         opt[1] = "一(one)";
 9         opt[2] = "丨(line)";
10         opt[3] = "丶(dot)";
11         opt[4] = "丿(slash)";
12         opt[5] = "乙(second)";
13         opt[6] = "亅(hook)";
14         opt[7] = "二(two)";
15         opt[8] = "亠(lid)";
16         opt[9] = "人(man)";
17         opt[10] = "儿(legs)";
18         opt[11] = "入(enter)";
19         opt[12] = "八(eight)";
20         opt[13] = "冂(down box)";
21         opt[14] = "冖(cover)";
22         opt[15] = "冫(ice)";
23         opt[16] = "几(table)";
24         opt[17] = "凵(open box)";
25         opt[18] = "刀(knife)";
26         opt[19] = "力(power)";
27         opt[20] = "勹(wrap)";
28         opt[21] = "匕(spoon)";
29         opt[22] = "匚(right open)";
30         opt[23] = "匸(hiding)";
31         opt[24] = "十(ten)";
32         opt[25] = "卜(divination)";
33         opt[26] = "卩(seal)";
34         opt[27] = "厂(cliff)";
35         opt[28] = "厶(private)";
36         opt[29] = "又(again)";
37         opt[30] = "口(mouth)";
38         opt[31] = "口(enclosure)";
39         opt[32] = "土(earth)";
40         opt[33] = "士(scholar)";
41         opt[34] = "夂(go)";
42         opt[35] = "夊(go slowly)";
43         opt[36] = "夕(evening)";
44         opt[37] = "大(big)";
45         opt[38] = "女(woman)";
46         opt[39] = "子(child)";
47         opt[40] = "宀(roof)";
48         opt[41] = "寸(inch)";
49         opt[42] = "小(small)";
50         opt[43] = "尢(lame)";
51         opt[44] = "尸(corpse)";
52         opt[45] = "屮(sprout)";
53         opt[46] = "山(mountain)";
54         opt[47] = "巛(river)";
55         opt[48] = "工(work)";
56         opt[49] = "己(oneself)";
57         opt[50] = "巾(turban)";
58         opt[51] = "干(dry)";
59         opt[52] = "幺(short thread)";
60         opt[53] = "广(dotted cliff)";
61         opt[54] = "廴(long stride)";
62         opt[55] = "廾(two hands)";
```

```
63          opt[56] = "弋(shoot)";
64          opt[57] = "弓(bow)";
65          opt[58] = "彐(snout)";
66          opt[59] = "彡(bristle)";
67          opt[60] = "彳(step)";
68          opt[61] = "心(heart)";
69          opt[62] = "戈(halberd)";
70          opt[63] = "戶(door)";
71          opt[64] = "手(hand)";
72          opt[65] = "支(branch)";
73          opt[66] = "攴(rap)";
74          opt[67] = "文(script)";
75          opt[68] = "斗(dipper)";
76          opt[69] = "斤(axe)";
77          opt[70] = "方(square)";
78          opt[71] = "无(not)";
79          opt[72] = "日(sun)";
80          opt[73] = "曰(say)";
81          opt[74] = "月(moon)";
82          opt[75] = "木(tree)";
83          opt[76] = "欠(lack)";
84          opt[77] = "止(stop)";
85          opt[78] = "歹(death)";
86          opt[79] = "殳(weapon)";
87          opt[80] = "毋(do not)";
88          opt[81] = "比(compare)";
89          opt[82] = "毛(fur)";
90          opt[83] = "氏(clan)";
91          opt[84] = "气(stream)";
92          opt[85] = "水(water)";
93          opt[86] = "火(fire)";
94          opt[87] = "爪(claw)";
95          opt[88] = "父(father)";
96          opt[89] = "爻(double x)";
97          opt[90] = "爿(half tree trunk)";
98          opt[91] = "片(slice)";
99          opt[92] = "牙(fang)";
100         opt[93] = "牛(cow)";
101         opt[94] = "犬(dog)";
102         opt[95] = "玄(profound)";
103         opt[96] = "玉(jade)";
104         opt[97] = "瓜(melon)";
105         opt[98] = "瓦(tile)";
106         opt[99] = "甘(sweet)";
107         opt[100] = "生(life)";
108         opt[101] = "用(use)";
109         opt[102] = "田(field)";
110         opt[103] = "疋(bolt of cloth)";
111         opt[104] = "疒(sickness)";
112         opt[105] = "癶(dotted tent)";
113         opt[106] = "白(white)";
114         opt[107] = "皮(skin)";
115         opt[108] = "皿(dish)";
116         opt[109] = "目(eye)";
117         opt[110] = "矛(spear)";
118         opt[111] = "矢(arrow)";
119         opt[112] = "石(stone)";
120         opt[113] = "示(spirit)";
121         opt[114] = "内(track)";
122         opt[115] = "禾(grain)";
123         opt[116] = "穴(cave)";
124         opt[117] = "立(stand)";
```

```
125         opt[118] = "竹(bamboo)";
126         opt[119] = "米(rice)";
127         opt[120] = "糸(silk)";
128         opt[121] = "缶(jar)";
129         opt[122] = "网(net)";
130         opt[123] = "羊(sheep)";
131         opt[124] = "羽(feaher)";
132         opt[125] = "老(old)";
133         opt[126] = "而(and)";
134         opt[127] = "耒(plow)";
135         opt[128] = "耳(ear)";
136         opt[129] = "聿(brush)";
137         opt[130] = "肉(meat)";
138         opt[131] = "臣(minister)";
139         opt[132] = "自(self)";
140         opt[133] = "至(arrive)";
141         opt[134] = "臼(mortar)";
142         opt[135] = "舌(tongue)";
143         opt[136] = "舛(oppose)";
144         opt[137] = "舟(boat)";
145         opt[138] = "艮(stopping)";
146         opt[139] = "色(color)";
147         opt[140] = "艸(grass)";
148         opt[141] = "虍(tiger)";
149         opt[142] = "虫(insect)";
150         opt[143] = "血(blood)";
151         opt[144] = "行(walk enclosure)";
152         opt[145] = "衣(clothes)";
153         opt[146] = "西(west)";
154         opt[147] = "見(see)";
155         opt[148] = "角(horn)";
156         opt[149] = "言(speech)";
157         opt[150] = "谷(valley)";
158         opt[151] = "豆(bean)";
159         opt[152] = "豕(pig)";
160         opt[153] = "豸(badger)";
161         opt[154] = "貝(shell)";
162         opt[155] = "赤(red)";
163         opt[156] = "走(run)";
164         opt[157] = "足(foot)";
165         opt[158] = "身(body)";
166         opt[159] = "車(cart)";
167         opt[160] = "辛(bitter)";
168         opt[161] = "辰(morning)";
169         opt[162] = "辵(walk)";
170         opt[163] = "邑(city)";
171         opt[164] = "酉(wine)";
172         opt[165] = "釆(distinguish)";
173         opt[166] = "里(village)";
174         opt[167] = "金(gold)";
175         opt[168] = "長(long)";
176         opt[169] = "門(gate)";
177         opt[170] = "阜(mound)";
178         opt[171] = "隶(slave)";
179         opt[172] = "隹(short tailed bird)";
180         opt[173] = "雨(rain)";
181         opt[174] = "青(blue)";
182         opt[175] = "非(wrong)";
183         opt[176] = "面(face)";
184         opt[177] = "革(leather)";
185         opt[178] = "韋(tanned leather)";
186         opt[179] = "韭(leek)";
```

```
187         opt[180] = "音(sound)";
188         opt[181] = "頁(leaf)";
189         opt[182] = "風(wind)";
190         opt[183] = "飛(fly)";
191         opt[184] = "食(eat)";
192         opt[185] = "首(head)";
193         opt[186] = "香(fragrant)";
194         opt[187] = "馬(horse)";
195         opt[188] = "骨(bone)";
196         opt[189] = "高(tall)";
197         opt[190] = "髟(hair)";
198         opt[191] = "鬥(fight)";
199         opt[192] = "鬯(sacrificial wine)";
200         opt[193] = "鬲(cauldron)";
201         opt[194] = "鬼(ghost)";
202         opt[195] = "魚(fish)";
203         opt[196] = "鳥(bird)";
204         opt[197] = "鹵(salt)";
205         opt[198] = "鹿(deer)";
206         opt[199] = "麥(wheat)";
207         opt[200] = "麻(hemp)";
208         opt[201] = "黃(yellow)";
209         opt[202] = "黍(millet)";
210         opt[203] = "黑(black)";
211         opt[204] = "黹(embroidery)";
212         opt[205] = "黽(frog)";
213         opt[206] = "鼎(tripod)";
214         opt[207] = "鼓(drum)";
215         opt[208] = "鼠(rat)";
216         opt[209] = "鼻(bise)";
217         opt[210] = "齊(even)";
218         opt[211] = "齒(tooth)";
219         opt[212] = "龍(dragon)";
220         opt[213] = "龜(turtle)";
221         opt[214] = "龠(flute)";
222
223         for(var i=1;i<opt.length;i++){
224                 var addNode = document.createElement('option');
225                 addNode.appendChild(document.createTextNode(opt[i]));
226                 selectBox.appendChild(addNode);
227         }
228 }
229
230 //テキストエリアにイベントを設置する準備
231 function setInsEvent(no,node){
232         switch(no){
233                 case 0:
234                         if(is.gecko || is.opera){
235                                 node.addEventListener('click',setStr0,false);
236                         }
237                         else{
238                                 node.attachEvent('onclick',setStr0);
239                         }
240                         break;
241                 case 1:
242                         if(is.gecko || is.opera){
243                                 node.addEventListener('click',setStr1,false);
244                         }
245                         else{
246                                 node.attachEvent('onclick',setStr1);
247                         }
248                         break;
```

```
249                 case 2:
250                         if(is.gecko || is.opera){
251                                 node.addEventListener('click',setStr2,false);
252                         }
253                         else{
254                                 node.attachEvent('onclick',setStr2);
255                         }
256                         break;
257                 case 3:
258                         if(is.gecko || is.opera){
259                                 node.addEventListener('click',setStr3,false);
260                         }
261                         else{
262                                 node.attachEvent('onclick',setStr3);
263                         }
264                         break;
265                 case 4:
266                         if(is.gecko || is.opera){
267                                 node.addEventListener('click',setStr4,false);
268                         }
269                         else{
270                                 node.attachEvent('onclick',setStr4);
271                         }
272                         break;
273                 case 5:
274                         if(is.gecko || is.opera){
275                                 node.addEventListener('click',setStr5,false);
276                         }
277                         else{
278                                 node.attachEvent('onclick',setStr5);
279                         }
280                         break;
281                 case 6:
282                         if(is.gecko || is.opera){
283                                 node.addEventListener('click',setStr6,false);
284                         }
285                         else{
286                                 node.attachEvent('onclick',setStr6);
287                         }
288                         break;
289                 case 7:
290                         if(is.gecko || is.opera){
291                                 node.addEventListener('click',setStr7,false);
292                         }
293                         else{
294                                 node.attachEvent('onclick',setStr7);
295                         }
296                         break;
297                 case 8:
298                         if(is.gecko || is.opera){
299                                 node.addEventListener('click',setStr8,false);
300                         }
301                         else{
302                                 node.attachEvent('onclick',setStr8);
303                         }
304                         break;
305                 case 9:
306                         if(is.gecko || is.opera){
307                                 node.addEventListener('click',setStr9,false);
308                         }
309                         else{
310                                 node.attachEvent('onclick',setStr9);
```

```
311                             }
312                             break;
313                     default:
314                             break;
315             }
316 }
317
318 //テキストエリアにイベントを設置
319 function setSearchFunc(){
320             for(var i=1;i<=4;i++){
321                     var searchArea = $('search_form:search_word'+i);
322
323                     switch(i){
324                     case 1:
325                             if(is.gecko || is.opera){
326                                     searchArea.addEventListener('click',setTxtBox1,f
327 alse);
328                             }
329                             else{
330                                     searchArea.attachEvent('onclick',setTxtBox1);
331                             }
332                             break;
333                     case 2:
334                             if(is.gecko || is.opera){
335                                     searchArea.addEventListener('click',setTxtBox2,f
336 alse);
337                             }
338                             else{
339                                     searchArea.attachEvent('onclick',setTxtBox2);
340                             }
341                             break;
342                     case 3:
343                             if(is.gecko || is.opera){
344                                     searchArea.addEventListener('click',setTxtBox3,f
345 alse);
346                             }
347                             else{
348                                     searchArea.attachEvent('onclick',setTxtBox3);
349                             }
350                             break;
351                     case 4:
352                             if(is.gecko || is.opera){
353                                     searchArea.addEventListener('click',setTxtBox4,f
354 alse);
355                             }
356                             else{
357                                     searchArea.attachEvent('onclick',setTxtBox4);
358                             }
359                             break;
360                     default:
361                             break;
362                     }
363             }
364 }
365
366 //各テキストエリアに対応した漢字をcpTableより挿入
367 function setStr0(){
368             setStr2TxtBox(cpTable[0]);
369 }
370 function setStr1(){
371             setStr2TxtBox(cpTable[1]);
372 }
```

```
373 function setStr2(){
374         setStr2TxtBox(cpTable[2]);
375 }
376 function setStr3(){
377         setStr2TxtBox(cpTable[3]);
378 }
379 function setStr4(){
380         setStr2TxtBox(cpTable[4]);
381 }
382 function setStr5(){
383         setStr2TxtBox(cpTable[5]);
384 }
385 function setStr6(){
386         setStr2TxtBox(cpTable[6]);
387 }
388 function setStr7(){
389         setStr2TxtBox(cpTable[7]);
390 }
391 function setStr8(){
392         setStr2TxtBox(cpTable[8]);
393 }
394 function setStr9(){
395         setStr2TxtBox(cpTable[9]);
396 }
397
398 //書き込むテキストエリアのid属性を指定
399 function setTxtBox1(){
400         focusedArea = $('search_form:search_word1');
401 }
402 function setTxtBox2(){
403         focusedArea = $('search_form:search_word2');
404 }
405 function setTxtBox3(){
406         focusedArea = $('search_form:search_word3');
407 }
408 function setTxtBox4(){
409         focusedArea = $('search_form:search_word4');
410 }
411
412 //部首画数選択で跳ぶためのインデックスを返す
413 function toJumpRadNum(index){
414         switch(index){
415                 case 1:
416                         return 1;
417                         break;
418                 case 2:
419                         return 7;
420                         break;
421                 case 3:
422                         return 30;
423                         break;
424                 case 4:
425                         return 61;
426                         break;
427                 case 5:
428                         return 95;
429                         break;
430                 case 6:
431                         return 118;
432                         break;
433                 case 7:
434                         return 147;
```

```
435                         break;
436                 case 8:
437                         return 167;
438                         break;
439                 case 9:
440                         return 176;
441                         break;
442                 case 10:
443                         return 187;
444                         break;
445                 case 11:
446                         return 195;
447                         break;
448                 case 12:
449                         return 201;
450                         break;
451                 case 13:
452                         return 205;
453                         break;
454                 case 14:
455                         return 209;
456                         break;
457                 case 15:
458                         return 211;
459                         break;
460                 case 16:
461                         return 212;
462                         break;
463                 case 17:
464                         return 214;
465                         break;
466                 default:
467                         break;
468         }
469         return 1;
470 }
```

（5）unisearch_gene.js

unisearch_gene.js　　Page 1

```
 1 //DOM操作により、予め設置してあるdivノードにシステムを生成する為の処理
 2
 3 function generate_dummy(){
 4         var root = $('writeDummy');
 5         var onCheck = document.createElement('img');
 6         onCheck.id = "hide";
 7         onCheck.className = "onCursor";
 8         onCheck.width = 10;
 9         onCheck.height = 10;
10         var url = "http://stl30.itc.u-toyama.ac.jp/dokb/ime/";
11         onCheck.src = url+"stor.png";
12         root.appendChild(onCheck);
13         root.appendChild(document.createElement('br'));
14         var attention = document.createElement('span');
15         attention.appendChild(document.createTextNode("漢字検索システム(β) -Uni
16 codeにおけるCJK統合漢字拡張A及びBに収録されている漢字を部首ベースによる検索ができます。"));
18         attention.id = "attention";
19         var suggest = document.createElement('span');
20         var font = document.createElement('a');
21         font.href = "http://wiki.livedoor.jp/qvarie/d/%b3%c8%c4%a5%b4%c1%bb%fa%a
22 5%d5%a5%a9%a5%f3%a5%c8%b4%d8%cf%a2";
23         font.appendChild(document.createTextNode("フォント"));
24         suggest.appendChild(document.createTextNode("IE等では表示に的確な"));
25         suggest.appendChild(font);
26         suggest.appendChild(document.createTextNode("を指定する必要があるため、M
27 ozilla Firefoxによる使用を推奨します。"));
28         suggest.id = "suggest";
29
30         var writeArea = document.createElement('div');
31         writeArea.id = "writeArea";
32         root.appendChild(writeArea);
33         writeArea.appendChild(attention);
34         writeArea.appendChild(document.createElement('br'));
35         writeArea.appendChild(suggest);
36         var selectTable = document.createElement('table');
37         var selectTbody = document.createElement('tbody');
38
39         var selectTr = document.createElement('tr');
40         var radLabelTd = document.createElement('td');
41         var strokeLabelTd = document.createElement('td');
42         var jumpLabelTd = document.createElement('td');
43
44         var boxTr = document.createElement('tr');
45         var radicalTd = document.createElement('td');
46         var strokeTd = document.createElement('td');
47         var jumpTd = document.createElement('td');
48
49         selectTable.border = 0;
50         radicalTd.rowSpan = 3;
51
52         selectTr.appendChild(jumpLabelTd);
53         selectTr.appendChild(radLabelTd);
54         selectTr.appendChild(strokeLabelTd);
55         selectTbody.appendChild(selectTr);
56
57         boxTr.appendChild(jumpTd);
58         boxTr.appendChild(radicalTd);
59         boxTr.appendChild(strokeTd);
60         selectTbody.appendChild(boxTr);
61         selectTable.appendChild(selectTbody);
62
```

```
63          var radicalForm = document.createElement('form');
64          var selectRad = document.createElement('select');
65
66          selectRad.size = 6;
67          selectRad.id = "radical";
68          selectRad.multiple = "multiple";
69
70          var selectRadStroke = document.createElement('select');
71          selectRadStroke.size = 3;
72          selectRadStroke.id = "jumpRadicalStroke";
73          selectRadStroke.multiple = "multiple";
74
75          jumpLabelTd.appendChild(document.createTextNode("部首画数選択"));
76          jumpLabelTd.appendChild(document.createElement('br'));
77          jumpLabelTd.appendChild(document.createTextNode("(select strokes in radi
78 cal)"));
79          jumpTd.appendChild(selectRadStroke);
80
81          var childLabelTr = document.createElement('tr');
82          var childBoxTr = document.createElement('tr');
83          var childLabelTd = document.createElement('td');
84          var childBoxTd = document.createElement('td');
85          var td2span = document.createElement('span');
86
87          selectTbody.appendChild(childLabelTr);
88          selectTbody.appendChild(childBoxTr);
89
90          td2span.id = "searchFromChar";
91
92          var inputChar = document.createElement('input');
93          inputChar.size = 4;
94          inputChar.type = "text";
95          inputChar.id = "searchRadFromChar";
96
97          var inputButton = document.createElement('input');
98          inputButton.type = "button";
99          inputButton.id = "searchRadFromButton";
100         inputButton.value = "search";
101
102         childLabelTr.appendChild(childLabelTd);
103         td2span.appendChild(document.createTextNode("文字から部首を検索"));
104         td2span.appendChild(document.createElement('br'));
105         td2span.appendChild(document.createTextNode("(search Radical from charac
106 ter)"));
107         childLabelTd.appendChild(td2span);
108
109         var dispType = document.createElement('input');
110         var dispForm = document.createElement('form');
111         var dispBoxTd = document.createElement('td');
112         dispType.type = "checkbox";
113         dispType.id = "dispWay";
114         dispForm.appendChild(dispType);
115         dispForm.align = "center";
116         dispBoxTd.appendChild(dispForm);
117
118         var dispLabel = document.createTextNode("コードポイント表記");
119         var checkLabelTd = document.createElement('td');
120         checkLabelTd.appendChild(dispLabel);
121         childLabelTr.appendChild(checkLabelTd);
122
123         childBoxTd.appendChild(inputChar);
124         childBoxTd.appendChild(inputButton);
```

```
125         childBoxTr.appendChild(childBoxTd);
126         childBoxTr.appendChild(dispBoxTd);
127
128         radLabelTd.appendChild(document.createTextNode("部首選択"));
129         radLabelTd.appendChild(document.createElement('br'));
130         radLabelTd.appendChild(document.createTextNode("(select radical)"));
131         radicalTd.appendChild(radicalForm);
132         radicalForm.appendChild(selectRad);
133
134         var menuDiv = document.createElement('div');
135         var selectStroke = document.createElement('select');
136
137         menuDiv.id = "strokeMenu";
138         menuDiv.align = "center";
139
140         selectStroke.id = "selectDummy";
141
142         strokeLabelTd.appendChild(document.createTextNode("画数選択"));
143         strokeLabelTd.appendChild(document.createElement('br'));
144         strokeLabelTd.appendChild(document.createTextNode("(select strokes)"));
145         menuDiv.appendChild(selectStroke);
146         strokeTd.appendChild(menuDiv);
147
148         var imgArea = document.createElement('div');
149         var imgDummy = document.createElement('div');
150         imgArea.id = "imgArea";
151         imgDummy.id = "imgDummy";
152
153
154         writeArea.appendChild(selectTable);
155         imgArea.appendChild(imgDummy);
156         writeArea.appendChild(imgArea);
157
158         //セレクトボックスの中身を生成
159         start_func();
160
161         //各イベントリスナーを設定
162         if(is.gecko || is.opera){
163                 onCheck.addEventListener('click',hide,false);
164                 dispType.addEventListener('change',checkBox,false);
165                 selectStroke.addEventListener('change',stroke_function,false);
166                 selectRad.addEventListener('change',radical_function,false);
167                 selectRadStroke.addEventListener('click',jumpRad,false);
168                 inputChar.addEventListener('keypress',char2radnum,false);
169                 inputButton.addEventListener('click',char2radnumB,false);
170         }
171         else{
172                 onCheck.attachEvent('onclick',hide);
173                 dispType.attachEvent('onchange',checkBox);
174                 selectStroke.attachEvent('onchange',stroke_function);
175                 selectRad.attachEvent('onchange',radical_function);
176                 selectRadStroke.attachEvent('onclick',jumpRad);
177                 inputChar.attachEvent('onkeydown',char2radnum);
178                 inputButton.attachEvent('onclick',char2radnumB);
179         }
180
181         //cookieから前の状態を復元
182         decomp();
183 }
```

（6）unisearch_search.js

unisearch_search.js　　Page 1

```
 1 //文字から属する部首を検索する処理
 2
 3 //テキストエリア内のkeypressイベント(onkeydown)の処理 enterが押されたか判定
 4 function char2radnum(e){
 5         if(e.keyCode == 13) {
 6                 char2radnumB();
 7         }
 8 }
 9
10 //searchボタンを押した時の処理
11 function char2radnumB(){
12         var text = $('searchRadFromChar');
13         //先頭文字の文字コードを得る
14         var code = char2codepoint(text.value);
15         //文字コードから属する部首を得る
16         getRadnumFromCode(code);
17 }
18
19 //テキストエリア内の先頭の文字の16進文字コード(UTF8)を得る
20 function char2codepoint(str){
21         var surrogate_1st = 0;
22         var code = null;
23         var i = 0;
24         do {
25                 var utf16_code = str.charCodeAt(i);
26                 if (surrogate_1st != 0) {
27                         if (utf16_code >= 0xDC00 && utf16_code <= 0xDFFF) {
28                                 var surrogate_2nd = utf16_code;
29                                 var unicode_code = (surrogate_1st - 0xD800) * (1
30  << 10) + (1 << 16) +
31                                                                         (surroga
32 te_2nd - 0xDC00);
33                                 code = unicode_code;
34                         } else {
35                         // Malformed surrogate pair ignored.
36                                 code = 4E00;
37                         }
38                                 surrogate_1st = 0;
39                 } else if (utf16_code >= 0xD800 && utf16_code <= 0xDBFF) {
40                         surrogate_1st = utf16_code;
41                 } else {
42                         code = utf16_code;
43                 }
44                 i++;
45         } while(code == null);
46         return(code.toString(16).toUpperCase());
47 }
48
49 //データベースへの要求
50 function getRadnumFromCode(code){
51         var req = new Ajax.Request( cgi+"searchcode.pl?"+code, { method: 'GET',
52 onComplete: moveRadNum});
53 }
54
55 //得た部首ナンバーで部首選択のセレクトボックスを移動
56 function moveRadNum(request){
57         var tempXML = request.responseXML;
58         var root = tempXML.documentElement;
59         var jump = root.getElementsByTagName('radical_num');
60         var radnum = jump[0].firstChild.nodeValue;
61
62         var radicalSelect = $('radical');
63         radicalSelect.selectedIndex = radnum-1;
64         radical_function();
65 }
```

（7）searchcode.pl

searchcode.pl　　Page 1

```
 1 #!C:/perl/bin/perl
 2
 3 use DBI;
 4
 5 @arg = split(/&/,$ENV{'QUERY_STRING'});
 6
 7 $code = -1;
 8 $radicalNumber = -1;
 9 # 引数の判定
10 # 1番目引数が4文字以上ならコードポイントを記したものであると判定
11 if(length($arg[0]) > 3){
12         $code = $arg[0];
13 }
14 # 3文字以下なら第1引数は部首ナンバーであると判定
15 else{
16         $radicalNumber = $arg[0];
17 }
18
19 $strokeNumber = -2;
20 # 引数の数が2つある場合、2つ目の引数をストロークナンバーと判定
21 if(scalar(@arg) eq 2){
22         $strokeNumber = $arg[1];
23 }
24
25 # データベースの設定
26 our $DB_NAME     = "dbi:Pg:dbname=unihan";
27 our $DB_USERNAME = "dokb_user";
28 our $DB_PASSWORD = "ialOEI4ljprE_jov8dlope";
29
30 # データベースに接続
31 eval{
32 $DB = DBI->connect($DB_NAME, $DB_USERNAME, $DB_PASSWORD,{RaiseError => 1, Auto
33 Commit => 0 });
34 };
35 if($@){
36 print "$@";
37 }
38 my $StrorCode;
39
40 print "Content-type: text/xml;¥n¥n";
41
42 # 各データを検索
43 # codeに値が設定されている場合、コードポイントから部首ナンバーを検索
44 if($code != -1){
45         $StrorCode = $DB->selectall_arrayref(
46                                                 qq|select radicalnum from code
47 point_table |
48                                         . qq|where codepoint='$code'|
49                                                 );
50 }
51 # codeに値が設定されてなく、strokeNumberが設定されている場合、部首ナンバーとス
52 トロークナンバーからコードポイントを検索
53 elsif($strokeNumber>-1){
54         $StrorCode = $DB->selectall_arrayref(
55                                                 qq|select codepoint from codep
56 oint_table |
57                                         . qq|where radicalnum=$radicalNumber a
58 nd strokenum=$strokeNumber|
59                                                 );
60 }
61 # codeに値が設定されてなく、strokeNumberの値が-1の場合、部首ナンバーのすべての
62 コードポイントを検索
63 elsif($strokeNumber eq -1){
```

```
64          $StrorCode = $DB->selectall_arrayref(
65                                                  qq|select codepoint from codep
66 oint_table |
67                                              . qq|where radicalnum=$radicalNumber|
68                                                  );
69 }
70 # 上記以外の場合、部首ナンバーからストロークナンバーを検索
71 else{
72          $StrorCode = $DB->selectall_arrayref(
73                              qq|select strokenum from stroke_table |
74                            . qq|where radicalnum=$radicalNumber|
75                                  );
76 }
77 # データがあれば
78 if(@$StrorCode)     {
79          # 部首ナンバーのXMLを生成
80          if($code > -1){
81                  print "<radical>¥n";
82                  foreach $number(@$StrorCode){
83                          print "<radical_num>".$$number[0]."</radical_num>¥n";
84                  }
85                  print "</radical>¥n";
86          }
87          # ストロークナンバーのXMLを生成
88          elsif($strokeNumber eq -2){
89                  print "<stroke>¥n";
90                  foreach $number(@$StrorCode){
91                          print "<stroke_num>".$$number[0]."</stroke_num>¥n";
92                  }
93                  print "</stroke>¥n";
94          }
95          # コードポイントのXMLを生成
96          else{
97                  print "<code>¥n";
98                  foreach $number(@$StrorCode){
99                          print "<codepoint>".$$number[0]."</codepoint>¥n";
100                 }
101                 print "</code>"
102         }
103 }
104
105 # データベースの接続を切断
106 $DB->disconnect();
107
108 1;
```

［資料９－２］刻手名データベース検索システム開発ソース・コード

１）kokushu-db：刻手データを表すモジュール

```
kokushu-db
    pom.xml

    src¥main
      java¥jp¥ac¥u-toyama¥itc¥kokushu¥data
        Kokushu.java

    src¥test
      java¥jp¥ac¥u_toyama¥itc¥kokushu¥dao
        Kokushu_Test.java
      resources¥META-INF
        persistence.xml
```

kokushu-db

（1）pom.xml

pom.xml　　Page 1

```
 1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/
 2 2001/XMLSchema-instance"
 3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
 4 maven-v4_0_0.xsd">
 5   <parent>
 6     <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
 7     <artifactId>kokushu</artifactId>
 8     <version>1.0-SNAPSHOT</version>
 9     <relativePath>../kokushu/pom.xml</relativePath>
10   </parent>
11
12   <modelVersion>4.0.0</modelVersion>
13   <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
14   <artifactId>kokushu-db</artifactId>
15   <packaging>jar</packaging>
16   <version>1.0-SNAPSHOT</version>
17   <name>kokushu-db</name>
18
19   <repositories>
20
21     <repository>
22       <id>repository.jboss.org</id>
23           <name>JBoss Maven Repository</name>
24       <url>http://repository.jboss.org/maven2</url>
25       <layout>default</layout>
26     </repository>
27
28   </repositories>
29
30   <dependencies>
31
32     <dependency>
33       <groupId>org.hibernate</groupId>
34       <artifactId>hibernate</artifactId>
35       <version>3.2.5.ga</version>
36     </dependency>
37
38     <dependency>
39       <groupId>org.hibernate</groupId>
40       <artifactId>hibernate-annotations</artifactId>
41       <version>3.3.0.ga</version>
42     </dependency>
43
44     <dependency>
45       <groupId>org.hibernate</groupId>
46       <artifactId>hibernate-entitymanager</artifactId>
47       <version>3.3.1.ga</version>
48     </dependency>
49
50     <dependency>
51       <groupId>org.hibernate</groupId>
52       <artifactId>hibernate-search</artifactId>
53       <version>3.0.0.GA</version>
54     </dependency>
55
56     <dependency>
57       <groupId>org.apache.lucene</groupId>
58       <artifactId>lucene-analyzers</artifactId>
59       <version>2.2.0</version>
60     </dependency>
61
62     <dependency>
63       <groupId>commons-lang</groupId>
64       <artifactId>commons-lang</artifactId>
```

```
65        <version>2.3</version>
66      </dependency>
67
68      <dependency>
69        <groupId>junit</groupId>
70        <artifactId>junit</artifactId>
71        <version>4.4</version>
72        <scope>test</scope>
73      </dependency>
74
75      <!-- Hibernate EntityManager 用 -->
76      <dependency>
77        <groupId>concurrent</groupId>
78        <artifactId>concurrent</artifactId>
79        <version>1.3.4</version>
80      </dependency>
81
82      <!-- Hibernate 用 -->
83      <dependency>
84        <groupId>jboss</groupId>
85        <artifactId>jboss-cache</artifactId>
86        <version>1.2.2</version>
87      </dependency>
88
89      <dependency>
90        <groupId>jaxen</groupId>
91        <artifactId>jaxen</artifactId>
92        <version>1.1.1</version>
93      </dependency>
94
95    </dependencies>
96
97 </project>
```

src¥main¥java¥jp¥ac¥u-toyama¥itc¥kokushu¥data

（1）Kokushu.java

Kokushu.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.data;
 2
 3 import java.io.Serializable;
 4
 5 import javax.persistence.Column;
 6 import javax.persistence.Entity;
 7 import javax.persistence.Id;
 8 import javax.persistence.Table;
 9 import javax.persistence.Transient;
10
11 import org.apache.commons.lang.builder.EqualsBuilder;
12 import org.apache.commons.lang.builder.HashCodeBuilder;
13 import org.apache.lucene.analysis.cjk.CJKAnalyzer;
14 import org.hibernate.search.annotations.Analyzer;
15 import org.hibernate.search.annotations.DocumentId;
16 import org.hibernate.search.annotations.Field;
17 import org.hibernate.search.annotations.Indexed;
18
19 /**
20  * <p>刻手データです。</p>
21  * <p>可変オブジェクトのため、このクラスはスレッドセーフではありません。</p>
22  *
23  * @author Keita Kita (2008/02/13 -)
24  * @since 1.0-SNAPSHOT
25  */
26 @Entity
27 @Table(name="kokushu")
28 @Indexed
29 @Analyzer(impl=CJKAnalyzer.class)
30 public class Kokushu implements Serializable {
31         /**
32          * <p>直列化の際に参照される識別番号です。</p>
33          */
34         private static final long serialVersionUID = 3556655785227571387L;
35
36
37         /**
38          * <p>出典がオリジナルであることを示す文字列です。</p>
39          */
40         private static final String ORIGINAL_REFERENCE = "ORIGINAL";
41
42         /**
43          * <p>「排列番號」を表わします。</p>
44          */
45         private String id = "";
46
47         /**
48          * <p>「出典」を表わします。</p>
49          */
50         @Field
51         private String reference = "";
52
53         /**
54          * <p>「書名・巻数・冊数」を表わします。</p>
55          */
56         @Field
57         private String title = "";
58
59         /**
60          * <p>「撰者」を表わします。</p>
61          */
62         @Field
63         private String author = "";
64
```

```
65          /**
66           * <p>「刊年」を表わします。</p>
67           */
68          @Field
69          private String year = "";
70
71          /**
72           * <p>「刊地」を表わします。</p>
73           */
74          @Field
75          private String place = "";
76
77          /**
78           * <p>「所蔵者」を表わします。</p>
79           */
80          @Field
81          private String owner = "";
82
83          /**
84           * <p>「版心部刻手名」を表わします。</p>
85           */
86          @Field
87          private String centerKokushuName = "";
88
89          /**
90           * <p>「刊記部刻手名」を表わします。</p>
91           */
92          @Field
93          private String imprintKokushuName = "";
94
95
96          /**
97           * <p>引数なしのコンストラクタです。</p>
98           */
99          public Kokushu() {
100                 // 処理なし
101         }
102
103
104         /**
105          * <p>出典がオリジナルかどうかを返します。</p>
106          *
107          * @return 出典がオリジナルな場合は true、他からの出典の場合は false
108          */
109         @Transient
110         public boolean isOriginal() {
111                 return ORIGINAL_REFERENCE.equals(this.reference);
112         }
113
114         /**
115          * <p>「排列番號」を返します。</p>
116          *
117          * @return 排列番號
118          */
119         @Id
120         @DocumentId
121         @Column(name="id", length=5, nullable=false, unique=true)
122         public String getId() {
123                 return this.id;
124         }
125
126         /**
127          * <p>「排列番號」を設定します。</p>
128          *
```

```
129          * @param id 排列番號
130          */
131         public void setId(String id) {
132                 this.id = id;
133         }
134
135         /**
136          * <p>「出典」を返します。</p>
137          *
138          * @return 出典
139          */
140         @Column(name="reference", nullable=false)
141         public String getReference() {
142                 return this.reference;
143         }
144
145         /**
146          * <p>「出典」を設定します。</p>
147          *
148          * @param reference 出典
149          */
150         public void setReference(String reference) {
151                 this.reference = reference;
152         }
153
154         /**
155          * <p>「書名・巻數（巻数）・册數（冊数）」を返します。</p>
156          *
157          * @return 書名・巻數（巻数）・册數（冊数）
158          */
159         @Column(name="title", nullable=false)
160         public String getTitle() {
161                 return this.title;
162         }
163
164         /**
165          * <p>「書名・巻數（巻数）・册數（冊数）」を設定します。</p>
166          *
167          * @param title 書名・巻數（巻数）・册數（冊数）
168          */
169         public void setTitle(String title) {
170                 this.title = title;
171         }
172
173         /**
174          * <p>「撰者」を返します。</p>
175          *
176          * @return 撰者
177          */
178         @Column(name="author", nullable=false)
179         public String getAuthor() {
180                 return this.author;
181         }
182
183         /**
184          * <p>「撰者」を設定します。</p>
185          *
186          * @param author 撰者
187          */
188         public void setAuthor(String author) {
189                 this.author = author;
190         }
191
192         /**
```

```
193          * <p>「刊年」を返します。</p>
194          *
195          * @return 刊年
196          */
197         @Column(name="year", nullable=false)
198         public String getYear() {
199                 return this.year;
200         }
201
202         /**
203          * <p>「刊年」を設定します。</p>
204          *
205          * @param year 刊年
206          */
207         public void setYear(String year) {
208                 this.year = year;
209         }
210
211         /**
212          * <p>「刊地」を返します。</p>
213          *
214          * @return 刊地
215          */
216         @Column(name="place", nullable=false)
217         public String getPlace() {
218                 return this.place;
219         }
220
221         /**
222          * <p>「刊地」を設定します。</p>
223          *
224          * @param place 刊地
225          */
226         public void setPlace(String place) {
227                 this.place = place;
228         }
229
230         /**
231          * <p>「所藏者」を返します。</p>
232          *
233          * @return 所藏者
234          */
235         @Column(name="owner", nullable=false)
236         public String getOwner() {
237                 return this.owner;
238         }
239
240         /**
241          * <p>「所藏者」を設定します。</p>
242          *
243          * @param owner 所藏者
244          */
245         public void setOwner(String owner) {
246                 this.owner = owner;
247         }
248
249         /**
250          * <p>「版心部刻手名」を返します。</p>
251          *
252          * @return 版心部刻手名
253          */
254         @Column(name="center_kokushu_name", nullable=false)
255         public String getCenterKokushuName() {
256                 return this.centerKokushuName;
```

```
257         }
258
259         /**
260          * <p>「版心部刻手名」を設定します。</p>
261          *
262          * @param centerKokushuName 版心部刻手名
263          */
264         public void setCenterKokushuName(String centerKokushuName) {
265                 this.centerKokushuName = centerKokushuName;
266         }
267
268         /**
269          * <p>「刊記部刻手名」を返します。</p>
270          *
271          * @return 刊記部刻手名
272          */
273         @Column(name="imprint_kokushu_name", nullable=false)
274         public String getImprintKokushuName() {
275                 return this.imprintKokushuName;
276         }
277
278         /**
279          * <p>「刊記部刻手名」を設定します。</p>
280          *
281          * @param imprintKokushuName 刊記部刻手名
282          */
283         public void setImprintKokushuName(String imprintKokushuName) {
284                 this.imprintKokushuName = imprintKokushuName;
285         }
286
287         @Override
288         public boolean equals(Object object) {
289                 if (this == object) {
290                         return true;
291                 }
292
293                 if (!(object instanceof Kokushu)) {
294                         return false;
295                 }
296
297                 Kokushu another = (Kokushu) object;
298
299                 // 排列番號が同じなら同じ刻手データだと考える
300                 // どちらかの排列番號が null の場合は、排列番號以外のデータ全て
301 を比較する
302
303                 if (this.id != null && another.getId() != null) {
304                         return (this.id.equals(another.getId()));
305                 }
306
307                 return (new EqualsBuilder().
308                         append(this.reference, another.getReference()).
309                         append(this.title, another.getTitle()).
310                         append(this.author, another.getAuthor()).
311                         append(this.year, another.getYear()).
312                         append(this.place, another.getPlace()).
313                         append(this.owner, another.getOwner()).
314                         append(this.centerKokushuName, another.getCenterKokushuN
315 ame()).
316                         append(this.imprintKokushuName, another.getImprintKokush
317 uName())).
318                         isEquals();
319         }
320
```

```
321         @Override
322         public int hashCode() {
323                 // 排列番號をハッシュコードとする
324
325                 if (this.id != null ) {
326                         return this.id.hashCode();
327                 }
328
329                 // 排列番號が null の場合は、排列番號以外のデータ全てからハッシ
330 ュコードを
331                 // 生成する
332
333                 return (new HashCodeBuilder().
334                         append(this.reference).
335                         append(this.title).
336                         append(this.author).
337                         append(this.year).
338                         append(this.place).
339                         append(this.owner).
340                         append(this.centerKokushuName).
341                         append(this.imprintKokushuName)).hashCode();
342         }
343 }
```

src¥test¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥dao

（1）KokushuTest.java

KokushuTest.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.dao;
 2
 3 import javax.persistence.EntityManager;
 4 import javax.persistence.EntityManagerFactory;
 5 import javax.persistence.Persistence;
 6
 7 import org.junit.After;
 8 import org.junit.AfterClass;
 9 import org.junit.Before;
10 import org.junit.BeforeClass;
11 import org.junit.Test;
12
13
14 /**
15  * <p>Kokushu が正しくデータベースにマッピングされているかどうかテストします。</
16 p>
17  *
18  * @author Keita Kita (2008/02/15 -)
19  * @since 1.0-SNAPSHOT
20  */
21 public class KokushuTest {
22         /**
23          * <p>EntityManagerFactory です。</p>
24          * <p>全オブジェクトで共有します。</p>
25          */
26         private static EntityManagerFactory factory;
27
28         /**
29          * <p>EntityManager です。</p>
30          */
31         private EntityManager manager;
32
33
34         // 以下、準備用のメソッド
35
36         /**
37          * <p>このクラスの全テストが始まる前に呼び出されます。</p>
38          */
39         @BeforeClass
40         public static void setUpFactory() {
41                 factory = Persistence.createEntityManagerFactory("testManager");
42
43         }
44
45         /**
46          * <p>このクラスの全テストが終了すると呼び出されます。</p>
47          */
48         @AfterClass
49         public static void tearDownFactory() {
50                 factory.close();
51         }
52
53         /**
54          * <p>各テストごとに、実行される前に呼び出されます。</p>
55          */
56         @Before
57         public void setUp() {
58                 this.manager = factory.createEntityManager();
59         }
60
61         /**
62          * <p>各テストごとに、実行が終了した後に呼び出されます。</p>
63          */
64         @After
```

```
65          public void tearDown() {
66                  this.manager.close();
67          }
68
69
70          // 以下、テストメソッド
71
72          /**
73           * <p>全データの取得をテストします。</p>
74           */
75          @Test
76          public void testGetAll() {
77                  this.manager.createQuery("SELECT k FROM Kokushu AS k").getResult
78  List();
79          }
80  }
```

resources¥META-INF

（1）persistence.xml

persistence.xml　　Page 1

```
 1 <?xml version="1.0" encoding="UTF-8" ?>
 2 <persistence xmlns="http://java.sun.com/xml/ns/persistence"
 3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4     xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.c
 5 om/xml/ns/persistence/persistence_1_0.xsd"
 6     version="1.0">
 7
 8
 9   <persistence-unit name="testManager" transaction-type="RESOURCE_LOCAL">
10     <provider>org.hibernate.ejb.HibernatePersistence</provider>
11
12     <class>jp.ac.u_toyama.itc.kokushu.data.Kokushu</class>
13
14     <properties>
15
16       <property name="hibernate.dialect"
17           value="org.hibernate.dialect.PostgreSQLDialect" />
18       <property name="hibernate.connection.driver_class"
19           value="org.postgresql.Driver" />
20       <property name="hibernate.connection.username" value="kokushu_viewer" />
21       <property name="hibernate.connection.password"
22           value="t18eErM7PRGGZeq9zgW6DWoBh5ZAUS" />
23       <property name="hibernate.connection.url"
24           value="jdbc:postgresql://localhost/kokushu_db" />
25       <property name="hibernate.show_sql" value="true" />
26
27       <property name="hibernate.search.default.directory_provider"
28           value="org.hibernate.search.store.FSDirectoryProvider" />
29       <property name="hibernate.search.default.indexBase"
30           value="target¥indexes" />
31
32     </properties>
33
34   </persistence-unit>
35
36 </persistence>
```

２）kokushu-ejb：刻手データを検索・取得するモジュール

```
kokushu-ejb
    pom.xml

    src¥main
      ¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥business
        ¥impl
          KokushuViewerImpl.java
        DataPage.java
        KokushuViewer.java
        PersistentUnitName.java

      ¥resources¥META-INF
        persistence.xml

    src¥test
      なし
```

kokushu-ejb

（1）pom.xml

pom.xml　　Page 1

```
 1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/
 2 2001/XMLSchema-instance"
 3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
 4 maven-v4_0_0.xsd">
 5   <parent>
 6     <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
 7     <artifactId>kokushu</artifactId>
 8     <version>1.0-SNAPSHOT</version>
 9     <relativePath>../kokushu/pom.xml</relativePath>
10   </parent>
11
12   <modelVersion>4.0.0</modelVersion>
13   <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
14   <artifactId>kokushu-ejb</artifactId>
15   <packaging>ejb</packaging>
16   <version>1.0-SNAPSHOT</version>
17   <name>kokushu-ejb</name>
18
19   <build>
20
21     <plugins>
22
23       <!-- EJB 3.0 -->
24       <plugin>
25         <groupId>org.apache.maven.plugins</groupId>
26         <artifactId>maven-ejb-plugin</artifactId>
27         <configuration>
28           <ejbVersion>3.0</ejbVersion>
29         </configuration>
30       </plugin>
31
32     </plugins>
33
34     <resources>
35
36       <resource>
37         <directory>src/main/resources</directory>
38         <filtering>true</filtering>
39       </resource>
40
41     </resources>
42
43   </build>
44
45   <dependencies>
46
47     <dependency>
48       <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
49       <artifactId>kokushu-db</artifactId>
50       <version>1.0-SNAPSHOT</version>
51     </dependency>
52
53     <dependency>
54       <groupId>javaee</groupId>
55       <artifactId>javaee-api</artifactId>
56       <version>5</version>
57       <scope>provided</scope>
58     </dependency>
59
60     <dependency>
61       <groupId>commons-beanutils</groupId>
62       <artifactId>commons-beanutils</artifactId>
63       <version>1.7.0</version>
64     </dependency>
```

```
65
66      <dependency>
67        <groupId>junit</groupId>
68        <artifactId>junit</artifactId>
69        <version>4.4</version>
70        <scope>test</scope>
71      </dependency>
72
73    </dependencies>
74
75 </project>
```

src¥main¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥business¥impl

（1）KokushuViewerImpl.java

KokushuViewerImpl.java Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.business.impl;
 2
 3 import java.lang.reflect.InvocationTargetException;
 4 import java.util.ArrayList;
 5 import java.util.List;
 6
 7 import javax.ejb.Stateless;
 8 import javax.persistence.EntityManager;
 9 import javax.persistence.PersistenceContext;
10 import javax.persistence.Query;
11
12 import jp.ac.u_toyama.itc.kokushu.business.DataPage;
13 import jp.ac.u_toyama.itc.kokushu.business.KokushuViewer;
14 import jp.ac.u_toyama.itc.kokushu.business.PersistentUnitName;
15 import jp.ac.u_toyama.itc.kokushu.data.Kokushu;
16
17 import org.apache.commons.beanutils.BeanUtils;
18 import org.apache.commons.lang.Validate;
19 import org.apache.lucene.analysis.cjk.CJKAnalyzer;
20 import org.apache.lucene.queryParser.MultiFieldQueryParser;
21 import org.apache.lucene.queryParser.ParseException;
22 import org.hibernate.search.jpa.FullTextEntityManager;
23 import org.hibernate.search.jpa.FullTextQuery;
24 import org.hibernate.search.jpa.Search;
25
26
27 /**
28  * <p>KokushuViewer の実装です。
29  *
30  * @author Keita Kita (2008/02/19 -)
31  * @since 1.0-SNAPSHOT
32  */
33 @Stateless
34 public class KokushuViewerImpl implements KokushuViewer {
35         /**
36          * <p>全文検索で検索対象とするフィールドを表す配列です。</p>
37          */
38         private static final String[] FIELDS = {
39                 "title", "author", "year", "place", "owner",
40                 "center_kokushu_name", "imprint_kokushu_name", "reference"};
41
42         /**
43          * <p>EntityManager です。</p>
44          */
45         @PersistenceContext(unitName=PersistentUnitName.FOR_VIEW)
46         private EntityManager manager;
47
48
49         /**
50          * <p>引数なしのコンストラクタです。</p>
51          */
52         public KokushuViewerImpl() {
53                 // 処理なし
54         }
55
56         /**
57          * <p>EntityManager を指定できるコンストラクタです。
58          *
59          * @param manager EntityManager
60          */
61         protected KokushuViewerImpl(EntityManager manager) {
62                 Validate.notNull(manager);
63
64                 this.manager = manager;
```

```
65          }
66
67          public DataPage<Kokushu> getList(int firstIndex, int size) {
68                  return new DataPage<Kokushu>(this.getTotal(), firstIndex,
69                          createKokushuListForView(
70                                  this.getKokushuEntities(firstIndex, size)));
71          }
72
73          public DataPage<Kokushu> getList(String query, int firstIndex, int size)
74  {
75                  return new DataPage<Kokushu>(this.getTotal(query), firstIndex,
76                          createKokushuListForView(
77                                  this.getKokushuEntities(query, firstIndex, size)
78 ));
79          }
80
81          /**
82           * <p>刻手データのリストを取得します。</p>
83           *
84           * @param firstIndex 取得する初めのインデックス
85           * @param size 取得するデータ数
86           * @return 刻手データのエンティティオブジェクトのリスト
87           */
88          @SuppressWarnings("unchecked")
89          private List<Kokushu> getKokushuEntities(int firstIndex, int size) {
90                  Query query = this.manager.createQuery("SELECT k FROM Kokushu AS
91  k");
92
93                  query.setFirstResult(firstIndex);
94                  query.setMaxResults(size);
95
96                  return query.getResultList();
97          }
98
99          /**
100          * <p>検索した刻手データのリストを取得します。</p>
101          *
102          * @param queryString 問い合わせ文
103          * @param firstIndex 取得する初めのインデックス
104          * @param size 取得するデータ数
105          * @return 刻手データのエンティティオブジェクトのリスト
106          * @throws RuntimeException queryString が空の文字列の場合
107          */
108         @SuppressWarnings("unchecked")
109         private List<Kokushu> getKokushuEntities(
110                         String queryString, int firstIndex, int size) {
111                 // 全文検索を行う EntityManager
112                 FullTextEntityManager fullTextManager =
113                         Search.createFullTextEntityManager(this.manager);
114
115                 Query query;
116
117                 try {
118                         query = fullTextManager.createFullTextQuery(parse(queryS
119 tring));
120                 }
121                 catch (ParseException e) {
122                         throw new RuntimeException(e);
123                 }
124
125                 query.setFirstResult(firstIndex);
126                 query.setMaxResults(size);
127
128                 return query.getResultList();
```

```
129         }
130
131         /**
132          * <p>解析済みの検索語を返します。</p>
133          *
134          * @param queryString 解析を行う検索語
135          * @return 解析済みの検索語
136          * @throws ParseException 検索語の構文にエラーがある場合
137          */
138         private static org.apache.lucene.search.Query parse(String queryString)
139                         throws ParseException {
140                 // 問い合わせ文を組み立てるパーサ
141                 MultiFieldQueryParser parser = new MultiFieldQueryParser(
142                         FIELDS, new CJKAnalyzer());
143
144                 return parser.parse(queryString);
145         }
146
147         /**
148          * <p>刻手データの Entity Bean のリストから、閲覧用のオブジェクトのリス
149 トを
150          * 生成します。</p>
151          *
152          * @param entities 刻手データの Entity Bean のリスト
153          * @return 閲覧用のオブジェクトのリスト
154          */
155         private static List<Kokushu> createKokushuListForView(
156                         List<Kokushu> entities ) {
157                 List<Kokushu> kokushuList = new ArrayList<Kokushu>(entities.size
158 ());
159
160                 for (Kokushu anEntity : entities) {
161                         kokushuList.add(createKokushuForView(anEntity));
162                 }
163
164                 return kokushuList;
165         }
166
167         /**
168          * <p>刻手データの Entity Bean から、閲覧用のオブジェクトを生成します。<
169 /p>
170          *
171          * @param entity 刻手データの Entity Bean
172          * @return 閲覧用のオブジェクト
173          */
174         private static Kokushu createKokushuForView(Kokushu entity) {
175                 Kokushu kokushu = new Kokushu();
176
177                 // Lazy Fetch されるものはないため、プロパティをそのままコピー
178                 try {
179                         BeanUtils.copyProperties(kokushu, entity);
180                 }
181                 catch (IllegalAccessException e) {
182                         throw new RuntimeException(e);
183                 }
184                 catch (InvocationTargetException e) {
185                         throw new RuntimeException(e);
186                 }
187
188                 return kokushu;
189         }
190
191         public int getTotal() {
192                 Query query = this.manager.createQuery(
```

```
193                         "SELECT COUNT(k) FROM Kokushu AS k");
194
195                 return ((Long) query.getSingleResult()).intValue();
196         }
197
198         public int getTotal(String query) {
199                 FullTextEntityManager fullTextManager =
200                         Search.createFullTextEntityManager(this.manager);
201                 FullTextQuery fullTextQuery;
202
203                 try {
204                         fullTextQuery = fullTextManager.createFullTextQuery(pars
205 e(query));
206                 }
207                 catch (ParseException e) {
208                         throw new RuntimeException(e);
209                 }
210
211                 return fullTextQuery.getResultSize();
212         }
213
214         public boolean checkQuerySyntax(String query) {
215                 try {
216                         parse(query);
217                 }
218                 // 検索語の構文にエラーがある場合
219                 catch (ParseException e) {
220                         return false;
221                 }
222
223                 // 検索語の構文にエラーがない場合
224                 return true;
225         }
226
227         public Kokushu getFromId(String id) {
228                 Validate.notNull(id);
229
230                 // 刻手データの Entity Bean
231                 Kokushu anEntity = this.manager.find(Kokushu.class, id);
232
233                 return createKokushuForView(anEntity);
234         }
235 }
```

src¥main¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥business

（1）DataPage.java

DataPage.java Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.business;
 2
 3
 4 import java.io.Serializable;
 5 import java.util.List;
 6
 7 /**
 8  * A simple class that represents a "page" of data out of a longer set, ie
 9  * a list of objects together with info to indicate the starting row and
10  * the full size of the dataset. EJBs can return instances of this type
11  * when returning subsets of available data.
12  *
13  * @param <T> the type of holded object
14  *
15  * @author http://wiki.apache.org/myfaces/WorkingWithLargeTables
16  * @version 1.0-SNAPSHOT
17  */
18 public class DataPage<T> implements Serializable {
19         /**
20          *
21          */
22         private static final long serialVersionUID = 4792094303227836029L;
23
24         /**
25          *
26          */
27         private int datasetSize;
28
29         /**
30          *
31          */
32         private int startRow;
33
34         /**
35          *
36          */
37         private List<T> data;
38
39
40         /**
41          * Create an object representing a sublist of a dataset.
42          *
43          * @param datasetSize is the total number of matching rows
44          * available.
45          *
46          * @param startRow is the index within the complete dataset
47          * of the first element in the data list.
48          *
49          * @param data is a list of consecutive objects from the
50          * dataset.
51          */
52         public DataPage(int datasetSize, int startRow, List<T> data) {
53                 this.datasetSize = datasetSize;
54                 this.startRow = startRow;
55                 this.data = data;
56         }
57
58         /**
59          * Return the number of items in the full dataset.
60          *
61          * @return the number of items
62          */
63         public int getDatasetSize() {
64                 return this.datasetSize;
```

```
65          }
66
67          /**
68           * Return the offset within the full dataset of the first
69           * element in the list held by this object.
70           *
71           * @return the offest with the full dataset of the first element in the
72 list
73           */
74          public int getStartRow() {
75                  return this.startRow;
76          }
77
78          /**
79           * Return the list of objects held by this object, which
80           * is a continuous subset of the full dataset.
81           *
82           * @return the list of obect
83           */
84          public List<T> getData() {
85                  return this.data;
86          }
87 }
88
```

（２）KokushuViewer.java

KokushuViewer.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.business;
 2
 3 import javax.ejb.Remote;
 4
 5 import jp.ac.u_toyama.itc.kokushu.data.Kokushu;
 6
 7 /**
 8  * <p>刻手データを閲覧するビジネスインタフェースです。</p>
 9  * <p>検索構文は、Apache Lucene の構文に従います。</p>
10  *
11  * @author Keita Kita (2008/02/19 -)
12  * @since 1.0-SNAPSHOT
13  */
14 @Remote
15 public interface KokushuViewer {
16         /**
17          * <p>ID から閲覧用の刻手データを取得します。</p>
18          * <p>刻手データが存在しないは null を返します。</p>
19          *
20          * @param id 刻手 ID
21          * @return 刻手データ、存在しない場合は null
22          * @throws IllegalArgumentException 引数が null の場合
23          */
24         Kokushu getFromId(String id);
25
26         /**
27          * <p>一覧用のデータを取得します。</p>
28          *
29          * @param firstIndex 取得する初めのインデックス
30          * @param size 取得する数
31          * @return 一覧用のデータ
32          */
33         DataPage<Kokushu> getList(int firstIndex, int size);
34
35         /**
36          * <p>一覧用の検索済みデータを取得します。</p>
37          * <p>このメソッドを使用する前に、検索語の構文のチェックを行ってください
38 。</p>
39          *
40          * @see #checkQuerySyntax(String)
41          *
42          * @param query 検索語
43          * @param firstIndex 取得する初めのインデックス
44          * @param size 取得する数
45          * @return 一覧用のデータ
46          * @throws RuntimeException query が空の文字列の場合、構文にエラーがある
47 場合
48          */
49         DataPage<Kokushu> getList(String query, int firstIndex, int size);
50
51         /**
52          * <p>指定の検索語の構文が正しいかどうかを検証します。</p>
53          *
54          * @param query 検証する検索語
55          * @return 検索語の構文が正しい場合は true、エラーがある場合は false
56          */
57         boolean checkQuerySyntax(String query);
58
59         /**
60          * <p>刻手データの総数を返します。</p>
61          *
62          * @return 刻手データの総数
63          */
64         int getTotal();
```

```
65
66          /**
67           * <p>検索済みの刻手データの総数を返します。</p>
68           * <p>このメソッドを使用する前に、検索語の構文のチェックを行ってください
69 。</p>
70           *
71           * @see #checkQuerySyntax(String)
72           *
73           * @param query 検索語
74           * @return 検索を行った刻手データの総数
75           * @throws RuntimeException query が空の文字列の場合、検索語の構文にエラ
76 ーが
77           * ある場合
78           */
79          int getTotal(String query);
80 }
```

（3）PersistentUnitName.java

PersistentUnitName.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.business;
 2
 3 /**
 4  * <p>Persistent Unit 名の定数を提供するクラスです。</p>
 5  *
 6  * @author Keita Kita (2008/02/20 -)
 7  * @since 1.0-SNAPSHOT
 8  */
 9 public class PersistentUnitName {
10         /**
11          * <p>閲覧用の Persistent Unit 名です。</p>
12          */
13         public static final String FOR_VIEW = "kokushuViewer";
14
15
16         /**
17          * <p>外部からは生成できません。</p>
18          */
19         private PersistentUnitName() {
20                 // 処理なし
21         }
22 }
```

¥resources¥META-INF

（1）persistence.xml

persistence.xml　　Page 1

```
 1 <?xml version="1.0" encoding="UTF-8" ?>
 2 <persistence xmlns="http://java.sun.com/xml/ns/persistence"
 3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4     xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.c
 5 om/xml/ns/persistence/persistence_1_0.xsd"
 6     version="1.0">
 7
 8
 9   <persistence-unit name="kokushuViewer" transaction-type="JTA">
10     <provider>org.hibernate.ejb.HibernatePersistence</provider>
11
12     <jta-data-source>jdbc/KokushuDB</jta-data-source>
13
14     <class>jp.ac.u_toyama.itc.kokushu.data.Kokushu</class>
15
16     <properties>
17
18       <property name="hibernate.dialect"
19           value="org.hibernate.dialect.PostgreSQLDialect" />
20       <property name="hibernate.connection.driver_class"
21           value="org.postgresql.Driver" />
22
23       <property name="hibernate.show_sql" value="false" />
24
25       <property name="hibernate.search.default.directory_provider"
26           value="org.hibernate.search.store.FSDirectoryProvider" />
27       <property name="hibernate.search.default.indexBase"
28           value="${indexDirectory}" />
29
30     </properties>
31
32   </persistence-unit>
33
34   <persistence-unit name="indexerViewer" transaction-type="RESOURCE_LOCAL">
35     <provider>org.hibernate.ejb.HibernatePersistence</provider>
36
37
38     <class>jp.ac.u_toyama.itc.kokushu.data.Kokushu</class>
39
40     <properties>
41
42       <property name="hibernate.dialect"
43           value="org.hibernate.dialect.PostgreSQLDialect" />
44       <property name="hibernate.connection.driver_class"
45           value="org.postgresql.Driver" />
46
47       <property name="hibernate.connection.username" value="kokushu_viewer" />
48       <property name="hibernate.connection.password"
49           value="t18eErM7PRGGZeq9zgW6DWoBh5ZAUS" />
50       <property name="hibernate.connection.url"
51           value="jdbc:postgresql://localhost/kokushu_db" />
52
53
54       <property name="hibernate.show_sql" value="false" />
55
56       <property name="hibernate.search.default.directory_provider"
57           value="org.hibernate.search.store.FSDirectoryProvider" />
58       <property name="hibernate.search.default.indexBase"
59           value="${indexDirectory}" />
60
61     </properties>
62
63   </persistence-unit>
64
65
66 </persistence>
```

３）kokushu-indexer：刻手データベースから全文検索用のインデックスを作成する
アプリケーション

```
kokushu-indexer
    pom.xml

    ¥src¥main
      ¥assembly
        bin.xml
      ¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥application¥indexer
        KokushuManualIndexer.java
      ¥resources
        commons-logging.properties
        simplelog.properties
      ¥script
        kokushu-indexer.bat

   ¥src test¥java
      なし
```

kokushu-indexer

（1）pom.xml

pom.xml　　Page 1

```
 1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/
 2 2001/XMLSchema-instance"
 3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
 4 maven-v4_0_0.xsd">
 5   <parent>
 6     <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
 7     <artifactId>kokushu</artifactId>
 8     <version>1.0-SNAPSHOT</version>
 9     <relativePath>../kokushu/pom.xml</relativePath>
10   </parent>
11
12   <modelVersion>4.0.0</modelVersion>
13   <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
14   <artifactId>kokushu-indexer</artifactId>
15   <packaging>jar</packaging>
16   <version>1.0-SNAPSHOT</version>
17   <name>kokushu-indexer</name>
18
19   <build>
20
21     <plugins>
22       <plugin>
23         <artifactId>maven-assembly-plugin</artifactId>
24         <configuration>
25           <descriptors>
26             <descriptor>src/main/assembly/bin.xml</descriptor>
27           </descriptors>
28         </configuration>
29       </plugin>
30
31       <plugin>
32         <artifactId>maven-surefire-plugin</artifactId>
33         <configuration>
34           <skip>true</skip>
35         </configuration>
36       </plugin>
37
38     </plugins>
39   </build>
40
41   <dependencies>
42
43     <dependency>
44       <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
45       <artifactId>kokushu-db</artifactId>
46       <version>1.0-SNAPSHOT</version>
47     </dependency>
48
49     <dependency>
50       <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
51       <artifactId>kokushu-ejb</artifactId>
52       <version>1.0-SNAPSHOT</version>
53     </dependency>
54
55     <dependency>
56       <groupId>junit</groupId>
57       <artifactId>junit</artifactId>
58       <version>4.4</version>
59       <scope>test</scope>
60     </dependency>
61
62   </dependencies>
63
64 </project>
```

¥src¥main¥assembly

（1）bin.xml

bin.xml　　Page 1

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <assembly>
 3
 4   <id>bin</id>
 5
 6   <formats>
 7     <format>zip</format>
 8   </formats>
 9
10   <includeBaseDirectory>false</includeBaseDirectory>
11
12   <dependencySets>
13
14     <dependencySet>
15       <outputDirectory>/lib</outputDirectory>
16       <outputFileNameMapping>${artifact.artifactId}.${artifact.extension}</outpu
17 tFileNameMapping>
18       <unpack>false</unpack>
19       <scope>runtime</scope>
20     </dependencySet>
21
22   </dependencySets>
23
24   <fileSets>
25
26     <fileSet>
27
28       <includes>
29         <include>README*</include>
30       </includes>
31
32       <outputDirectory>/</outputDirectory>
33
34     </fileSet>
35
36     <fileSet>
37
38       <directory>src/main/script</directory>
39
40       <outputDirectory>/</outputDirectory>
41
42     </fileSet>
43
44     <fileSet>
45
46       <directory>target</directory>
47
48       <includes>
49
50         <include>*.jar</include>
51
52       </includes>
53
54       <outputDirectory>/</outputDirectory>
55
56     </fileSet>
57
58   </fileSets>
59
60 </assembly>
```

¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥application¥indexer

（1）KokushuManualIndexer.java

KokushuManualIndexer.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.application.indexer;
 2
 3 import java.util.List;
 4
 5 import javax.persistence.EntityManager;
 6 import javax.persistence.EntityManagerFactory;
 7 import javax.persistence.EntityTransaction;
 8 import javax.persistence.Persistence;
 9
10 import org.apache.commons.lang.Validate;
11 import org.hibernate.search.jpa.FullTextEntityManager;
12 import org.hibernate.search.jpa.Search;
13
14 /**
15  * <p>刻手データをインデックス化するアプリケーションです。</p>
16  *
17  * @author Keita Kita (2008/02/19 -)
18  * @since 1.0-SNAPSHOT
19  */
20 public class KokushuManualIndexer {
21         /**
22          * <p>EntityManager のユニット名です。</p>
23          */
24         private static final String UNIT_NAME = "indexerViewer";
25
26         /**
27          * <p>EntityManager です。</p>
28          */
29         private final FullTextEntityManager manager;
30
31
32         /**
33          * <p>コンストラクタです。</p>
34          *
35          * @param manager EntityManager
36          * @throws IllegalArgumentException 引数が null の場合
37          */
38         public KokushuManualIndexer(EntityManager manager) {
39                 Validate.notNull(manager);
40
41                 this.manager = Search.createFullTextEntityManager(manager);
42         }
43
44         /**
45          * <p>インデックスを作成します。</p>
46          */
47         public void index() {
48                 EntityTransaction transaction = this.manager.getTransaction();
49
50                 transaction.begin();
51
52                 List kokushuList = this.manager.createQuery(
53                         "SELECT k FROM Kokushu AS k").getResultList();
54
55                 for (Object kokushu : kokushuList) {
56                         this.manager.index(kokushu);
57                 }
58
59                 transaction.commit();
60         }
61
62         /**
63          * <p>メインメソッドです。</p>
64          *
```

```
65          * @param args コマンドライン引数
66          */
67         public static void main(String[] args) {
68                 EntityManagerFactory factory =
69                         Persistence.createEntityManagerFactory(UNIT_NAME);
70
71                 System.out.println("インデックス化を始めます。");
72
73                 KokushuManualIndexer indexer = new KokushuManualIndexer(
74                         factory.createEntityManager());
75
76                 try {
77                         indexer.index();
78                 }
79                 catch (Exception e) {
80                         System.out.println("インデックス化の作業中にエラーが発生
81 しました。");
82                         e.printStackTrace();
83                         System.out.println("インデックス化に失敗しました。");
84                         System.exit(1);
85                 }
86                 finally {
87                         factory.close();
88                 }
89
90                 System.out.println("インデックス化に成功しました。");
91         }
92 }
```

¥src¥main¥resources

（1）commons-logging.properties

commons-logging.properties　　Page 1

```
1 org.apache.commons.logging.Log = org.apache.commons.logging.impl.SimpleLog
2 org.apache.commons.logging.simplelog.defaultlog = warn
```

（2）simplelog.properties

simplelog.properties　　Page 1

```
1 org.apache.commons.logging.simplelog.defaultlog = fatal
```

¥src¥main¥script

（1）kokushu-indexer.bat

kokushu-indexer.bat　　Page 1

```
1 java -classpath *;lib/* jp.ac.u_toyama.itc.kokushu.application.indexer.KokushuMa
2 nualIndexer %*
```

4）kokushu-webapp：刻手データを検索するウェブアプリケーション

```
kokushu-webapp
    pom.xml

    src¥main
      ¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥webapp
        ¥catalogue
          KokushuCataloguePage.java
          KokushuDetailPage.java
        ¥converter
          EmptyStringConverter.java
        ¥model
          EmptyStringConverter.java
        ¥search
          Query.java
      ¥resources　なし
      ¥webapp
        ¥authentication
          error.jsp
          index.jsp
        ¥catalogue
          detail.jsp
          index.jsp
        ¥stylesheets
          catalogue.css
          catalogue.css.orig
          common.css
          detail.css
          login.css
          top.css
        ¥WEB-INF
          face-config.xml
          sun-web.xml
          web.xml
        index2.jsp
        index.jsp
        main_page.jsp

    src¥test
      ¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥webapp¥search
        QueryTest.java
```

kokushu-webapp

（1）pom.xml

pom.xml　　Page 1

```
 1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.or
 2 g/2001/XMLSchema-instance"
 3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.or
 4 g/maven-v4_0_0.xsd">
 5   <parent>
 6     <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
 7     <artifactId>kokushu</artifactId>
 8     <version>1.0-SNAPSHOT</version>
 9     <relativePath>../kokushu/pom.xml</relativePath>
10   </parent>
11
12   <modelVersion>4.0.0</modelVersion>
13   <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
14   <artifactId>kokushu-webapp</artifactId>
15   <packaging>war</packaging>
16   <version>1.0-SNAPSHOT</version>
17   <name>kokushu-webapp Maven Webapp</name>
18
19   <dependencies>
20
21     <dependency>
22       <groupId>junit</groupId>
23       <artifactId>junit</artifactId>
24       <version>4.4</version>
25       <scope>test</scope>
26     </dependency>
27
28     <dependency>
29       <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
30       <artifactId>kokushu-db</artifactId>
31       <version>1.0-SNAPSHOT</version>
32       <scope>provided</scope>
33     </dependency>
34
35     <dependency>
36       <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
37       <artifactId>kokushu-ejb</artifactId>
38       <version>1.0-SNAPSHOT</version>
39       <scope>provided</scope>
40     </dependency>
41
42    <dependency>
43       <groupId>javax.faces</groupId>
44       <artifactId>jsf-impl</artifactId>
45       <version>1.2_07</version>
46       <scope>provided</scope>
47     </dependency>
48
49     <dependency>
50       <groupId>commons-lang</groupId>
51       <artifactId>commons-lang</artifactId>
52       <version>2.3</version>
53     </dependency>
54
55     <dependency>
56       <groupId>javaee</groupId>
57       <artifactId>javaee-api</artifactId>
58       <version>5</version>
59       <scope>provided</scope>
60     </dependency>
61
62     <dependency>
```

```
63        <groupId>org.apache.myfaces.tomahawk</groupId>
64        <artifactId>tomahawk</artifactId>
65        <version>1.1.6</version>
66      </dependency>
67
68      <dependency>
69        <groupId>org.apache.myfaces.trinidad</groupId>
70        <artifactId>trinidad-api</artifactId>
71        <version>1.2.7</version>
72      </dependency>
73
74      <dependency>
75        <groupId>org.apache.myfaces.trinidad</groupId>
76        <artifactId>trinidad-impl</artifactId>
77        <version>1.2.7</version>
78      </dependency>
79
80      <dependency>
81        <groupId>taglibs</groupId>
82        <artifactId>standard</artifactId>
83        <version>1.1.2</version>
84      </dependency>
85
86    </dependencies>
87
88    <build>
89      <finalName>kokushu-webapp</finalName>
90      <sourceDirectory>src/main/java</sourceDirectory>
91      <testSourceDirectory>src/test/java</testSourceDirectory>
92    </build>
93
94  </project>
```

src¥main¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥webapp¥catalogue

（1）KokushuCataloguePage.java

KokushuCataloguePage.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.webapp.catalogue;
 2
 3 import java.util.Collections;
 4 import java.util.List;
 5 import java.util.regex.Matcher;
 6 import java.util.regex.Pattern;
 7
 8 import javax.ejb.EJB;
 9 import javax.faces.application.FacesMessage;
10 import javax.faces.component.UIComponent;
11 import javax.faces.component.UIData;
12 import javax.faces.component.UIInput;
13 import javax.faces.context.FacesContext;
14 import javax.faces.model.DataModel;
15
16 import jp.ac.u_toyama.itc.kokushu.business.DataPage;
17 import jp.ac.u_toyama.itc.kokushu.business.KokushuViewer;
18 import jp.ac.u_toyama.itc.kokushu.data.Kokushu;
19 import jp.ac.u_toyama.itc.kokushu.webapp.model.PagedListDataModel;
20 import jp.ac.u_toyama.itc.kokushu.webapp.search.Query;
21
22 import org.apache.commons.lang.Validate;
23 import org.apache.myfaces.trinidad.context.RequestContext;
24
25
26
27 /**
28  * <p>刻手データの一覧ページの Managed Bean です。</p>
29  * <p>このクラスはセッション間で共有されるため、スレッドセーフです。</p>
30  *
31  * @author Keita Kita (2008/02/22 -)
32  * @since 1.0-SNAPSHOT
33  */
34 public class KokushuCataloguePage {
35         /**
36          * <p>刻手データを1度に表示する数です。</p>
37          */
38         private static final int PAGE_SIZE = 50;
39
40         /**
41          * <p>詳細情報を表示するための結果文字列です。</p>
42          */
43         private static final String DETAIL_OUTCOME = "detail";
44
45         /**
46          * <p>詳細情報を表示する刻手データの ID を格納するキーです。</p>
47          */
48         public static final String DETAILED_KOKUSHU_ID_KEY = "detailedKokushuId"
49 ;
50
51         /**
52          * <p>検索文中の不正な文字を検出するための正規表現パターンです。</p>
53          */
54         private static final Pattern INVALID_CARACTERS = Pattern.compile(
55                 "[" + Query.getStringOfInvalidCharacterClasses() + "]");
56
57         /**
58          * <p>標準の検索方法です。</p>
59          */
60         private static final Query.Method DEFAULT_SEARCH_METHOD =
61                 Query.Method.AND;
62
```

```
63          /**
64           * <p>一覧ページの状態を表す列挙です。</p>
65           */
66          private enum PageState {
67                  /**
68                   * <p>初めて到達した状態を表します。</p>
69                   */
70                  INITIAL,
71
72                  /**
73                   * <p>検索を行った状態を表します。</p>
74                   */
75                  SEARCHED,
76
77                  /**
78                   * <p>全てのデータを表示した状態を表します。</p>
79                   */
80                  ALL,
81          }
82
83
84          /**
85           * <p>刻手データの一覧を取得する EJB です。</p>
86           */
87          @EJB
88          private KokushuViewer viewer;
89
90          /**
91           * <p>刻手データを取得する DataModel です。</p>
92           */
93          private DataModel kokushuDataModel =
94                  new NullKokushuListDataModel(PAGE_SIZE);
95
96          /**
97           * <p>現在のページの状態です。</p>
98           * <p>スレッドセーフにするため、synchronized 文で保護されます。</p>
99           */
100         private PageState currentState = PageState.INITIAL;
101
102         /**
103          * <p>検索語です。</p>
104          */
105         private volatile String keyword = "";
106
107         /**
108          * <p>検索方法です。</p>
109          */
110         private Query.Method searchMethod = DEFAULT_SEARCH_METHOD;
111
112         /**
113          * <p>選択される項目を持つ UI コンポーネントです。</p>
114          */
115         private UIData selectedComponent;
116
117
118         /**
119          * <p>検索語を設定します。</p>
120          *
121          * @param keyword 検索語
122          * @throws IllegalArgumentException 引数が null の場合
123          */
124         public void setKeyword(String keyword) {
```

```
125                 Validate.notNull(keyword);
126
127                 this.keyword = keyword;
128         }
129
130         /**
131          * <p>検索語を返します。</p>
132          *
133          * @return 検索語
134          */
135         public String getKeyword() {
136                 assert (this.keyword != null);
137
138                 return this.keyword;
139         }
140
141         /**
142          * <p>検索方法を返します。</p>
143          *
144          * @return 検索方法を表す文字列
145          */
146         public String getSearchMethod() {
147                 return this.searchMethod.getDescription();
148         }
149
150         /**
151          * <p>検索方法のインデックスを返します。</p>
152          *
153          * @return 検索方法のインデックス
154          */
155         public String getSearchMethodIndex() {
156                 return this.searchMethod.getIndexString();
157         }
158
159         /**
160          * <p>検索方法のインデックスを設定します。</p>
161          *
162          * @param searchMethod 検索方法のインデックス
163          * @throws IllegalArgumentException 引数が不正な場合
164          */
165         public void setSearchMethodIndex(String searchMethod) {
166                 this.searchMethod = Query.Method.indexOf(searchMethod);
167         }
168
169         /**
170          * <p>検索語の構文にエラーがないかどうか検証を行います。</p>
171          * <p>エラーがある場合は、検索語を入力するコンポーネントにメッセージを
172          * 設定します。</p>
173          * <p>これは、JSF の Validation メソッドです。</p>
174          *
175          * @param context FacesContext
176          * @param component UIComponent
177          * @param value 検証対象の値
178          */
179         public void validateKeyword(FacesContext context,
180                         UIComponent component, Object value) {
181                 // 不正な文字とのマッチを行うオブジェクト
182                 Matcher matcherForInvalidity =
183                         INVALID_CARACTERS.matcher((String) value);
184
185                 if (matcherForInvalidity.find()) {
186                         // 検索語を入力するコンポーネントにエラーメッセージを設
```

```
187 定
188
189                         ((UIInput) component).setValid(false);
190                         context.addMessage(component.getClientId(context),
191                                 new FacesMessage(FacesMessage.SEVERITY_WARN,
192                                         "検索語の構文にエラーがあります。",
193                                         "検索語の構文にエラーがあります。")
194                                 );
195                         return;
196                 }
197
198                 // 不正な文字が見つからなかった場合は、何もしない
199         }
200
201
202         /**
203          * <p>刻手データを一度に表示する数を返します。</p>
204          *
205          * @return 刻手データを一度に表示する数
206          */
207         public int getPageSize() {
208                 return PAGE_SIZE;
209         }
210
211         /**
212          * <p>刻手データを表す DataModel を返します。</p>
213          *
214          * @return 刻手データを表す DataModel
215          */
216         public DataModel getItems() {
217                 return this.kokushuDataModel;
218         }
219
220         /**
221          * <p>刻手データの総数を返します。</p>
222          *
223          * @return 刻手データの総数
224          */
225         public int getTotal() {
226                 return this.viewer.getTotal();
227         }
228
229         /**
230          * <p>結果の総数を返します。</p>
231          *
232          * @return 結果の総数
233          */
234         public int getResultCount() {
235                 return this.kokushuDataModel.getRowCount();
236         }
237
238         /**
239          * <p>ユーザがこのページを初めて訪れたかどうかを返します。</p>
240          *
241          * @return ユーザが初めて訪れた場合は true、そうでない場合は false
242          */
243         public synchronized boolean isInitial() {
244                 assert (this.currentState != null);
245
246                 return (this.currentState == PageState.INITIAL);
247         }
248
```

```
249         /**
250          * <p>現在、検索結果を表示する状態かどうかを返します。</p>
251          *
252          * @return 現在、検索結果を表示する状態の場合は true、そうでない場合は f
253 alse
254          */
255         public synchronized boolean isSearched() {
256                 assert (this.currentState != null);
257
258                 return (this.currentState == PageState.SEARCHED);
259         }
260
261         /**
262          * <p>現在、全ての刻手データを表示する状態かどうかを返します。</p>
263          *
264          * @return 現在、全ての刻手データを表示する状態の場合は true、
265          *       そうでない場合は false
266          */
267         public synchronized boolean isAll() {
268                 assert (this.currentState != null);
269
270                 return (this.currentState == PageState.ALL);
271         }
272
273         /**
274          * <p>結果が存在するかどうかを返します。</p>
275          *
276          * @return 結果が存在する場合は true 、存在しない場合は false
277          */
278         public synchronized boolean isResultExistent() {
279                 assert (this.currentState != null);
280
281                 // 初期状態の場合は、false
282                 if (this.currentState == PageState.INITIAL) {
283                         return false;
284                 }
285
286                 // 以下、結果が存在する状態
287
288                 return (0 < this.kokushuDataModel.getRowCount());
289         }
290
291
292         /**
293          * <p>選択される項目を持つ UI コンポーネントを取得します。</p>
294          *
295          * @return 選択される項目を持つ UI コンポーネント
296          */
297         public UIData getSelectedComponent() {
298                 return this.selectedComponent;
299         }
300
301         /**
302          * <p>選択される項目を持つ UI コンポーネントを設定します。</p>
303          *
304          * @param selectedComponent 選択される項目を持つ UI コンポーネント
305          */
306         public void setSelectedComponent(UIData selectedComponent) {
307                 this.selectedComponent = selectedComponent;
308         }
309
310
```

```
311         /*
312          * 以下、アクションメソッド
313          */
314
315
316         /**
317          * <p>検索を行います。</p>
318          *
319          * @return 結果文字列（他のページへは遷移しないため、空の文字列）
320          */
321         public String search() {
322                 // 検索語が空の場合は、全ての刻手データを表示する
323                 if ("".equals(this.keyword)) {
324                         return this.showAll();
325                 }
326
327                 this.currentState = PageState.SEARCHED;
328
329                 this.kokushuDataModel = new SearchedKokushuListDataModel(
330                         Query.createQuery(this.keyword, this.searchMethod), PAGE
331 _SIZE);
332
333                 this.resetTableIndex();
334
335                 return "";
336         }
337
338         /**
339          * <p>全ての刻手データを表示します。</p>
340          *
341          * @return 結果文字列（他のページへは遷移しないため、空の文字列）
342          */
343         public String showAll() {
344                 // 全結果表示状態へ移行
345                 this.currentState = PageState.ALL;
346
347                 this.kokushuDataModel = new AllKokushuListDataModel(PAGE_SIZE);
348
349                 this.resetTableIndex();
350
351                 return "";
352         }
353
354         /**
355          * <p>テーブルを初期状態に戻します。</p>
356          */
357         private void resetTableIndex() {
358                 this.selectedComponent.setFirst(0);
359         }
360
361         /**
362          * <p>選択された刻手データの詳細を表示します。</p>
363          *
364          * @return 結果文字列
365          */
366         public String showDetail() {
367                 // 選択された刻手データの ID
368                 String selectedId = ((Kokushu)
369                         this.selectedComponent.getRowData()).getId();
370
371                 // 選択されたデータの ID を記憶
372                 RequestContext context = RequestContext.getCurrentInstance();
```

```
373                 context.getPageFlowScope().put(DETAILED_KOKUSHU_ID_KEY, selected
374 Id);
375
376                 return DETAIL_OUTCOME;
377         }
378
379         /*
380          * 以下、内部クラス
381          */
382
383         /**
384          * <p>刻手データを取得しない DataModel です。
385          */
386         private class NullKokushuListDataModel extends
387                         PagedListDataModel<Kokushu> {
388                 /**
389                  * <p>コンストラクタ。</p>
390                  *
391                  * @param pageSize 一度に表示するページ数
392                  */
393                 public NullKokushuListDataModel(int pageSize) {
394                         super(pageSize);
395                 }
396
397                 @Override
398                 public DataPage<Kokushu> fetchPage(int startRow, int pageSize) {
399
400                         List<Kokushu> nullKokushuList= Collections.emptyList();
401
402                         return new DataPage<Kokushu>(0, 0, nullKokushuList);
403                 }
404         }
405
406         /**
407          * <p>全刻手データを取得する DataModel です。</p>
408          */
409         private class AllKokushuListDataModel extends
410                         PagedListDataModel<Kokushu> {
411                 /**
412                  * <p>コンストラクタです。</p>
413                  *
414                  * @param pageSize 1度に表示するデータ数
415                  */
416                 public AllKokushuListDataModel(int pageSize) {
417                         super(pageSize);
418                 }
419
420                 @Override
421                 public DataPage<Kokushu> fetchPage(int startRow, int pageSize) {
422
423                         return KokushuCataloguePage.this.viewer.
424                                 getList(startRow, pageSize);
425                 }
426         }
427
428         /**
429          * <p>検索された刻手データを取得する DataModel です。</p>
430          */
431         private class SearchedKokushuListDataModel extends
432                         PagedListDataModel<Kokushu> {
433                 /**
434                  * <p>検索語です。</p>
```

```
435                  */
436                 private final String keyword;
437
438
439                 /**
440                  * <p>コンストラクタです。</p>
441                  *
442                  * @param keyword 検索語
443                  * @param pageSize 1度に表示するデータ数
444                  * @throws IllegalArgumentException keyword が空の文字列や null
445 の場合
446                  */
447                 public SearchedKokushuListDataModel(String keyword, int pageSize
448 ) {
449                         super(pageSize);
450
451                         Validate.notEmpty(keyword);
452
453                         this.keyword = keyword;
454                 }
455
456                 @Override
457                 public DataPage<Kokushu> fetchPage(int startRow, int pageSize) {
458
459                         return KokushuCataloguePage.this.viewer.getList(
460                                 this.keyword, startRow, pageSize);
461                 }
462         }
463 }
```

src¥main¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥webapp¥catalogue

（2）KokushuDetailPage.java

KokushuDetailPage.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.webapp.catalogue;
 2
 3 import javax.ejb.EJB;
 4
 5 import jp.ac.u_toyama.itc.kokushu.business.KokushuViewer;
 6 import jp.ac.u_toyama.itc.kokushu.data.Kokushu;
 7
 8 /**
 9  * <p>刻手データの詳細情報を表示するページです。</p>
10  * <p>このクラスはスレッドセーフではありません。</p>
11  *
12  * @author Keita Kita (2008/03/06 -)
13  * @since 1.0-SNAPSHOT
14  */
15 public class KokushuDetailPage {
16         /**
17          * <p>刻手データを取得するオブジェクトです。</p>
18          */
19         @EJB
20         private KokushuViewer viewer;
21
22         /**
23          * <p>詳細を表示する刻手 ID です。</p>
24          */
25         private String kokushuId;
26
27         /**
28          * <p>詳細を表示する刻手データです。</p>
29          */
30         private Kokushu kokushu;
31
32
33         /**
34          * <p>刻手データが存在するかどうかを返します。</p>
35          * <p>刻手データの詳細を表示する前に、このメソッドで刻手データの存在を
36          * 確かめる必要があります。</p>
37          *
38          * @return 刻手データが存在する場合は true、そうでない場合は false
39          */
40         public boolean isExistent() {
41                 // 刻手 ID が設定されていない、もしくは刻手データが存在しない場
42 合は false
43
44                 return (this.kokushuId != null);
45         }
46
47         /**
48          * <p>設定された刻手 ID から、刻手データを読み込む。</p>
49          *
50          * @throws IllegalStateException 刻手 ID が設定されていない場合
51          */
52         private void setKokushu() {
53                 if (this.kokushuId == null) {
54                         throw new IllegalStateException("刻手 ID が設定されてい
55 ません。");
56                 }
57
58                 this.kokushu = this.viewer.getFromId(this.kokushuId);
59         }
60
61
62         /**
```

```
63          * <p>刻手データを返します。</p>
64          *
65          * @return 刻手データ
66          * @throws IllegalStateException 設定された刻手データが存在しない場合
67          */
68         public Kokushu getKokushu() {
69                 if (this.kokushu == null) {
70                         throw new IllegalStateException("刻手データが存在しませ
71 ん。");
72                 }
73
74                 return this.kokushu;
75         }
76
77         /**
78          * <p>詳細を表示する刻手 ID を設定します。</p>
79          *
80          * @param id 詳細を表示する刻手 ID
81          */
82         public void setKokushuId(String id) {
83                 this.kokushuId = id;
84
85                 // 刻手 ID が設定された場合は、読み込んでおく
86                 if (this.kokushuId != null) {
87                         this.setKokushu();
88                 }
89         }
90
91         /**
92          * <p>詳細を表示する刻手 ID を返します。</p>
93          *
94          * @return 詳細を表示する刻手 ID、設定されていない場合は null
95          */
96         public String getKokushuId() {
97                 return this.kokushuId;
98         }
99 }
```

¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥converter

（1）EmptyStringConverter.java

EmptyStringConverter.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.webapp.converter;
 2
 3 import javax.faces.component.UIComponent;
 4 import javax.faces.context.FacesContext;
 5 import javax.faces.convert.Converter;
 6
 7 /**
 8  * <p>空白の文字列を、記号に置き換えます。</p>
 9  * <p>空白の文字列には、以下が含まれます。</p>
10  * <ul>
11  *  <li>空の文字列</li>
12  *  <li>半角スペースのみからなる文字列</li>
13  * </ul>
14  * <p>このクラスは、状態を持たないため、スレッドセーフです。</p>
15  *
16  * @author Keita Kita (2008/03/26 -)
17  * @since 1.0-SNAPSHOT
18  */
19 public class EmptyStringConverter implements Converter {
20         /**
21          * <p>空白の文字列を置き換える記号です。</p>
22          */
23         private static final String REPLACING = "－";
24
25
26         public Object getAsObject(
27                         FacesContext context, UIComponent component, String valu
28 e) {
29                 if (context == null || component == null) {
30                         throw new NullPointerException();
31                 }
32
33                 if (value == null) {
34                         return REPLACING;
35                 }
36
37                 if (value.trim().isEmpty()) {
38                         return REPLACING;
39                 }
40                 else {
41                         return value;
42                 }
43         }
44
45         public String getAsString(
46                         FacesContext context, UIComponent component, Object valu
47 e) {
48                 return (String) this.getAsObject(context, component, (String) va
49 lue);
50         }
51 }
```

¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥model

（1）EmptyStringConverter.java

PagedListDataModel.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.webapp.model;
 2
 3
 4 import javax.faces.model.DataModel;
 5
 6 import jp.ac.u_toyama.itc.kokushu.business.DataPage;
 7
 8
 9 /**
10  * <p>A special type of JSF DataModel to allow a datatable and datascroller
11  * to page through a large set of data without having to hold the entire
12  * set of data in memory at once.</p>
13  * <p>
14  * Any time a managed bean wants to avoid holding an entire dataset,
15  * the managed bean should declare an inner class which extends this
16  * class and implements the fetchData method. This method is called
17  * as needed when the table requires data that isn't available in the
18  * current data page held by this object.</p>
19  * <p>
20  * This does require the managed bean (and in general the business
21  * method that the managed bean uses) to provide the data wrapped in
22  * a DataPage object that provides info on the full size of the dataset.</p>
23  *
24  * @param <T> the type of data contained List
25  *
26  * @author http://wiki.apache.org/myfaces/WorkingWithLargeTables
27  * @since 1.0-SNAPSHOT
28  */
29 public abstract class PagedListDataModel<T> extends DataModel {
30         /**
31          *
32          */
33     int pageSize;
34
35     /**
36      *
37      */
38     int rowIndex;
39
40     /**
41      *
42      */
43     DataPage<T> page;
44
45     /**
46      * Create a datamodel that pages through the data showing the specified
47      * number of rows on each page.
48      *
49      * @param pageSize
50      */
51     public PagedListDataModel(int pageSize) {
52         super();
53         this.pageSize = pageSize;
54         this.rowIndex = -1;
55         this.page = null;
56     }
57
58     /**
59      * Not used in this class; data is fetched via a callback to the
60      * fetchData method rather than by explicitly assigning a list.
61      */
62     @Override
```

```
63      public void setWrappedData(Object o) {
64          throw new UnsupportedOperationException("setWrappedData");
65      }
66
67      @Override
68      public int getRowIndex() {
69          return this.rowIndex;
70      }
71
72      /**
73       * Specify what the "current row" within the dataset is. Note that
74       * the UIData component will repeatedly call this method followed
75       * by getRowData to obtain the objects to render in the table.
76       */
77      @Override
78      public void setRowIndex(int index) {
79          this.rowIndex = index;
80      }
81
82      /**
83       * Return the total number of rows of data available (not just the
84       * number of rows in the current page!).
85       */
86      @Override
87      public int getRowCount() {
88          return getPage().getDatasetSize();
89      }
90
91      /**
92       * Return a DataPage object; if one is not currently available then
93       * fetch one. Note that this doesn't ensure that the datapage
94       * returned includes the current rowIndex row; see getRowData.
95       * @return DataPage
96       */
97      private DataPage<T> getPage() {
98          if (this.page != null)
99              return this.page;
100
101         int rowIndex = getRowIndex();
102         int startRow = rowIndex;
103         if (rowIndex == -1) {
104             // even when no row is selected, we still need a page
105             // object so that we know the amount of data available.
106            startRow = 0;
107         }
108
109         // invoke method on enclosing class
110         this.page = fetchPage(startRow, this.pageSize);
111         return this.page;
112     }
113
114     /**
115      * Return the object corresponding to the current rowIndex.
116      * If the DataPage object currently cached doesn't include that
117      * index then fetchPage is called to retrieve the appropriate page.
118      */
119     @Override
120     public Object getRowData(){
121         if (this.rowIndex < 0) {
122             throw new IllegalArgumentException(
123                 "Invalid rowIndex for PagedListDataModel; not within page");
124         }
```

```
125
126         // ensure page exists; if rowIndex is beyond dataset size, then
127         // we should still get back a DataPage object with the dataset size
128         // in it...
129         if (this.page == null) {
130                 this.page = fetchPage(this.rowIndex, this.pageSize);
131         }
132
133         // Check if rowIndex is equal to startRow,
134         // useful for dynamic sorting on pages
135
136         if (this.rowIndex == this.page.getStartRow()){
137                 this.page = fetchPage(this.rowIndex, this.pageSize);
138         }
139
140         int datasetSize = this.page.getDatasetSize();
141         int startRow = this.page.getStartRow();
142         int nRows = this.page.getData().size();
143         int endRow = startRow + nRows;
144
145         if (this.rowIndex >= datasetSize) {
146             throw new IllegalArgumentException("Invalid rowIndex");
147         }
148
149         if (this.rowIndex < startRow) {
150                 this.page = fetchPage(this.rowIndex, this.pageSize);
151             startRow = this.page.getStartRow();
152         } else if (this.rowIndex >= endRow) {
153                 this.page = fetchPage(this.rowIndex, this.pageSize);
154             startRow = this.page.getStartRow();
155         }
156
157         return this.page.getData().get(this.rowIndex - startRow);
158     }
159
160     @Override
161     public Object getWrappedData() {
162         return this.page.getData();
163     }
164
165     /**
166      * Return true if the rowIndex value is currently set to a
167      * value that matches some element in the dataset. Note that
168      * it may match a row that is not in the currently cached
169      * DataPage; if so then when getRowData is called the
170      * required DataPage will be fetched by calling fetchData.
171      */
172     @Override
173     public boolean isRowAvailable() {
174         DataPage<T> page = getPage();
175         if (page == null)
176             return false;
177
178         int rowIndex = getRowIndex();
179         if (rowIndex < 0) {
180             return false;
181         } else if (rowIndex >= page.getDatasetSize()) {
182             return false;
183         } else {
184             return true;
185         }
186     }
```

```
187
188     /**
189      * Method which must be implemented in cooperation with the
190      * managed bean class to fetch data on demand.
191      * @param startRow
192      * @param pageSize
193      * @return DataPage
194      */
195     public abstract DataPage<T> fetchPage(int startRow, int pageSize);
196 }
```

¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥search

（1）Query.java

Query.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.webapp.search;
 2
 3 import java.util.Arrays;
 4 import java.util.Collections;
 5 import java.util.Iterator;
 6 import java.util.LinkedList;
 7 import java.util.List;
 8 import java.util.regex.Pattern;
 9
10 import org.apache.commons.lang.ArrayUtils;
11 import org.apache.commons.lang.Validate;
12 import org.apache.lucene.queryParser.QueryParser;
13
14 /**
15  * <p>検索用の問い合わせを生成するクラスです。</p>
16  * <p>このクラスは静的クラスなため、スレッドセーフです。</p>
17  *
18  * @author Keita Kita (2008/03/12 -)
19  * @since 1.0-SNAPSHOT
20  */
21 public final class Query {
22         /**
23          * <p>検索方法を表す列挙です。</p>
24          * <p>ページでは各列挙のインデックスによって検索方法を設定するため、
25          * 本列挙の順序を変えてはなりません。</p>
26          */
27         public enum Method {
28                 /**
29                  * <p>AND 検索を表します。</p>
30                  */
31                 AND("AND"),
32
33                 /**
34                  * <p>OR 検索を表します。</p>
35                  */
36                 OR("OR");
37
38
39                 /**
40                  * <p>Apache Lucene の構文を表します。</p>
41                  */
42                 private final String syntax;
43
44
45                 /**
46                  * <p>コンストラクタです。</p>
47                  *
48                  * @param syntax Apache Lucene の構文
49                  */
50                 Method(String syntax) {
51                         this.syntax = syntax;
52                 }
53
54                 /**
55                  * <p>Apache Lucene の演算子を返します。</p>
56                  *
57                  * @return Apache Lucene の演算子
58                  */
59                 String getOperator() {
60                         return this.syntax;
61                 }
62
```

```
63                  /**
64                   * <p>インデックスの文字列表現を返します。</p>
65                   *
66                   * @return インデックスの文字列表現
67                   */
68                  public String getIndexString() {
69                          return Integer.toString(ArrayUtils.indexOf(Method.values
70 (), this));
71                  }
72
73                  /**
74                   * <p>検索方法の文字列表現を返します。</p>
75                   * <p>例えば、AND 検索の場合は、「AND」となります。</p>
76                   *
77                   * @return 検索方法の文字列表現
78                   */
79                  public String getDescription() {
80                          return this.syntax;
81                  }
82
83                  /**
84                   * <p>インデックスの文字列表現から、検索方法を返します。</p>
85                   *
86                   * @param indexString インデックスの文字列表現
87                   * @return 検索方法
88                   * @throws IllegalArgumentException 引数が不正な場合
89                   */
90                  public static Method indexOf(String indexString) {
91                          // 引数の検索方法を列挙のインデックスに変換した値
92                          int index;
93
94                          try {
95                                  index = Integer.parseInt(indexString);
96                          }
97                          catch (NumberFormatException e) {
98                                  throw new IllegalArgumentException("検索方法の値
99 が不正です。", e);
100                         }
101
102                         try {
103                                 return Method.values()[index];
104                         }
105                         catch (IndexOutOfBoundsException e) {
106                                 throw new IllegalArgumentException("検索方法の値
107 が不正です。", e);
108                         }
109                 }
110         }
111
112         /**
113          * <p>入力を受け付けない文字のリストです。</p>
114          * <p>各要素は、その文字を表す正規表現の文字クラスです。</p>
115          */
116         private static final List<String> INVALID_CHARACTER_CLASSES =
117                 Collections.unmodifiableList(
118                         Arrays.asList("¥¥p{Cc}", "¥¥n", "¥¥r", "¥¥f"));
119
120         /**
121          * <p>検索語の構文を表す正規表現パターンです。</p>
122          */
123         // 制御文字・改行文字以外の文字のみから構成される
124         private static final Pattern QUERY_PATTERN =
```

```
125                     Pattern.compile("[^" + getStringOfInvalidCharacterClasses() + "]
126 +");
127
128         /**
129          * <p>Apache Lucene の Method に定義されている以外の演算子です。</p>
130          */
131         private static final List OPERATORS = Arrays.asList("AND", "OR", "NOT");
132
133
134         /**
135          * <p>Apache Lucene の特殊文字（演算子など）です。</p>
136          * <p>参照：<a href="http://lucene.apache.org/java/2_3_1/queryparsersynt
137 ax.html#Escaping%20Special%20Characters">
138          * Escaping Special Characters</a></p>
139          */
140 //      private static final List SPECIAL_CHARACTERS = Arrays.asList(
141 //              "+", "-", "&&", "||", "!", "(", ")", "{", "}", "[", "]", "^", "¥
142 "",
143 //              "~", "*", "?", ":", "¥¥");
144
145
146         /**
147          * <p>検索語の構文が正しいかどうかを返します。</p>
148          * <p>以下の場合、不正な構文とみなされます。</p>
149          *
150          * <ul>
151          *   <li>引数が null の場合</li>
152          *   <li>文字列が空の場合</li>
153          *   <li>空白しかない場合</li>
154          *   <li>制御文字が含まれている場合</li>
155          *   <li>改行文字が含まれている場合</li>
156          * </ul>
157          *
158          * @param query 構文を検証する文字列
159          * @return 検索語の構文が正しい場合は true、正しくない場合は false
160          */
161         public static boolean isRight(String query) {
162                 if (query == null) {
163                         return false;
164                 }
165
166                 return ((!splitQueries(query).isEmpty()) &&
167                         QUERY_PATTERN.matcher(query).matches());
168         }
169
170         /**
171          * <p>不正な文字を表した、正規表現の文字クラスの列を返します。</p>
172          * <p>例えば、「a」と「b」が不正な文字の場合、本メソッドの戻り値は、「ab
173 」と
174          * なります。</p>
175          *
176          * @return 不正な文字を表した、正規表現の文字クラスの列
177          */
178         public static String getStringOfInvalidCharacterClasses() {
179                 StringBuilder stringOfInvalidity = new StringBuilder();
180
181                 for (String anInvalidity : INVALID_CHARACTER_CLASSES) {
182                         stringOfInvalidity.append(anInvalidity);
183                 }
184
185                 return stringOfInvalidity.toString();
186         }
```

```
187
188         /**
189          * <p>検索文を生成します。</p>
190          * <p>引数の検索文内の特殊文字は、メソッド内部でエスケープされます。</p>
191          *
192          * @see #isRight(String)
193          *
194          * @param input 検索語をスペース（全角・半角）で区切った文字列
195          * @param method 検索方法
196          * @return 検索文
197          * @throws IllegalArgumentException 引数が null の場合、検索語がない場合
198          *      検索語が不正な場合
199          */
200         public static String createQuery(String input, Method method) {
201                 Validate.notNull(input);
202                 Validate.notNull(method);
203                 Validate.isTrue(isRight(input));
204
205                 // 最終結果の検索文
206                 StringBuilder query = new StringBuilder();
207                 // 最初の検索語かどうかを表すフラグ
208                 boolean isFirst = true;
209
210                 // 特殊文字がエスケープ済みの検索文を処理する
211                 for (String aQuery : splitQueries(QueryParser.escape(input))) {
212                         // はじめの検索語の前には演算子を付けない
213                         if (!isFirst) {
214                                 appendOperator(query, method);
215                         }
216
217                         appendWord(query, aQuery, isFirst);
218
219                         isFirst = false;
220                 }
221
222                 return query.toString();
223         }
224
225         /**
226          * <p>演算子を検索文に追加します。</p>
227          *
228          * @param builder 追加対象の StringBuilder
229          * @param operator 追加対象の演算子
230          */
231         private static void appendOperator(StringBuilder builder,
232                         Method operator) {
233                 assert (builder != null);
234                 assert (operator != null);
235
236                 builder.append(" ");
237                 builder.append(operator.getOperator());
238         }
239
240         /**
241          * <p>語を検索文に追加します。</p>
242          *
243          * @param builder 追加対象の StringBuilder
244          * @param word 追加対象の語
245          * @param isFirst はじめの語かどうか
246          */
247
248
```

```
249         private static void appendWord(StringBuilder builder,
250                         String word, boolean isFirst) {
251                 assert (builder != null);
252                 assert (word != null);
253
254                 // はじめの語には空白を付けない
255                 if (!isFirst) {
256                         builder.append(" ");
257                 }
258
259                 // 演算子の場合は、ダブルクオートでくくる
260                 if (isOperator(word)) {
261                         builder.append("¥"");
262                         builder.append(word);
263                         builder.append("¥"");
264                 }
265                 else {
266                         builder.append(word);
267                 }
268         }
269
270         /**
271          * <p>指定の文字が演算子かどうかを返します。</p>
272          *
273          * @param word 演算子かどうかを確かめる語
274          * @return 演算子の場合は true、そうでない場合は false
275          */
276         private static boolean isOperator(String word) {
277                 assert (word != null);
278
279                 return OPERATORS.contains(word);
280         }
281
282         /**
283          * <p>検索文を分割します。</p>
284          * <p>分割は、空白（全角・半角）で行われます。</p>
285          *
286          * @param queriesString 検索語をスペース（全角・半角）で区切った文字列
287          * @return 検索語の配列
288          */
289         private static List<String> splitQueries(String queriesString) {
290                 assert (queriesString != null);
291
292                 // 分割結果
293                 List<String> queries = new LinkedList<String>(
294                         Arrays.asList(queriesString.split("¥¥p{javaWhitespace}+"
295 )));
296
297                 // 先端の空の結果は除く
298                 for (Iterator<String> i = queries.iterator(); i.hasNext(); ) {
299                         if (i.next().isEmpty()) {
300                                 i.remove();
301                         }
302                         else {
303                                 break;
304                         }
305                 }
306
307                 return queries;
308         }
309 }
```

¥resources

なし

¥webapp¥authentication

（1）error.jsp

error.jsp　　Page 1

```
1 <%@page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8" %>
2 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
3
4
5 <!-- 認証に失敗したことを設定して、　　認証ページへ戻る -->
6
7 <c:set var="isError" scope="session" value="true" />
8 <c:redirect url="index.jsp" />
```

（2）index.jsp

index.jsp　　Page 1

```
 1 <%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8" isELIgnor
 2 ed="false" %>
 3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
 4 <%@ include file="/stylesheets/styles.jsp" %>
 5 <%@ include file="/main_pages.jsp" %>
 6
 7 <html>
 8
 9 <head>
10   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
11   <link rel="stylesheet" href="${common_style_url}" type="text/css" />
12   <link rel="stylesheet" href="${login_style_url}" type="text/css" />
13   <title>朝鮮朝刊本刻手名データベース検索システム ログイン</title>
14 </head>
15
16 <body>
17
18   <h1>朝鮮朝刊本刻手名データベース検索システム</h1>
19   <hr />
20
21   <h2>ユーザ認証</h2>
22   <p>閲覧には認証が必要です。</p>
23
24
25   <%-- アプリケーションサーバに認証処理を任せる --%>
26   <form method="POST" action="j_security_check">
27
28     <dl id="login_form">
29       <dt>ユーザ名</dt>
30       <dd> <input type="text" name="j_username" /> </dd>
31       <dt>パスワード</dt>
32       <dd> <input type="password" name="j_password" /> </dd>
33     </dl>
34
35     <p id="single_navigation">
36       <input type="submit" value="認証" /> </span>
37     </p>
38
39   </form>
40
41   <c:if test="${isError}">
42     <p id="warning">
43                 認証に失敗しました。ユーザ名またはパスワードに間違いがあります
44 。</p>
45     <c:remove var="isError" />
46   </c:if>
47
48   <hr />
49
50   <p id="top_page_navigation">
51     <a href="${top_page_url}">トップページに戻る</a>
52   </p>
53
54 </body>
55
56 </html>
```

¥catalogue

（1）detail.jsp

detail.jsp　　Page 1

```
 1 <%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8" isELIgnor
 2 ed="false" %>
 3 <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
 4 <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
 5 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
 6 <%@ taglib uri="http://myfaces.apache.org/tomahawk" prefix="t" %>
 7 <%@ include file="/stylesheets/styles.jsp" %>
 8
 9 <html>
10
11 <f:view>
12
13 <head>
14   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
15
16   <meta http-equiv="Content-Style-Type" content="text/css" />
17   <link rel="stylesheet" href="${common_style_url}" type="text/css" />
18   <link rel="stylesheet" href="${catalogue_style_url}" type="text/css" />
19   <link rel="stylesheet" href="${detail_style_url}" type="text/css" />
20
21   <title>
22   <h:outputText value="#{KokushuDetail.kokushu.title} #{KokushuDetail.kokushu.
23 author}" />
24   </title>
25 </head>
26
27 <body>
28
29   <h1>
30     <h:outputText value="#{KokushuDetail.kokushu.title} #{KokushuDetail.kokush
31 u.author}" />
32   </h1>
33
34   <div class="detail_tab">
35
36     <table class="detail_table">
37
38       <tbody>
39
40         <tr>
41           <th>排列番號</th>
42           <td><h:outputText value="#{KokushuDetail.kokushu.id}" /></td>
43         </tr>
44
45         <tr>
46           <th>書名・巻數・册數</th>
47           <td><h:outputText value="#{KokushuDetail.kokushu.title}"
48             converter="EmptyStringConverter" /></td>
49         </tr>
50
51         <tr>
52           <th>撰者</th>
53           <td><h:outputText value="#{KokushuDetail.kokushu.author}"
54             converter="EmptyStringConverter" /></td>
55         </tr>
56
57         <tr>
58           <th>刊年</th>
59           <td><h:outputText value="#{KokushuDetail.kokushu.year}"
60             converter="EmptyStringConverter" /></td>
61         </tr>
62
```

```
63          <tr>
64            <th>刊地</th>
65            <td><h:outputText value="#{KokushuDetail.kokushu.place}"
66              converter="EmptyStringConverter" /></td>
67          </tr>
68
69          <tr>
70            <th>所蔵者</th>
71            <td><h:outputText value="#{KokushuDetail.kokushu.owner}"
72              converter="EmptyStringConverter"/></td>
73          </tr>
74
75          <tr>
76            <th>版心部刻手名</th>
77            <td><h:outputText value="#{KokushuDetail.kokushu.centerKokushuName}"
78
79              converter="EmptyStringConverter" /></td>
80          </tr>
81
82          <tr>
83            <th>刊記部刻手名</th>
84            <td><h:outputText value="#{KokushuDetail.kokushu.imprintKokushuName}
85 "
86              converter="EmptyStringConverter" /></td>
87          </tr>
88
89          <tr>
90            <th>出典</th>
91            <td><h:outputText value="#{KokushuDetail.kokushu.reference}"
92              converter="EmptyStringConverter" /></td>
93          </tr>
94
95        </tbody>
96
97      </table>
98
99    </div>
100
101   <div class="close_button">
102     <button type="button" onclick="window.close()">閉じる</button>
103   </div>
104
105
106 </body>
107
108 </f:view>
109
110 </html>
```

（2）index.jsp

index.jsp　　Page 1

```
 1 <%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8" isELIgnored
 2 ="false" %>
 3 <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
 4 <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
 5 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
 6 <%@ taglib uri="http://myfaces.apache.org/tomahawk" prefix="t" %>
 7 <%@ include file="/stylesheets/styles.jsp" %>
 8
 9 <html>
10
11 <head>
12   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
13
14   <meta http-equiv="Content-Style-Type" content="text/css" />
15   <link rel="stylesheet" href="${common_style_url}" type="text/css" />
16   <link rel="stylesheet" href="${catalogue_style_url}" type="text/css" />
17
18   <title>朝鮮朝刊本刻手名一覧</title>
19 </head>
20
21 <body>
22
23 <f:view>
24
25
26   <h1>朝鮮朝刊本刻手名データベース検索システム</h1>
27
28   <div id="search_form_panel">
29
30     <h:form id="search_form">
31
32       <h:inputText id="keyword" styleClass="keyword_text_field"
33         value="#{KokushuCatalogue.keyword}"
34         validator="#{KokushuCatalogue.validateKeyword}" />
35       <h:commandButton id="search_submit" value="検索"
36         action="#{KokushuCatalogue.search}" />
37       <h:message for="keyword" />
38       <h:selectOneRadio value="#{KokushuCatalogue.searchMethodIndex}">
39         <%— itemValue の値は、Query.SearchMethod
40               の各列挙のインデックス —%>
41         <f:selectItem itemValue="0" itemLabel="AND 検索" />
42         <f:selectItem itemValue="1" itemLabel="OR 検索" />
43       </h:selectOneRadio>
44
45     </h:form>
46     <div class="directions_switch"><h2>検索方法</h2></div>
47     <div class="directions">
48       <p>各検索語は、スペースを空けて入力してください。</p>
49       <p>例：「伊川 宋邵雍」</p>
50     </div>
51
52   </div>
53
54   <div id="all_form_panel">
55
56     <h:form id="all_form">
57
58       <h:commandButton id="all_submit" value="全てのデータを見る"
59         action="#{KokushuCatalogue.showAll}" />
60       <h:outputText id="total_data" styleClass="total_data"
61         value="全 #{KokushuCatalogue.total} 件" />
62
63     </h:form>
64
65   </div>
66
```

```
 67   <%-- 初めてこのページに来た場合の説明 --%>
 68   <f:subview id="initial_results" rendered="#{KokushuCatalogue.initial}">
 69     <t:htmlTag id="initial_text" styleClass="result_summary" value="p">
 70     ここに結果が表示されます。</t:htmlTag>
 71   </f:subview>
 72
 73   <%-- 検索結果が0件だった場合 --%>
 74   <f:subview id="no_results"
 75       rendered="#{!KokushuCatalogue.resultExistent && !KokushuCatalogue.initial}
 76 ">
 77     <t:htmlTag id="no_results_text" styleClass="result_summary" value="p">
 78     検索に一致する刻手データはありませんでした。</t:htmlTag>
 79   </f:subview>
 80
 81   <%-- 結果 --%>
 82   <f:subview id="results" rendered="#{KokushuCatalogue.resultExistent}">
 83
 84     <t:htmlTag value="p" styleClass="result_summary"
 85         rendered="#{KokushuCatalogue.searched}">
 86       <h:outputText value="「#{KokushuCatalogue.keyword}」"/>
 87       <h:outputText value="（#{KokushuCatalogue.searchMethod}検索）" />
 88       で検索した結果：
 89      <h:outputText value="#{KokushuCatalogue.resultCount}" />件
 90     </t:htmlTag>
 91
 92     <t:htmlTag value="p" styleClass="result_summary"
 93         rendered="#{KokushuCatalogue.all}">
 94       全ての刻手データ：<h:outputText value="#{KokushuCatalogue.resultCount}" />
 95 件
 96     </t:htmlTag>
 97
 98     <h:form target="_blank" id="catalogue_form">
 99
100       <h:dataTable id="catalogue_table" value="#{KokushuCatalogue.items}"
101           var="item" rows="#{KokushuCatalogue.pageSize}"
102           binding="#{KokushuCatalogue.selectedComponent}"
103           styleClass="catalogue_table"
104           columnClasses="catalogue_column"
105           headerClass="catalogue_header">
106
107         <h:column>
108           <f:facet name="header"><h:outputText value="書名・巻数・冊数" /></f:fa
109 cet>
110           <h:outputText value="#{item.title}"
111             converter="EmptyStringConverter" />
112          </h:column>
113
114         <h:column>
115           <f:facet name="header"><h:outputText value="撰者" /></f:facet>
116           <h:outputText value="#{item.author}"
117             converter="EmptyStringConverter"/>
118          </h:column>
119
120         <h:column>
121           <f:facet name="header"><h:outputText value="刊年" /></f:facet>
122           <h:outputText value="#{item.year}"
123             converter="EmptyStringConverter" />
124          </h:column>
125
126         <h:column>
127           <f:facet name="header"><h:outputText value="刊地" /></f:facet>
128           <h:outputText value="#{item.place}"
129             converter="EmptyStringConverter"/>
130          </h:column>
131
132         <h:column>
```

```
133           <h:commandButton value="詳細"
134             action="#{KokushuCatalogue.showDetail}" />
135           </h:column>
136
137       </h:dataTable>
138
139     </h:form>
140
141     <div class="data_scroller">
142     <h:form>
143       <t:dataScroller id="scroller" fastStep="10"
144           for=":results:catalogue_form:catalogue_table"
145           pageCountVar="pageCount" pageIndexVar="pageIndex"
146           paginator="true" paginatorMaxPages="9"
147           paginatorActiveColumnClass="catalogue_active_page"
148           paginatorColumnClass="catalogue_page"
149           paginatorTableClass="catalogue_paginator"
150           styleClass="catalogue_scroller">
151
152         <f:facet name="first"><h:outputText value="最初へ" /></f:facet>
153         <f:facet name="previous"><h:outputText value="前へ" /></f:facet>
154         <f:facet name="next"><h:outputText value="次へ" /></f:facet>
155         <f:facet name="last"><h:outputText value="最後へ" /></f:facet>
156         <f:facet name="fastforward">
157           <h:outputText value="10ページ先へ" />
158          </f:facet>
159         <f:facet name="fastrewind">
160           <h:outputText value="10ページ前へ" />
161          </f:facet>
162
163         <h:outputFormat value="{0}/{1}">
164           <f:param value="#{pageIndex}" />
165           <f:param value="#{pageCount}" />
166          </h:outputFormat>
167
168       </t:dataScroller>
169     </h:form>
170
171     </div>
172
173   </f:subview>
174
175   <hr />
176
177   <h:form>
178     <div class="navigator">
179       <h:commandButton value="トップページに戻る" action="topPage" />
180     </div>
181   </h:form>
182
183 </f:view>
184
185 </body>
186
187 </html>
```

¥stylesheets

（1）catalogue.css

catalogue.css　　Page 1

```
 1 @charset "UTF-8";
 2
 3 #search_form_panel {
 4   border-right : 1px solid #0c0c0c;
 5   width : 50%;
 6   height : 18em;
 7   float : left;
 8 }
 9
10 #search_form_panel h2 {
11   background : white; /* 本来は inherit */
12   color : black; /* 本来は inherit */
13   font-size : 1.2em;
14   margin-left : 9%;
15   margin-top : 2em;
16   padding-bottom : 0px;
17 }
18
19 #search_form_panel .directions {
20   margin-left : 13%;
21 }
22
23 .directions dt {
24   font-weight : bold;
25 }
26
27 .directions dd {
28   margin-left : 2em;
29 }
30
31 #search_form {
32   margin-top : 3em;
33   margin-left : 25%;
34 }
35
36 .keyword_text_field {
37   width : 20em;
38 }
39
40 #all_form_panel {
41   width : 45%;
42   height : 18em;
43   float : left;
44   margin : 0px;
45 }
46
47 #all_form {
48   margin-top : 3em;
49   margin-left : 40%;
50 }
51
52 .total_data {
53   display : block;
54   margin-top : 3em;
55   margin-left : 1em;
56 }
57
58 .result_summary {
59   border-top : 1px dashed blue;
60   clear : both;
61   padding-top : 3em;
62   margin-bottom : 3em;
63   text-align : center;
64 }
65
66 div.search_input_form table {
```

```
  border : solid 1px #000000;
}

div.search_input_form th {
  margin : 3px;
  padding : 5px;
  border : solid 1px gray;
}

div.search_input_form td {
  border : solid 1px gray;
}

table.catalogue_table {
  margin-left : auto;
  margin-right : auto;
  padding : 0px;
  border : solid 1px #000000;
}

th.catalogue_header {
  margin : 0px;
  padding : 5px;
  border : solid 1px gray;
  color : #ffffff;
  background-color : #9999ff;
  text-align : center;
}

td.catalogue_column {
  margin : 0px;
  padding : 5px;
  border : solid 1px gray;
  text-align : left;
}

table.catalogue_scroller {
  margin : 1px;
  border : solid 1px gray;
  margin-left : auto;
  margin-right : auto;
}

.data_scroller {
  text-align : center;
}

.catalogue_active_page {
  font-weight : bold;
}

table.catalogue_paginator {
  margin : 5px;
  padding : 5px;
  border : solid 1px gray;
}

td.catalogue_active_page {

}

td.catalogue_page {

}
```

(2) catalogue.css.orig

catalogue.css.orig Page 1

```
 1 @charset "UTF-8";
 2
 3 #search_form_panel {
 4   border-right : 1px solid #0c0c0c;
 5   width : 45%;
 6   height : 10em;
 7   float : left;
 8 }
 9
10 #search_form {
11   margin-top : 3em;
12   margin-left : 7em;
13 }
14
15 .keyword_text_field {
16   width : 20em;
17 }
18
19 #all_form_panel {
20   width : 45%;
21   height : 10em;
22   float : left;
23   margin : 0px;
24 }
25
26 #all_form {
27   margin-top : 3em;
28   margin-left : 7em;
29 }
30
31 .total_data {
32   display : block;
33   margin-left : 4em;
34 }
35
36 .result_summary {
37   border-top : 1px dashed blue;
38   clear : both;
39   padding-top : 1em;
40 }
41
42 div.search_input_form table {
43   margin : 15px;
44   border : solid 1px #000000;
45 }
46
47 div.search_input_form th {
48   margin : 3px;
49   padding : 5px;
50   border : solid 1px gray;
51 }
52
53 div.search_input_form td {
54   border : solid 1px gray;
55 }
56
57 table.catalogue_table {
58   margin : 0px;
59   padding : 0px;
60   border : solid 1px #000000;
61 }
62
```

```
 63 th.catalogue_header {
 64   margin : 0px;
 65   padding : 5px;
 66   border : solid 1px gray;
 67   color : #ffffff;
 68   background-color : #9999ff;
 69   text-align : center;
 70 }
 71
 72 td.catalogue_column {
 73   margin : 0px;
 74   padding : 5px;
 75   border : solid 1px gray;
 76   text-align : left;
 77 }
 78
 79 table.catalogue_scroller {
 80   margin : 1px;
 81   border : solid 1px gray;
 82   margin-left : auto;
 83   margin-right : auto;
 84 }
 85
 86 .data_scroller {
 87   text-align : center;
 88 }
 89
 90 .catalogue_active_page {
 91   font-weight : bold;
 92 }
 93
 94 table.catalogue_paginator {
 95   margin : 5px;
 96   padding : 5px;
 97   border : solid 1px gray;
 98 }
 99
100 td.catalogue_active_page {
101
102 }
103
104 td.catalogue_page {
105
106 }
```

（3）common.css

common.css　　Page 1

```
 1 @charset "UTF-8";
 2
 3 h1 {
 4   color : #ffffff;
 5   background : #6633ff;
 6   margin : 0px;
 7   padding : 5px
 8 }
 9
10 h2{
11   color : #ffffff;
12   background : #6633ff;
13   margin : 0px;
14   padding : 5px;
15 }
16
17 table.menu td input {
18   width : 8em;
19   text-align : center;
20 }
21
22 address {
23   text-align : right;
24   font-style : normal;
25 }
26
27 p {
28   text-indent : 1em;
29   margin : 0em;
30   padding : 0em;
31   line-height : 1.5em;
32 }
33
34 .input_form tbody td {
35   padding-top : 0.5em;
36   padding-bottom : 0.5em;
37 }
38
39 .input_area {
40   width : 40em;
41   height : 5em;
42 }
43
44 .input_select {
45   width : 40em;
46   font-size : 1em;
47 }
48
49
50 /*----------------------------------------------------------------------------
51 --
52   ナビゲーション（ページ遷移用のボタンやリンクなど）
53 ----------------------------------------------------------------------------
54 */
55
56 /* トップページへ遷移するナビゲーション */
57 #top_page_navigation {
58
59 }
60
61 /* 次のページへ遷移するナビゲーション */
62 #next_navigation {
```

```
 63   margin-left : 80%;
 64 }
 65
 66 /* 前のページへ遷移するナビゲーション */
 67 #previous_navigation {
 68   margin-right : 80%;
 69 }
 70
 71 /* 単一のナビゲーション（中央に置くナビゲーション）
 72    ブロック要素専用
 73  */
 74 #single_navigation {
 75   text-align : center;
 76 }
```

(4) detail.css

detail.css Page 1

```
 1 @charset "UTF-8";
 2
 3 h2 {
 4   margin-top : 1em;
 5   padding-left : 1em;
 6 }
 7
 8 div.detail_tab {
 9   text-align : center;
10 }
11
12 table.detail_table {
13   margin : 0px;
14   padding : 0px;
15   border : solid 1px #000000;
16   margin-left : auto;
17   margin-right : auto;
18 }
19
20 th {
21   margin : 0px;
22   padding : 5px;
23   border : solid 1px gray;
24   color : #ffffff;
25   background-color : #9999ff;
26   text-align : right;
27 }
28
29 td {
30   border : solid 1px gray;
31 }
32
33 .close_button {
34   text-align : center;
35   margin-top : 3em;
36 }
```

(5) index.jsp

index.jsp Page 1

```
 1 <%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8" isELIgnor
 2 ed="false" %>
 3 <%@ include file="/no_cache.inc" %>
 4 <%@ include file="/main_pages.jsp" %>
 5
 6 <html>
 7   <head>
 8     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
 9     <title>index</title>
10   </head>
11   <body>
12       <jsp:forward page="${top_page_url}" />
13   </body>
14 </html>
```

（6）login.css

login.css　　Page 1

```
@charset "UTF-8";

/*-------------------------------------------------------------------------------
-
  ログインフォーム
-------------------------------------------------------------------------------
*/

/* ログインフォーム全体 */
#login_form {
  text-align : center;
}

/* ログインフォーム中のラベル */
dt {
  text-align : left;
  margin-top : 1em;
  margin-left : 37%;
  font-weight : bold;
}

/* ログインフォーム中の入力フォームのあるブロック */
dd {
  margin-left : 2em;
  margin-bottom : 1em;
}

/* ログインフォーム中の入力フォーム */
dd input {
  width : 17em;
}


/*-------------------------------------------------------------------------------
--
  メッセージ関係
-------------------------------------------------------------------------------
*/

/* ログイン失敗メッセージ */
#warning {
  color : red;
  text-align : center;
}

```

(7) styles.jsp

styles.jsp Page 1

```
1 <%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8" isELIgnor
2 ed="false" %>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4
5 <%-- トップページ共通 --%>
6   <c:url var="top_style_url" value="/stylesheets/top.css" />
7
8 <%-- 全ページ共通 --%>
9   <c:url var="common_style_url" value="/stylesheets/common.css" />
10
11 <%-- 認証ページ用 --%>
12   <c:url var="login_style_url" value="/stylesheets/login.css" />
13
14 <%-- データ一覧用 --%>
15   <c:url var="catalogue_style_url" value="/stylesheets/catalogue.css" />
16
17 <%-- データ詳細表示用 --%>
18   <c:url var="detail_style_url" value="/stylesheets/detail.css" />
19
20
```

(6) top.css

top.css Page 1

```
1 @charset "UTF-8";
2
3 .contents {
4   margin-top : 3%;
5   margin-bottom : 3%;
6   margin-left : 5%;
7   margin-right : 5%;
8 }
9
10 #menu_buttons {
11   margin-top : 3%;
12   text-align : center;
13 }
14
15 .catalogue_button {
16   font-size : 2em;
17 }
```

¥WEB-INF

（1）face-config.xml

faces-config.xml　　Page 1

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <faces-config xmlns="http://java.sun.com/xml/ns/javaee"
 3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 5     http://java.sun.com/xml/ns/j2ee/web-facesconfig_1_2.xsd"
 6   version="1.2">
 7
 8 <!--
 9   <application>
10
11     <default-render-kit-id>
12       org.apache.myfaces.trinidad.core
13     </default-render-kit-id>
14
15   </application>
16
17 -->
18
19 <!--**************************************************************************
20
21         遷移ルール
22 **************************************************************************-->
23
24
25   <!-- 前ページ共通 -->
26   <navigation-rule>
27
28     <!-- トップページへの遷移 -->
29     <navigation-case>
30       <from-outcome>topPage</from-outcome>
31       <to-view-id>/index2.jsp</to-view-id>
32     </navigation-case>
33
34     <!-- 一覧への遷移 -->
35     <navigation-case>
36       <from-outcome>catalogue</from-outcome>
37       <to-view-id>/catalogue/index.jsp</to-view-id>
38
39       <!-- 認証ページへ進ませるため、リダイレクトが必要 -->
40       <redirect />
41     </navigation-case>
42
43   </navigation-rule>
44
45   <!-- 一覧からの遷移 -->
46   <navigation-rule>
47
48     <from-view-id>/catalogue/*</from-view-id>
49
50     <!-- 詳細ページへの遷移 -->
51     <navigation-case>
52       <from-outcome>detail</from-outcome>
53       <to-view-id>/catalogue/detail.jsp</to-view-id>
54     </navigation-case>
55
56   </navigation-rule>
57
58 <!--**************************************************************************
59
60         Managed Bean
61 **************************************************************************-->
62
```

```
63
64   <!-- 刻手データの一覧ページ -->
65   <managed-bean>
66     <managed-bean-name>KokushuCatalogue</managed-bean-name>
67     <managed-bean-class>jp.ac.u_toyama.itc.kokushu.webapp.catalogue.KokushuCat
68 aloguePage</managed-bean-class>
69     <managed-bean-scope>session</managed-bean-scope>
70   </managed-bean>
71
72   <!-- 刻手データの詳細ページ -->
73   <managed-bean>
74
75     <managed-bean-name>KokushuDetail</managed-bean-name>
76     <managed-bean-class>jp.ac.u_toyama.itc.kokushu.webapp.catalogue.KokushuDet
77 ailPage</managed-bean-class>
78     <managed-bean-scope>request</managed-bean-scope>
79
80     <managed-property>
81       <property-name>kokushuId</property-name>
82       <value>#{pageFlowScope.detailedKokushuId}</value>
83     </managed-property>
84
85   </managed-bean>
86
87 <!--**************************************************************************
88
89         Converter
90 ***************************************************************************-->
91
92
93   <converter>
94     <converter-id>EmptyStringConverter</converter-id>
95     <converter-class>jp.ac.u_toyama.itc.kokushu.webapp.converter.EmptyStringCo
96 nverter</converter-class>
97   </converter>
98
99 </faces-config>
```

(2) sun-web.xml

sun-web.xml Page 1

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <!DOCTYPE sun-web-app PUBLIC
 3   "-//Sun Microsystems, Inc.//DTD Application Server 9.0 Servlet 2.5//EN"
 4   "http://www.sun.com/software/appserver/dtds/sun-web-app_2_5-0.dtd">
 5
 6 <sun-web-app>
 7
 8     <security-role-mapping>
 9         <role-name>user</role-name>
10         <group-name>user</group-name>
11     </security-role-mapping>
12
13 </sun-web-app>
```

(3) web.xml

web.xml Page 1

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <web-app xmlns="http://java.sun.com/xml/ns/javaee"
 3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 5       http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 6     version="2.5">
 7
 8
 9   <context-param>
10     <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
11     <param-value>client</param-value>
12   </context-param>
13
14   <env-entry>
15     <env-entry-name>com.sun.faces.ClientStateSavingPassword</env-entry-name>
16     <env-entry-type>java.lang.String</env-entry-type>
17     <env-entry-value>8KFphKribCDU59rTRAbAGPKXTxdg9Y</env-entry-value>
18   </env-entry>
19
20
21   <!-- Faces Servlet -->
22   <servlet>
23     <servlet-name>FacesServlet</servlet-name>
24     <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
25     <load-on-startup>1</load-on-startup>
26   </servlet>
27
28
29   <!-- Apache MyFaces Trinidad -->
30   <servlet>
31     <servlet-name>resources</servlet-name>
32     <servlet-class>org.apache.myfaces.trinidad.webapp.ResourceServlet</servlet
33 -class>
34   </servlet>
35
36   <!-- This cannot be configured currently -->
37   <servlet-mapping>
38     <servlet-name>resources</servlet-name>
39     <url-pattern>/adf/*</url-pattern>
40   </servlet-mapping>
41
42   <filter>
43     <filter-name>trinidad</filter-name>
44     <filter-class>org.apache.myfaces.trinidad.webapp.TrinidadFilter</filter-cl
45 ass>
46   </filter>
47
48   <filter-mapping>
49     <filter-name>trinidad</filter-name>
50     <!-- This assumes that the FacesServlet has been registered -->
51     <servlet-name>FacesServlet</servlet-name>
52   </filter-mapping>
53
54
55   <!-- virtual path mapping -->
56
57   <servlet-mapping>
58     <servlet-name>FacesServlet</servlet-name>
59     <url-pattern>/pages/*</url-pattern>
60   </servlet-mapping>
61
62
```

```
63  <!-- Welcome files -->
64
65  <welcome-file-list>
66    <welcome-file>index.jsp</welcome-file>
67    <welcome-file>index.html</welcome-file>
68  </welcome-file-list>
69
70  <!-- セキュリティ設定 -->
71
72  <security-role>
73    <role-name>user</role-name>
74  </security-role>
75
76  <security-constraint>
77    <display-name>Security Constraint</display-name>
78
79    <web-resource-collection>
80      <web-resource-name>Catalogue Page</web-resource-name>
81      <url-pattern>/pages/catalogue/*</url-pattern>
82      <http-method>GET</http-method>
83      <http-method>POST</http-method>
84      <http-method>HEAD</http-method>
85      <http-method>PUT</http-method>
86      <http-method>OPTIONS</http-method>
87      <http-method>TRACE</http-method>
88      <http-method>DELETE</http-method>
89    </web-resource-collection>
90
91    <auth-constraint>
92      <role-name>user</role-name>
93    </auth-constraint>
94
95    <user-data-constraint>
96      <transport-guarantee>CONFIDENTIAL</transport-guarantee>
97    </user-data-constraint>
98
99  </security-constraint>
100
101 <login-config>
102   <auth-method>FORM</auth-method>
103
104   <realm-name>kokushu-realm</realm-name>
105
106   <form-login-config>
107     <form-login-page>/authentication/index.jsp</form-login-page>
108     <form-error-page>/authentication/error.jsp</form-error-page>
109   </form-login-config>
110
111 </login-config>
112
113 </web-app>
```

src¥main¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥webapp

（1）index2.jsp

index2.jsp　　Page 1

```
 1 <%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8" isELIgnor
 2 ed="false" %>
 3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
 4 <%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
 5 <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
 6 <%@ include file="/stylesheets/styles.jsp" %>
 7 <%@ include file="/main_pages.jsp" %>
 8
 9
10
11 <html>
12 <head>
13   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
14
15   <meta http-equiv="Content-Style-Type" content="text/css" />
16   <link rel="stylesheet" href="${common_style_url}" type="text/css" />
17   <link rel="stylesheet" href="${top_style_url}" type="text/css" />
18
19   <title>朝鮮朝刊本刻手名データベース検索システム</title>
20 </head>
21
22 <body>
23
24 <f:view>
25
26   <h1>朝鮮朝刊本刻手名データベース検索システム</h1>
27
28   <div class="contents">
29     <p>本データベースは、元富山大学人文学部教授の藤本幸夫先生が、
30     数十年にわたって「日本現存朝鮮古刊本の調査とその語学的・書誌学的研究」をさ
31 れ、
32     平成19年3月に出版された「朝鮮朝刊本刻手名集（第二版）」に掲載されたマスタ
33 ーデータを
34     データベース化したものである。</p>
35     <p>藤本教授曰く「刻手名を有する書籍は必ずしも多くはないが、刻手名によって
36 、
37     ある書籍の刊年と刊地を決定し得ることがある。どの地方で、どのような書籍が刊
38 行されたかを
39     しることは、出版文化を考える場合、極めて重要である」と。</p>
40     <p>筆者らが進める、刻手名と印影・写真画像による日本現存朝鮮古書の画像デー
41 タベースとの
42     連動が実現すれば、画像と刻手名による古書の同定が進み、世界の各地に居ながら
43 古文書の
44     発掘、同定及び内容に関する研究ができるようになる。</p>
45     <p>刻手名データベースは、その画像データベースとともに基盤となるデータベー
46 スであり、
47     今後、印影の画像が加われば、古書の原文画像データベースとともに、
48     古書研究に不可欠のものとなる。</p>
49
50     <div id="menu_buttons">
51       <h:form>
52         <h:commandButton styleClass="catalogue_button" value = "閲覧" action="
53 catalogue" />
54       </h:form>
55     </div>
56   </div>
57
58   <h2>更新履歴</h2>
59   <ul>
60     <li>2008/03/31 第1版作成</li>
61   </ul>
62   <hr />
```

index2.jsp　　Page 2

```
 63   <address>
 64   運営：<a href="http://www.itc.u-toyama.ac.jp/DOKB/">日本現存朝鮮古書データベ
 65 ース作成チーム</a>
 66   </address>
 67
 68 </f:view>
 69
 70 </body>
 71 </html>
 72
```

（2）index.jsp

index.jsp　　Page 1

```
  1 <%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8" isELIgnor
  2 ed="false" %>
  3 <%@ include file="/main_pages.jsp" %>
  4
  5 <html>
  6   <head>
  7     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  8     <title>index</title>
  9   </head>
 10   <body>
 11       <jsp:forward page="/${jsf_mapping}/index2.jsp" />
 12   </body>
 13 </html>
```

（3）main_page.jsp

main_pages.jsp　　Page 1

```
  1 <%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8" isELIgnor
  2 ed="false" %>
  3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
  4
  5 <%-- JSF が動作するサーブレットのパス --%>
  6   <c:set var="jsf_mapping" value="pages" />
  7
  8 <%-- トップページ --%>
  9   <c:url var="top_page_url" value="/index.jsp" />
 10
 11   <c:url var="faces_page_url" value="/${jsf_mapping}/index2.jsp" />
 12
```

src¥test¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥webapp¥search

（1）QueryTest.java

QueryTest.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.webapp.search;
 2
 3 import junit.framework.Assert;
 4
 5 import org.junit.Test;
 6
 7
 8 /**
 9  * <p>検索文の生成をテストします。</p>
10  *
11  * @author Keita Kita (2008/03/13 -)
12  * @since 1.0-SNAPSHOT
13  */
14 public class QueryTest {
15         /**
16          * <p>検索語の構文の確認に、null を渡した場合をテストします。</p>
17          */
18         public void testExistenceMethodWithNull() {
19                 Assert.assertFalse(Query.isRight(null));
20         }
21
22         /**
23          * <p>検索語の構文の確認で、不正な場合をテストします。</p>
24          */
25         @Test
26         public void testValidatingIsNot() {
27                 // 空の文字列
28                 Assert.assertFalse(Query.isRight(""));
29                 // 半角スペースのみ
30                 Assert.assertFalse(Query.isRight(" "));
31                 Assert.assertFalse(Query.isRight("  "));
32                 // 全角スペースのみ
33                 Assert.assertFalse(Query.isRight("　"));
34                 Assert.assertFalse(Query.isRight("　　"));
35                 // 半角・全角混合
36                 Assert.assertFalse(Query.isRight(" 　"));
37                 // 制御文字・改行文字
38                 Assert.assertFalse(Query.isRight("abな な¥u007Fde¥nekde"));
39         }
40
41         /**
42          * <p>検索語の構文の確認で、検索語がある場合をテストします。</p>
43          */
44         @Test
45         public void testValidating() {
46                 // 半角スペースで区切る
47                 Assert.assertTrue(Query.isRight("a a"));
48                 Assert.assertTrue(Query.isRight("a "));
49                 Assert.assertTrue(Query.isRight(" a"));
50                 // 全角スペースで区切る
51                 Assert.assertTrue(Query.isRight("a　a"));
52                 Assert.assertTrue(Query.isRight("a　"));
53                 Assert.assertTrue(Query.isRight("　a"));
54         }
55
56         /**
57          * <p>検索語の生成で、null を渡した場合をテストします。</p>
58          */
59         @Test(expected=IllegalArgumentException.class)
60         public void testCreatingWithNull() {
61                 Query.createQuery(null, Query.Method.AND);
62         }
63
64         /**
65          * <p>検索語の生成で、null を渡した場合をテストします。</p>
66          */
```

```
67          @Test(expected=IllegalArgumentException.class)
68          public void testCreatingWithNull2() {
69                  Query.createQuery("abcde", null);
70          }
71
72          /**
73           * <p>検索語の生成で、空の検索文を渡した場合をテストします。</p>
74           */
75          @Test(expected=IllegalArgumentException.class)
76          public void testCreatingWithEmpty() {
77                  Query.createQuery("", Query.Method.AND);
78          }
79
80          /**
81           * <p>検索語の生成で、空白のみの検索文を渡した場合をテストします。</p>
82           */
83          @Test(expected=IllegalArgumentException.class)
84          public void testCreatingWithSpacesOnly() {
85                  Query.createQuery(" ", Query.Method.AND);
86          }
87
88          /**
89           * <p>検索語の生成で、制御文字や改行文字を渡した場合をテストします。</p>
90
91           */
92          @Test(expected=IllegalArgumentException.class)
93          public void testCreatingWithControlAndNewLine() {
94                  Query.createQuery("ああ¥nおお¥u0092bcd", Query.Method.AND);
95          }
96
97          /**
98           * <p>検索語の生成をテストします。</p>
99           */
100         @Test
101         public void testCreating() {
102                 // 単一の検索語
103                 Assert.assertEquals(
104                         "abc", Query.createQuery("abc", Query.Method.AND));
105
106                 // 複数の検索語
107                 Assert.assertEquals(
108                         "abc AND def", Query.createQuery("abc    def", Query.Met
109 hod.AND));
110                 Assert.assertEquals(
111                         "abc", Query.createQuery("   abc", Query.Method.OR));
112                 Assert.assertEquals(
113                         "abc", Query.createQuery("abc   ", Query.Method.AND));
114                 Assert.assertEquals(
115                         "abc OR def", Query.createQuery("  abc    def   ", Query
116 .Method.OR));
117
118                 // AND など、演算子を含めた検索語
119                 Assert.assertEquals(
120                         "abc AND ¥"AND¥" AND def",
121                         Query.createQuery("abc AND def", Query.Method.AND));
122                 Assert.assertEquals(
123                         "abc AND ¥"OR¥" AND def",
124                         Query.createQuery("abc OR def", Query.Method.AND));
125                 Assert.assertEquals(
126                         "abc OR ¥¥+ OR def",
127                         Query.createQuery("abc + def", Query.Method.OR));
128         }
129 }
```

５）kokushu-webapp-user-db：刻手データを検索するウェブアプリケーションの登録ユーザを表すモジュール

```
kokushu-webapp-user-db
    pom.xml

    src¥main
      ¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥webapp¥user
        Group.java
        User.java

    src¥test
      ¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥webapp¥user
        GroupTest.java
        UserTest.java

      ¥resources¥META-INF
        persistence.xml
```

kokushu-webapp-user-db

（1）pom.xml

pom.xml　　Page 1

```
 1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/
 2 2001/XMLSchema-instance"
 3         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apach
 4 e.org/maven-v4_0_0.xsd">
 5   <parent>
 6     <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
 7     <artifactId>kokushu</artifactId>
 8     <version>1.0-SNAPSHOT</version>
 9     <relativePath>../kokushu/pom.xml</relativePath>
10   </parent>
11
12   <modelVersion>4.0.0</modelVersion>
13   <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
14   <artifactId>kokushu-webapp-user-db</artifactId>
15   <packaging>jar</packaging>
16   <version>1.0-SNAPSHOT</version>
17   <name>kokushu-webapp-user-db</name>
18   <url>http://maven.apache.org</url>
19
20   <repositories>
21
22     <repository>
23       <id>repository.jboss.org</id>
24           <name>JBoss Maven Repository</name>
25       <url>http://repository.jboss.org/maven2</url>
26       <layout>default</layout>
27     </repository>
28
29   </repositories>
30
31   <dependencies>
32
33     <dependency>
34       <groupId>org.hibernate</groupId>
35       <artifactId>hibernate</artifactId>
36       <version>3.2.5.ga</version>
37     </dependency>
38
39     <dependency>
40       <groupId>org.hibernate</groupId>
41       <artifactId>hibernate-annotations</artifactId>
42       <version>3.3.0.ga</version>
43     </dependency>
44
45     <dependency>
46       <groupId>org.hibernate</groupId>
47       <artifactId>hibernate-entitymanager</artifactId>
48       <version>3.3.1.ga</version>
49     </dependency>
50
51     <dependency>
52       <groupId>commons-lang</groupId>
53       <artifactId>commons-lang</artifactId>
54       <version>2.3</version>
55     </dependency>
56
57     <dependency>
58       <groupId>commons-codec</groupId>
59       <artifactId>commons-codec</artifactId>
60       <version>1.3</version>
61     </dependency>
62
63     <dependency>
64       <groupId>junit</groupId>
```

```
65        <artifactId>junit</artifactId>
66        <version>4.4</version>
67        <scope>test</scope>
68      </dependency>
69
70      <!-- Hibernate EntityManager 用 -->
71      <dependency>
72        <groupId>concurrent</groupId>
73        <artifactId>concurrent</artifactId>
74        <version>1.3.4</version>
75      </dependency>
76
77      <!-- Hibernate 用 -->
78      <dependency>
79        <groupId>jboss</groupId>
80        <artifactId>jboss-cache</artifactId>
81        <version>1.2.2</version>
82      </dependency>
83
84      <dependency>
85        <groupId>jaxen</groupId>
86        <artifactId>jaxen</artifactId>
87        <version>1.1.1</version>
88      </dependency>
89
90    </dependencies>
91
92 </project>
93
```

src¥main¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥webapp¥user

（1）Group.java

Group.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.webapp.user;
 2
 3 import java.io.Serializable;
 4 import java.util.HashSet;
 5 import java.util.Set;
 6
 7 import javax.persistence.Column;
 8 import javax.persistence.Entity;
 9 import javax.persistence.GeneratedValue;
10 import javax.persistence.GenerationType;
11 import javax.persistence.Id;
12 import javax.persistence.ManyToMany;
13 import javax.persistence.Table;
14
15 import org.apache.commons.lang.Validate;
16 import org.apache.commons.lang.builder.EqualsBuilder;
17 import org.apache.commons.lang.builder.HashCodeBuilder;
18
19 /**
20  * <p>ウェブアプリケーションのユーザが所属するグループを表します。</p>
21  * <p>このクラスはスレッドセーフではありません。</p>
22  *
23  * @author Keita Kita (2008/04/10 -)
24  * @since 1.0-SNAPSHOT
25  */
26 @Entity
27 @Table(name="user_group")
28 public class Group implements Serializable {
29         /**
30          * <p>直列化の際に参照される識別番号です。</p>
31          */
32         private static final long serialVersionUID = -7626583417642461843L;
33
34         /**
35          * <p>グループに所属するユーザです。</p>
36          */
37         private Set<User> users = new HashSet<User>();
38
39         /**
40          * <p>ID です。</p>
41          */
42         private Integer id;
43
44         /**
45          * <p>グループ名です。</p>
46          */
47         private String name = "";
48
49
50         /**
51          * <p>引数なしのコンストラクタです。</p>
52          */
53         public Group() {
54                 // 処理なし
55         }
56
57         /**
58          * <p>このグループに所属するユーザを返します。</p>
59          *
60          * @return このグループに所属するユーザ
61          */
62         @ManyToMany(mappedBy="groups")
63         public Set<User> getUsers() {
64                 return this.users;
```

```
65          }
66
67          /**
68           * <p>このグループに所属するユーザを設定します。</p>
69           *
70           * @param users このグループに所属するユーザ
71           */
72          public void setUsers(Set<User> users) {
73                  this.users = users;
74          }
75
76          /**
77           * <p>ID を返します。</p>
78           *
79           * @return ID
80           */
81          @Id
82          @Column(name="group_id")
83          @GeneratedValue(strategy=GenerationType.IDENTITY)
84          public Integer getId() {
85                  return this.id;
86          }
87
88          /**
89           * <p>ID を設定します。</p>
90           *
91           * @param id 設定する ID
92           */
93          protected void setId(Integer id) {
94                  this.id = id;
95          }
96
97          /**
98           * <p>グループ名を返します。</p>
99           *
100          * @return グループ名
101          */
102         @Column(name="group_name", nullable=false, unique=true)
103         public String getName() {
104                 return this.name;
105         }
106
107         /**
108          * <p>グループ名を設定します。</p>
109          *
110          * @param name グループ名
111          * @throws IllegalArgumentException 引数が null の場合
112          */
113         public void setName(String name) {
114                 Validate.notNull(name);
115
116                 this.name = name;
117         }
118
119         @Override
120         public boolean equals(Object object) {
121                 if (this == object) {
122                         return true;
123                 }
124
125                 if (!(object instanceof Group)) {
126                         return false;
127                 }
128
```

```
129                 Group another = (Group) object;
130
131                 // グループ名が等しければ同じオブジェクトと見なす
132
133                 return (new EqualsBuilder()).
134                         append(this.name, another.getName()).isEquals();
135         }
136
137         @Override
138         public int hashCode() {
139                 return (new HashCodeBuilder()).append(this.name).toHashCode();
140         }
141 }
```

（2）User.java

User.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.webapp.user;
 2
 3 import java.io.Serializable;
 4 import java.security.MessageDigest;
 5 import java.security.NoSuchAlgorithmException;
 6 import java.util.HashSet;
 7 import java.util.Set;
 8
 9 import javax.persistence.Column;
10 import javax.persistence.Entity;
11 import javax.persistence.GeneratedValue;
12 import javax.persistence.GenerationType;
13 import javax.persistence.Id;
14 import javax.persistence.JoinColumn;
15 import javax.persistence.JoinTable;
16 import javax.persistence.ManyToMany;
17 import javax.persistence.Table;
18
19 import org.apache.commons.codec.binary.Hex;
20 import org.apache.commons.lang.Validate;
21 import org.apache.commons.lang.builder.EqualsBuilder;
22 import org.apache.commons.lang.builder.HashCodeBuilder;
23
24 /**
25  * <p>ウェブアプリケーションのユーザを表します。</p>
26  * <p>このクラスはスレッドセーフではありません。</p>
27  *
28  * @author Keita Kita (2008/04/10 -)
29  * @since 1.0-SNAPSHOT
30  */
31 @Entity
32 @Table(name="webapp_user")
33 public class User implements Serializable {
34         /**
35          * <p>直列化の際に参照される識別番号です。</p>
36          */
37         private static final long serialVersionUID = -7654988111479489266L;
38
39
40         /**
41          * <p>パスワードからハッシュを生成する際の、メッセージダイジェストの
42          * アルゴリズム名です。</p>
43          */
44         private static final String MESSAGE_DIGEST_NAME = "SHA-512";
45
46
47         /**
48          * <p>ID です。</p>
49          */
50         private Integer id;
51
52         /**
53          * <p>ユーザ名です。</p>
54          */
55         private String name = "";
56
57         /**
58          * <p>ユーザが所属しているグループです。</p>
59          */
60         private Set<Group> groups = new HashSet<Group>();
61
62         /**
63          * <p>パスワードです。</p>
64          * <p>16進表現のハッシュで格納されます。</p>
```

```
 65          */
 66         private String password;
 67
 68
 69         /**
 70          * <p>引数なしのコンストラクタです。</p>
 71          */
 72         public User() {
 73                 // 処理なし
 74         }
 75
 76         /**
 77          * <p>ID を返します。</p>
 78          *
 79          * @return ID
 80          */
 81         @Id
 82         @Column(name="user_id", nullable=false, unique=true)
 83         @GeneratedValue(strategy=GenerationType.IDENTITY)
 84         public Integer getId() {
 85                 return this.id;
 86         }
 87
 88         /**
 89          * <p>ID を設定します。</p>
 90          *
 91          * @param id ID
 92          * @throws IllegalArgumentException 引数が null の場合
 93          */
 94         protected void setId(Integer id) {
 95                 Validate.notNull(id);
 96
 97                 this.id = id;
 98         }
 99
100         /**
101          * <p>ユーザ名を返します。</p>
102          *
103          * @return ユーザ名
104          */
105         @Column(name="user_name", nullable=false, unique=true)
106         public String getName() {
107                 return this.name;
108         }
109
110         /**
111          * <p>ユーザ名を設定します。</p>
112          *
113          * @param name ユーザ名
114          * @throws IllegalArgumentException 引数が null の場合
115          */
116         public void setName(String name) {
117                 Validate.notNull(name);
118
119                 this.name = name;
120         }
121
122         /**
123          * <p>ユーザが所属しているグループを返します。</p>
124          *
125          * @return ユーザが所属しているグループ
126          */
127         @ManyToMany
128         @JoinTable(
```

```
129                 name="user_group_map",
130                 joinColumns=
131                         @JoinColumn(name="user_id", referencedColumnName="user_i
132 d"),
133                 inverseJoinColumns=
134                         @JoinColumn(name="group_id", referencedColumnName="group
135 _id")
136         )
137         public Set<Group> getGroups() {
138                 return this.groups;
139         }
140
141         /**
142          * <p>ユーザが所属しているグループを設定します。</p>
143          *
144          * @param groups ユーザが所属しているグループのセット
145          */
146         public void setGroups(Set<Group> groups) {
147                 this.groups = groups;
148         }
149
150         /**
151          * <p>ハッシュに変換された16進表記のパスワードを返します。</p>
152          *
153          * @return ハッシュに変換された16進表記のパスワード
154          */
155         @Column(name="password")
156         public String getPassword() {
157                 return this.password;
158         }
159
160         /**
161          * <p>ハッシュに変換された16進表記のパスワードを設定します。</p>
162          *
163          * @param password ハッシュに変換された16進表記のパスワード
164          * @throws IllegalArgumentException 引数が null の場合
165          */
166         protected void setPassword(String password) {
167                 Validate.notNull(password);
168
169                 this.password = password;
170         }
171
172         /**
173          * <p>パスワードを設定します。</p>
174          * <p>設定されたパスワードは、16進表記のハッシュに変換されます。</p>
175          *
176          * @param password パスワード
177          * @throws IllegalArgumentException 引数が null の場合、
178          * パスワードが空の文字列の場合
179          * @throws IllegalStateException ハッシュに変換するアルゴリズムを
180          *       使用できない場合
181          */
182         public void setPasswordString(String password) {
183                 Validate.notNull(password);
184                 Validate.isTrue(0 < password.length());
185
186                 // 使用するメッセージダイジェストアルゴリズム
187                 MessageDigest digest = getDigest();
188
189                 this.password = new String(
190                         Hex.encodeHex(digest.digest(password.getBytes())));
191         }
192
```

```
193         /**
194          * <p>パスワードからハッシュを生成するメッセージダイジェストアルゴリズム
195 を
196          * 返します。</p>
197          *
198          * @return メッセージダイジェストアルゴリズム
199          * @throws IllegalStateException アルゴリズムが使用できない状況の場合
200          */
201         private static MessageDigest getDigest() {
202                 try {
203                         return MessageDigest.getInstance(MESSAGE_DIGEST_NAME);
204                 }
205                 catch (NoSuchAlgorithmException e) {
206                         throw new IllegalStateException(e);
207                 }
208         }
209
210         @Override
211         public boolean equals(Object object) {
212                 if (this == object) {
213                         return true;
214                 }
215                 if (!(object instanceof User)) {
216                         return false;
217                 }
218
219                 User another = (User) object;
220
221                 // ユーザ名とパスワードが一致すれば、同じオブジェクトと見なす
222                 return (new EqualsBuilder()).append(this.name, another.getName()
223 ).
224                         append(this.password, another.getPassword()).
225                         isEquals();
226         }
227
228         @Override
229         public int hashCode() {
230                 return (new HashCodeBuilder()).
231                         append(this.name).
232                         append(this.password).
233                         toHashCode();
234         }
235 }
```

src¥test¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥webapp¥user

（1）GroupTest.java

GroupTest.java　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.webapp.user;
 2
 3 import javax.persistence.EntityManager;
 4 import javax.persistence.EntityManagerFactory;
 5 import javax.persistence.Persistence;
 6 import javax.persistence.Query;
 7
 8 import junit.framework.Assert;
 9
10 import org.junit.After;
11 import org.junit.AfterClass;
12 import org.junit.Before;
13 import org.junit.BeforeClass;
14 import org.junit.Test;
15
16
17 /**
18  * <p>Group のテストを行います。</p>
19  *
20  * @author Keita Kita (2008/04/10 -)
21  * @since 1.0-SNAPSHOT
22  */
23 public class GroupTest {
24         /**
25          * <p>EntityManagerFactory です。</p>
26          */
27         private static EntityManagerFactory factory;
28
29         /**
30          * <p>EntityManager です。</p>
31          */
32         private EntityManager manager;
33
34
35         /**
36          * <p>全テストメソッドの前に呼び出されます。</p>
37          */
38         @BeforeClass
39         public static void setUpAll() {
40                 factory = Persistence.createEntityManagerFactory("testManager");
41
42         }
43
44         /**
45          * <p>全テストメソッドの終了後に呼び出されます。</p>
46          */
47         @AfterClass
48         public static void tearDownAll() {
49                 factory.close();
50         }
51
52         /**
53          * <p>各テストメソッドの前に呼び出されます。</p>
54          */
55         @Before
56         public void setUp() {
57                 this.manager = factory.createEntityManager();
58         }
59
60         /**
61          * <p>各テストメソッドの後に呼び出されます。</p>
62          */
63         @After
64         public void tearDown() {
```

```
65                  this.manager.close();
66          }
67
68          /**
69           * <p>Group の取り出しをテストします。</p>
70           */
71          @Test
72          public void testReading() {
73                  Query query = this.manager.createQuery("SELECT g FROM Group AS g
74  ");
75
76                  Assert.assertNotNull(query.getResultList());
77          }
78
79          /**
80           * <p>Group の格納をテストします。</p>
81           * <p>最後にはロールバックを行います。</p>
82           */
83          @Test
84          public void testCreating() {
85                  Group group = new Group();
86
87                  group.setName(GroupTest.class.getCanonicalName());
88
89                  Query query = this.manager.createQuery(
90                          "SELECT g FROM Group As g WHERE name = :name");
91                  query.setParameter("name", GroupTest.class.getCanonicalName());
92
93
94                  this.manager.getTransaction().begin();
95
96                  Assert.assertEquals(0, query.getResultList().size());
97
98                  this.manager.persist(group);
99
100                 int resultSize = query.getResultList().size();
101
102                 this.manager.getTransaction().rollback();
103
104                 Assert.assertEquals(resultSize, 1);
105         }
106
107         /**
108          * <p>グループにユーザを追加するテストを行います。</p>
109          * <p>最後にはロールバックを行います。</p>
110          */
111         @Test
112         public void testAddingUser() {
113                 Group group = new Group();
114                 group.setName(GroupTest.class.getCanonicalName());
115
116                 User user = new User();
117                 user.setName(User.class.getCanonicalName());
118                 user.setPasswordString(User.class.getCanonicalName());
119
120                 group.getUsers().add(user);
121                 user.getGroups().add(group);
122
123                 Query userQuery = this.manager.createQuery(
124                         "SELECT u FROM User AS u WHERE name = :name");
125                 userQuery.setParameter("name", User.class.getCanonicalName());
126
127                 Query groupQuery = this.manager.createQuery(
128                         "SELECT g FROM Group AS g WHERE name = :name");
```

```
129                 groupQuery.setParameter("name", GroupTest.class.getCanonicalName
130 ());
131
132                 this.manager.getTransaction().begin();
133
134                 this.manager.persist(user);
135                 this.manager.persist(group);
136
137                 User persistedUser = (User) userQuery.getSingleResult();
138                 int groupCount = persistedUser.getGroups().size();
139                 Group persistedGroup = (Group) groupQuery.getSingleResult();
140                 int userCount = persistedGroup.getUsers().size();
141
142                 this.manager.getTransaction().rollback();
143
144                 Assert.assertEquals(1, groupCount);
145                 Assert.assertEquals(1, userCount);
146         }
147
148         /**
149          * <p>Group の等価性のテストを行います。</p>
150          * <p>グループ名が等しければ、同じオブジェクトと見なされるはずです。</p>
151
152          */
153         @Test
154         public void testEquals() {
155                 Group group = new Group();
156                 group.setName(GroupTest.class.getCanonicalName());
157
158                 Group another = new Group();
159                 another.setName(GroupTest.class.getCanonicalName());
160
161                 Assert.assertEquals(group, another);
162
163                 this.manager.getTransaction().begin();
164
165                 this.manager.persist(group);
166
167                 Query query = this.manager.createQuery(
168                         "SELECT g FROM Group As g WHERE name = :name");
169
170                 query.setParameter("name", GroupTest.class.getCanonicalName());
171
172                 group = (Group) query.getResultList().get(0);
173
174
175                 this.manager.getTransaction().rollback();
176
177                 Assert.assertEquals(group, another);
178                 Assert.assertEquals(group.hashCode(), another.hashCode());
179         }
180 }
```

（2）UserTest.java

UserTest.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.webapp.user;
 2
 3 import java.util.regex.Matcher;
 4 import java.util.regex.Pattern;
 5
 6 import javax.persistence.EntityManager;
 7 import javax.persistence.EntityManagerFactory;
 8 import javax.persistence.Persistence;
 9 import javax.persistence.Query;
10
11 import junit.framework.Assert;
12
13 import org.junit.After;
14 import org.junit.AfterClass;
15 import org.junit.Before;
16 import org.junit.BeforeClass;
17 import org.junit.Test;
18
19
20 /**
21  * <p>User のテストを行います。</p>
22  *
23  * @author Keita Kita (2008/04/11 -)
24  * @since 1.0-SNAPSHOT
25  */
26 public class UserTest {
27         /**
28          * <p>EntityManagerFactory です。</p>
29          */
30         private static EntityManagerFactory factory;
31
32         /**
33          * <p>EntityManager です。</p>
34          */
35         private EntityManager manager;
36
37         /**
38          * <p>パスワードのハッシュを表す正規表現です。</p>
39          */
40         private static final Pattern PASSWORD_HASH_PATTERN =
41                 Pattern.compile("[0-9a-fA-F]+");
42
43
44         /**
45          * <p>全テストメソッド実行の前に呼び出されます。</p>
46          */
47         @BeforeClass
48         public static void setUpAll() {
49                 factory = Persistence.createEntityManagerFactory("testManager");
50
51         }
52
53         /**
54          * <p>全テストメソッド実行の後に呼び出されます。</p>
55          */
56         @AfterClass
57         public static void tearDownAll() {
58                 factory.close();
59         }
60
61         /**
62          * <p>各テストメソッドの前に呼び出されます。</p>
63          */
64         @Before
```

```
65          public void setUp() {
66                  this.manager = factory.createEntityManager();
67          }
68
69          /**
70           * <p>各テストメソッドの後に呼び出されます。</p>
71           */
72          @After
73          public void tearDown() {
74                  this.manager.close();
75          }
76
77
78          /**
79           * <p>読み出しのテストを行います。</p>
80           */
81          @Test
82          public void testReading() {
83                  Query query = this.manager.createQuery("SELECT u FROM User As u"
84  );
85
86                  Assert.assertNotNull(query.getResultList());
87          }
88
89          /**
90           * <p>書き込みのテストを行います。</p>
91           * <p>テスト終了後にロールバックを行います。</p>
92           */
93          @Test
94          public void testCreating() {
95                  User user = new User();
96
97                  user.setName(UserTest.class.getCanonicalName());
98                  user.setPasswordString(UserTest.class.getCanonicalName());
99
100                 Query query = this.manager.createQuery(
101                         "SELECT u FROM User AS u WHERE name = :name");
102                 query.setParameter("name", UserTest.class.getCanonicalName());
103
104
105                 this.manager.getTransaction().begin();
106
107                 Assert.assertEquals(0, query.getResultList().size());
108
109                 this.manager.persist(user);
110
111                 int resultCount = query.getResultList().size();
112
113                 this.manager.getTransaction().rollback();
114
115                 Assert.assertEquals(1, resultCount);
116         }
117
118         /**
119          * <p>等価性のテストを行います。</p>
120          */
121         @Test
122         public void testEquals() {
123                 User user = new User();
124
125                 user.setName(UserTest.class.getCanonicalName());
126                 user.setPasswordString(UserTest.class.getCanonicalName());
127
128                 User another = new User();
```

```
129
130                 another.setName(UserTest.class.getCanonicalName());
131                 another.setPasswordString(UserTest.class.getCanonicalName());
132
133                 Query query = this.manager.createQuery(
134                         "SELECT u FROM User AS u WHERE name = :name");
135                 query.setParameter("name", UserTest.class.getCanonicalName());
136
137
138                 Assert.assertEquals(user, another);
139
140                 this.manager.getTransaction().begin();
141
142                 this.manager.persist(user);
143
144                 user = (User) query.getSingleResult();
145
146                 this.manager.getTransaction().rollback();
147
148                 Assert.assertEquals(user, another);
149                 Assert.assertEquals(user.hashCode(), another.hashCode());
150         }
151
152         /**
153          * <p>パスワードがハッシュ化されているかどうかのテストを行います。</p>
154          */
155         @Test
156         public void testConvertingPassword() {
157                 String password = "password";
158
159                 User user = new User();
160
161                 user.setPasswordString(UserTest.class.getCanonicalName());
162
163                 Assert.assertFalse(password.equals(user.getPassword()));
164
165                 Matcher matcher = PASSWORD_HASH_PATTERN.matcher(user.getPassword
166 ());
167                 Assert.assertTrue(matcher.matches());
168         }
169 }
```

¥resources¥META-INF

（1）persistence.xml

persistence.xml　　Page 1

```
 1 <?xml version="1.0" encoding="UTF-8" ?>
 2 <persistence xmlns="http://java.sun.com/xml/ns/persistence"
 3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4     xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.c
 5 om/xml/ns/persistence/persistence_1_0.xsd"
 6     version="1.0">
 7
 8
 9   <persistence-unit name="testManager" transaction-type="RESOURCE_LOCAL">
10     <provider>org.hibernate.ejb.HibernatePersistence</provider>
11
12     <class>jp.ac.u_toyama.itc.kokushu.webapp.user.Group</class>
13     <class>jp.ac.u_toyama.itc.kokushu.webapp.user.User</class>
14
15     <properties>
16
17       <property name="hibernate.dialect"
18           value="org.hibernate.dialect.PostgreSQLDialect" />
19       <property name="hibernate.connection.driver_class"
20           value="org.postgresql.Driver" />
21       <property name="hibernate.connection.username" value="kokushu_webapp_user_
22 owner" />
23       <property name="hibernate.connection.password"
24           value="m9IGhsVj1uUu1RuLwDm5Pzkkam5mzj" />
25       <property name="hibernate.connection.url"
26           value="jdbc:postgresql://localhost/kokushu_webapp_user" />
27       <property name="hibernate.show_sql" value="true" />
28
29     </properties>
30
31   </persistence-unit>
32
33 </persistence>
```

６）koiushu-webapp-user-manager：刻手データを検索するウェブアプリケーションの登録ユー ザを管理するアプリケーション

```
koiushu-webapp-user-manager
    pom.xml

    ¥src¥main
      ¥assembly
        bin.xml
      ¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥application¥usermanager
        AbstractKokushuUserDatabase.java
        KokushuUserDatabase.java
        UserManager.java
      ¥resources
        ¥META-INF
          Persistence.xml
        commons-logging.properties
        simplelog.properties
      ¥script
        kokushu-webapp-user-manager.bat

    ¥src test
      ¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥application¥usermanager
        KokushuUserDatabaseTest.java
      ¥resources
        ¥META-INF
          Persistence.xml
        simplelog.properties
```

koiushu-webapp-user-manager

（1）pom.xml

pom.xml　　Page 1

```
 1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/
 2 2001/XMLSchema-instance"
 3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
 4 maven-v4_0_0.xsd">
 5
 6   <parent>
 7     <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
 8     <artifactId>kokushu</artifactId>
 9     <version>1.0-SNAPSHOT</version>
10     <relativePath>../kokushu/pom.xml</relativePath>
11   </parent>
12
13   <modelVersion>4.0.0</modelVersion>
14   <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
15   <artifactId>kokushu-webapp-user-manager</artifactId>
16   <packaging>jar</packaging>
17   <version>1.0-SNAPSHOT</version>
18   <name>kokushu-webapp-user-manager</name>
19   <url>http://maven.apache.org</url>
20
21
22   <build>
23
24     <plugins>
25
26      <plugin>
27
28         <artifactId>maven-assembly-plugin</artifactId>
29
30         <configuration>
31
32           <descriptors>
33             <descriptor>src/main/assembly/bin.xml</descriptor>
34           </descriptors>
35
36         </configuration>
37
38       </plugin>
39
40     </plugins>
41
42   </build>
43
44   <dependencies>
45     <dependency>
46       <groupId>jp.ac.u_toyama.itc.kokushu</groupId>
47       <artifactId>kokushu-webapp-user-db</artifactId>
48       <version>1.0-SNAPSHOT</version>
49     </dependency>
50
51     <dependency>
52       <groupId>jp.ac.u_toyama.itc.dokb</groupId>
53       <artifactId>simple-account-manager</artifactId>
54       <version>1.0-SNAPSHOT</version>
55     </dependency>
56
57     <dependency>
58       <groupId>commons-lang</groupId>
59       <artifactId>commons-lang</artifactId>
60       <version>2.3</version>
61     </dependency>
62
63     <dependency>
64       <groupId>junit</groupId>
65       <artifactId>junit</artifactId>
66       <version>4.4</version>
67       <scope>test</scope>
68     </dependency>
69
70   </dependencies>
71
72 </project>
```

¥src¥main¥assembly

（1）bin.xml

bin.xml　　Page 1

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <assembly>
 3
 4   <id>bin</id>
 5
 6   <formats>
 7     <format>zip</format>
 8   </formats>
 9
10   <includeBaseDirectory>false</includeBaseDirectory>
11
12   <dependencySets>
13
14     <dependencySet>
15       <outputDirectory>/lib</outputDirectory>
16       <outputFileNameMapping>${artifact.artifactId}.${artifact.extension}</outpu
17 tFileNameMapping>
18       <unpack>false</unpack>
19       <scope>runtime</scope>
20     </dependencySet>
21
22   </dependencySets>
23
24   <fileSets>
25
26     <fileSet>
27
28       <includes>
29         <include>README*</include>
30       </includes>
31
32       <outputDirectory>/</outputDirectory>
33
34     </fileSet>
35
36     <fileSet>
37
38       <directory>src/main/script</directory>
39
40       <outputDirectory>/</outputDirectory>
41
42     </fileSet>
43
44     <fileSet>
45
46       <directory>target</directory>
47
48       <includes>
49
50         <include>*.jar</include>
51
52       </includes>
53
54       <outputDirectory>/</outputDirectory>
55
56     </fileSet>
57
58   </fileSets>
59
60
61
62 </assembly>
```

¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥application¥usermanager

（1）AbstractKokushuUserDatabase.java

AbstractKokushuUserDatabase.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.application.usermanager;
 2
 3 import javax.persistence.EntityManager;
 4 import javax.persistence.EntityManagerFactory;
 5 import javax.persistence.Persistence;
 6
 7 import jp.ac.u_toyama.itc.dokb.application.AccountDatabase;
 8 import jp.ac.u_toyama.itc.kokushu.webapp.user.User;
 9
10 import org.apache.commons.lang.Validate;
11
12 /**
13  * <p>刻手データベース検索システムのユーザ情報にアクセスする共通の
14  * 機能を提供します。</p>
15  * <p>EntityManager やトランザクションの管理を行います。サブクラスは、
16  * EntityManager やトランザクションの管理を行う必要がありません。</p>
17  * <p>このクラスはスレッドセーフではありません。</p>
18  *
19  * @author Keita Kita (2008/04/18 -)
20  * @since 1.0-SNAPSHOT
21  */
22 public abstract class AbstractKokushuUserDatabase
23                 implements AccountDatabase {
24         /**
25          * <p>ユーザ情報にアクセスする PersistentUnit 名です。</p>
26          */
27         private static final String UNIT_NAME = "userManager";
28
29         /**
30          * <p>ユーザ情報にアクセスする EntityManagerFactory です。
31          */
32         private final EntityManagerFactory factory;
33
34
35         /**
36          * <p>コンストラクタです。</p>
37          */
38         public AbstractKokushuUserDatabase() {
39                 this.factory = Persistence.createEntityManagerFactory(UNIT_NAME)
40 ;
41         }
42
43         public final void close() {
44                 if (this.factory.isOpen()) {
45                         this.factory.close();
46                 }
47         }
48
49         public final void create(String name, String password) {
50                 Validate.notNull(name);
51                 Validate.notNull(password);
52
53                 // 登録するユーザ
54                 User user = new User();
55
56                 user.setName(name);
57                 user.setPasswordString(password);
58
59                 // ユーザ情報にアクセスする EntityManager
60                 EntityManager manager = this.factory.createEntityManager();
61
62                 manager.getTransaction().begin();
63
64                 try {
```

```
65                          this.createUser(user, manager);
66                          manager.getTransaction().commit();
67                  }
68                  catch (RuntimeException e) {
69                          manager.getTransaction().rollback();
70
71                          throw e;
72                  }
73                  finally {
74                          manager.close();
75                  }
76          }
77
78          /**
79           * <p>指定のユーザを登録します。</p>
80           * <p>本メソッドはトランザクション内で実行されます。</p>
81           * <p>トランザクションや EntityManager の管理は、
82           * 本メソッドの呼び出し元が行います。</p>
83           *
84           * @see #create(String, String)
85           *
86           * @param user 登録するユーザ (非 null)
87           * @param manager EntityManager (非 null)
88           * @throws RuntimeException ユーザの登録に失敗した場合
89           */
90          protected abstract void createUser(User user, EntityManager manager);
91
92          public final void delete(String name) {
93                  Validate.notNull(name);
94
95                  // ユーザ情報にアクセスする EntityManager
96                  EntityManager manager = this.factory.createEntityManager();
97
98                  manager.getTransaction().begin();
99
100                 try {
101                         this.deleteUser(name, manager);
102                         manager.getTransaction().commit();
103                 }
104                 catch (RuntimeException e) {
105                         manager.getTransaction().rollback();
106
107                         throw e;
108                 }
109                 finally {
110                         manager.close();
111                 }
112         }
113
114         /**
115          * <p>指定のユーザの削除を行います。</p>
116          * <p>本メソッドは、トランザクション内で実行されます。</p>
117          * <p>トランザクションや EntityManager の管理は、
118          * 本メソッドの呼び出し元が行います。</p>
119          *
120          * @see #delete(String)
121          *
122          * @param name 削除するユーザ名 (非 null)
123          * @param manager EntityManager (非 null)
124          * @throws IllegalArgumentException ユーザが存在しない場合
125          * @throws RuntimeException ユーザの削除に失敗した場合
126          */
127         protected abstract void deleteUser(String name, EntityManager manager);
128
```

```
129         public final String list() {
130                 // ユーザ情報にアクセスする EntityManager
131                 EntityManager manager = this.factory.createEntityManager();
132
133                 manager.getTransaction().begin();
134
135                 try {
136                         String result = this.listUsers(manager);
137
138                         manager.getTransaction().commit();
139
140                         return result;
141                 }
142                 catch (RuntimeException e) {
143                         manager.getTransaction().rollback();
144
145                         throw e;
146                 }
147                 finally {
148                         manager.close();
149                 }
150         }
151
152         /**
153          * <p>ユーザの情報の文字列表現を返します。</p>
154          * <p>本メソッドは、トランザクション内で実行されます。</p>
155          * <p>トランザクションや EntityManager の管理は、
156          * 本メソッドの呼び出し元が行います。</p>
157          *
158          * @see #list()
159          *
160          * @param manager EntityManager (非 null)
161          * @return ユーザの情報の文字列表現
162          */
163         protected abstract String listUsers(EntityManager manager);
164 }
```

（2）KokushuUserDatabase.java

KokushuUserDatabase.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.application.usermanager;
 2
 3 import java.util.List;
 4
 5 import javax.persistence.EntityManager;
 6 import javax.persistence.NoResultException;
 7 import javax.persistence.Query;
 8
 9 import jp.ac.u_toyama.itc.kokushu.webapp.user.Group;
10 import jp.ac.u_toyama.itc.kokushu.webapp.user.User;
11
12 /**
13  * <p>刻手データベース検索システムのユーザ情報にアクセスします。</p>
14  * <p>このクラスはスレッドセーフではありません。</p>
15  *
16  * @author Keita Kita (2008/04/16 -)
17  * @since 1.0-SNAPSHOT
18  */
19 public class KokushuUserDatabase extends AbstractKokushuUserDatabase {
20         /**
21          * <p>ユーザが所属するグループ名です。</p>
22          */
23         private static final String GROUP_NAME = "user";
24
25
26         @Override
27         protected void createUser(User user, EntityManager manager) {
28                 // ユーザ名が重複していないかどうか検証
29                 if (this.isDuplicateName(user.getName(), manager)) {
30                         throw new IllegalArgumentException(
31                                 user.getName() + "はすでに登録済みです。");
32                 }
33
34                 // 登録するグループ（エンティティオブジェクト）
35                 Group group = this.getGroup(manager);
36
37                 // グループにユーザを追加
38                 group.getUsers().add(user);
39                 user.getGroups().add(group);
40
41                 manager.persist(user);
42         }
43
44         /**
45          * <p>指定のユーザ名がすでに登録済みかどうかを調べます。</p>
46          * <p>本メソッドは、トランザクション内で実行する必要があります。</p>
47          *
48          * @param name すでに登録済みかどうかを調べるユーザ名
49          * @param manager ユーザ情報にアクセスする EntityManager
50          * @return すでに登録済みの場合は true
51          */
52         private boolean isDuplicateName(String name, EntityManager manager) {
53                 assert (name != null);
54
55                 // 指定の名前のユーザが存在するかどうかを調べるクエリー
56                 Query query = manager.createQuery(
57                         "SELECT COUNT(u) FROM User AS u WHERE name = :name");
58                 query.setParameter("name", name);
59
60                 return (1 <= (Long) query.getSingleResult());
61         }
62
63         /**
64          * <p>ユーザが所属するグループのオブジェクトを取得します。</p>
```

```
65          * <p>本メソッドはトランザクション内で実行する必要があります。</p>
66          *
67          * @param manager グループの情報にアクセスする EntityManager
68          * @return ユーザが所属するグループのエンティティオブジェクト
69          * @throws RuntimeException データベースのアクセスに失敗した場合
70          */
71         private Group getGroup(EntityManager manager) {
72                 // ユーザが所属するグループを取得するクエリー
73                 Query query = manager.createQuery(
74                         "SELECT g FROM Group AS g WHERE name = :name");
75                 query.setParameter("name", GROUP_NAME);
76
77                 // ユーザが所属するグループがデータベース内になければ生成し、
78                 // あれば登録されているものを返す
79
80                 if (query.getResultList().size() == 0) {
81                         // 新たに登録するグループ
82                         Group group = new Group();
83                         group.setName(GROUP_NAME);
84
85                         manager.persist(group);
86                 }
87
88                 // Group.name は unique 制約のため、1つしかないはず
89                 return (Group) query.getSingleResult();
90         }
91
92         @Override
93         protected void deleteUser(String name, EntityManager manager) {
94                 // 指定の名前のユーザを検索するクエリー
95                 Query query = manager.createQuery(
96                         "SELECT u FROM User AS u WHERE name = :name");
97                 query.setParameter("name", name);
98
99                 // User.name は unique 制約のため、オブジェクトは1つしかないはず
100
101                try {
102                        manager.remove(query.getSingleResult());
103                }
104                catch (NoResultException e) {
105                        throw new IllegalArgumentException("ユーザが存在しません
106 。", e);
107                }
108        }
109
110        @Override
111        @SuppressWarnings("unchecked")
112        protected String listUsers(EntityManager manager) {
113                // 全ユーザを取得するクエリー
114                Query query = manager.createQuery("SELECT u FROM User AS u");
115                // 文字列表現を格納する文字列バッファ
116                StringBuilder list = new StringBuilder();
117
118                list.append("ユーザ名：所属するグループ名¥n");
119                list.append("--------------------------¥n");
120
121                List<User> users = query.getResultList();
122
123                // 文字列表現の生成
124
125                for (User aUser : users) {
126                        list.append(aUser.getName());
127                        list.append(" : ");
128
```

```
129                         // 初めのグループの出力かどうかを表すフラグ
130                         boolean isFirst = true;
131
132                         // 所属するグループを並べる
133                         for (Group aGroup : aUser.getGroups()) {
134                                 // 2つめ以降のグループ名の前に区切りを出力
135                                 if (!isFirst) {
136                                         list.append(", ");
137                                 }
138                                 list.append(aGroup.getName());
139                         }
140
141                         list.append("¥n");
142                 }
143
144                 return list.toString();
145         }
146 }
```

（3）UserManager.java

UserManager.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.application.usermanager;
 2
 3 import jp.ac.u_toyama.itc.dokb.application.SimpleAccountManagerMain;
 4
 5 /**
 6  * <p>刻手データベース検索システムのユーザ情報を管理するアプリケーションです。</
 7 p>
 8  * <p>ユーザ名とパスワードは、ともに半角英数字です。</p>
 9  * <p>引数</p>
10  * <dl>
11  *      <dt>-c, --create &lt;ユーザ名&gt; &lt;パスワード&gt;</dt>
12  *      <dd>ユーザを登録します。すでにユーザがある場合は、
13  *      メッセージを表示して終了します。
14  *      <dt>-d, --delete &lt;ユーザ名&gt;</dt>
15  *      <dd>ユーザを削除します。すでにユーザがない場合は
16  *      何もしません。</dd>
17  *  <dt>-h, --help</dt>
18  *  <dd>ヘルプを表示して終了します。</dd>
19  * </dl>
20  *
21  * @author Keita Kita (2008/04/16 -)
22  * @since 1.0-SNAPSHOT
23  */
24 public class UserManager {
25         /**
26          * <p>アプリケーションを実行します。</p>
27          *
28          * @param args プログラム引数
29          */
30         public static final void main(String[] args) {
31                 SimpleAccountManagerMain application =
32                         new SimpleAccountManagerMain(new KokushuUserDatabase());
33
34
35                 application.main(args);
36         }
37 }
```

¥resources¥META-INF

（1）Persistence.xml

persistence.xml　　Page 1

```
 1 <?xml version="1.0" encoding="UTF-8" ?>
 2 <persistence xmlns="http://java.sun.com/xml/ns/persistence"
 3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4     xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.c
 5 om/xml/ns/persistence/persistence_1_0.xsd"
 6     version="1.0">
 7
 8
 9   <persistence-unit name="userManager" transaction-type="RESOURCE_LOCAL">
10     <provider>org.hibernate.ejb.HibernatePersistence</provider>
11
12     <class>jp.ac.u_toyama.itc.kokushu.webapp.user.Group</class>
13     <class>jp.ac.u_toyama.itc.kokushu.webapp.user.User</class>
14
15     <properties>
16
17       <property name="hibernate.dialect"
18           value="org.hibernate.dialect.PostgreSQLDialect" />
19       <property name="hibernate.connection.driver_class"
20           value="org.postgresql.Driver" />
21       <property name="hibernate.connection.username" value="kokushu_webapp_user_
22 owner" />
23       <property name="hibernate.connection.password"
24           value="m9IGhsVj1uUu1RuLwDm5Pzkkam5mzj" />
25       <property name="hibernate.connection.url"
26           value="jdbc:postgresql://localhost/kokushu_webapp_user" />
27       <property name="hibernate.show_sql" value="false" />
28
29     </properties>
30
31   </persistence-unit>
32
33 </persistence>
```

¥resources

（1）commons-logging.properties

commons-logging.properties　　Page 1

```
1 org.apache.commons.logging.Log = org.apache.commons.logging.impl.SimpleLog
2 org.apache.commons.logging.simplelog.defaultlog = warn
```

（2）simplelog.properties

commons-logging.properties　　Page 1

```
1 org.apache.commons.logging.Log = org.apache.commons.logging.impl.SimpleLog
2 org.apache.commons.logging.simplelog.defaultlog = warn
```

¥script

（1）kokushu-webapp-user-manager.bat

kokushu-webapp-user-manager.bat　　Page 1

```
1 java -classpath *;lib/* jp.ac.u_toyama.itc.kokushu.application.usermanager.UserM
2 anager %*
```

¥src¥test¥java¥jp¥ac¥u_toyama¥itc¥kokushu¥application¥usermanager

（1）KokushuUserDatabaseTest.java

KokushuUserDatabaseTest.java　　Page 1

```
 1 package jp.ac.u_toyama.itc.kokushu.application.usermanager;
 2
 3 import javax.persistence.EntityManager;
 4 import javax.persistence.EntityManagerFactory;
 5 import javax.persistence.Persistence;
 6 import javax.persistence.Query;
 7
 8 import jp.ac.u_toyama.itc.kokushu.webapp.user.User;
 9 import junit.framework.Assert;
10
11 import org.junit.After;
12 import org.junit.AfterClass;
13 import org.junit.Before;
14 import org.junit.BeforeClass;
15 import org.junit.Test;
16
17
18 /**
19  * <p>KokushuUserDatabase をテストします。</p>
20  * <p>本テストでは、親クラスが担当する処理のテストは行いません。</p>
21  *
22  * @author Keita Kita (2008/04/18 -)
23  * @since 1.0-SNAPSHOT
24  */
25 public class KokushuUserDatabaseTest {
26         /**
27          * <p>ユーザ情報にアクセスする EntityManagerFactory です。</p>
28          */
29         private static EntityManagerFactory factory;
30
31         /**
32          * <p>ユーザ情報にアクセスする EntityManager です。</p>
33          */
34         private EntityManager manager;
35
36         /**
37          * <p>テスト対象のオブジェクトです。</p>
38          */
39         private KokushuUserDatabase database;
40
41
42         /**
43          * <p>全テストメソッドが実行される前に呼び出されます。</p>
44          */
45         @BeforeClass
46         public static void setUpAll() {
47                 factory = Persistence.createEntityManagerFactory("testManager");
48
49         }
50
51         /**
52          * <p>全テストメソッドの実行後に呼び出されます。</p>
53          */
54         @AfterClass
55         public static void tearDownAll() {
56                 factory.close();
57         }
58
59         /**
60          * <p>各テストメソッドの実行前に呼び出されます。</p>
61          */
62         @Before
63         public void setUp() {
64                 this.database = new KokushuUserDatabase();
```

```
65                  this.manager = factory.createEntityManager();
66                  this.manager.getTransaction().begin();
67          }
68
69          /**
70           * <p>各テストメソッドの実行後に呼び出されます。</p>
71           * <p>ロールバックを行います。</p>
72           */
73          @After
74          public void tearDown() {
75                  this.manager.getTransaction().rollback();
76                  this.manager.close();
77
78                  // 親クラスの担当する処理には関わらないため、close メソッドは呼
79 ばない
80          }
81
82          /**
83           * <p>ユーザの登録をテストします。</p>
84           * <p>正しくデータベースに登録を行えるかどうかをテストします。</p>
85           */
86          @Test
87          public void testCreating() {
88                  // 登録対象のユーザ
89                  // 2つ登録することで、グループが重複して登録されないかどうかを見
90 る
91                  User user1 = new User();
92                  user1.setName("testCreating");
93                  user1.setPasswordString(User.class.getCanonicalName());
94
95                  User user2 = new User();
96                  user2.setName(KokushuUserDatabaseTest.class.getCanonicalName());
97
98                  user2.setPasswordString(KokushuUserDatabaseTest.class.getCanonic
99 alName());
100
101                 this.database.createUser(user1, this.manager);
102                 this.database.createUser(user2, this.manager);
103
104                 // ユーザが登録されたかどうかを確かめる
105
106                 // 登録済みのユーザを検索するクエリー
107                 Query userQuery = this.manager.createQuery(
108                         "SELECT u FROM User AS u WHERE name = :name");
109                 userQuery.setParameter("name", user2.getName());
110
111                 // データベースから取り出したユーザ
112                 User registed = (User) userQuery.getSingleResult();
113                 // グループに登録されているはず
114                 Assert.assertFalse(registed.getGroups().isEmpty());
115
116                 // グループが登録されたかどうかを確かめる
117
118                 // グループを検索するクエリー
119                 Query groupQuery = this.manager.createQuery(
120                         "SELECT g FROM Group AS g WHERE name = :name");
121                 groupQuery.setParameter("name", "user");
122
123                 // 1つのみ登録されているはず
124                 Assert.assertNotNull(groupQuery.getSingleResult());
125         }
126
127         /**
128          * <p>ユーザの削除をテストします。</p>
```

```
129          * <p>正しく削除されるかどうかをテストします。</p>
130          */
131         @Test
132         public void testDeleting() {
133                 // 登録・削除するユーザ
134                 User user = new User();
135
136                 user.setName(User.class.getCanonicalName());
137                 user.setPasswordString(User.class.getCanonicalName());
138
139                 // 登録するユーザ (削除はしない)
140                 User userNotDeleted = new User();
141
142                 userNotDeleted.setName(
143                         KokushuUserDatabaseTest.class.getCanonicalName());
144                 userNotDeleted.setPasswordString(
145                         KokushuUserDatabaseTest.class.getCanonicalName());
146
147                 // 登録
148
149                 this.database.createUser(user, this.manager);
150                 this.database.createUser(userNotDeleted, this.manager);
151
152                 // 登録したユーザを検索するクエリー
153                 Query userQuery = this.manager.createQuery(
154                         "SELECT COUNT(u) FROM User AS u WHERE name = :name");
155                 // グループの数を検索するクエリー
156                 Query groupQuery = this.manager.createQuery(
157                         "SELECT COUNT(g) FROM Group AS g WHERE name = :name");
158
159                 // 登録されているはず
160                 userQuery.setParameter("name", User.class.getCanonicalName());
161                 Assert.assertEquals(new Long(1), userQuery.getSingleResult());
162
163                 // 削除
164
165                 this.database.deleteUser(User.class.getCanonicalName(), this.man
166 ager);
167
168                 // 削除されているはず
169                 Assert.assertEquals(new Long(0), userQuery.getSingleResult());
170
171                 // 削除されていないユーザは残っているはず
172                 userQuery.setParameter("name",
173                         KokushuUserDatabaseTest.class.getCanonicalName());
174                 Assert.assertEquals(new Long(1), userQuery.getSingleResult());
175
176                 // グループも残っているはず
177                 groupQuery.setParameter("name", "user");
178                 Assert.assertEquals(new Long(1), groupQuery.getSingleResult());
179         }
180
181         /**
182          * <p>ユーザ情報の文字列表現化をテストします。</p>
183          * <p>ただ単に、文字列が返されているかどうかをテストするだけです。</p>
184          */
185         @Test
186         public void testListing() {
187                 User user = new User();
188                 user.setName("testListing");
189                 user.setPasswordString(User.class.getCanonicalName());
190
191                 this.database.createUser(user, this.manager);
192
```

```
193                     // 文字列が返されているかどうかをテストする
194                     Assert.assertTrue(this.database.list().length() > 0);
195             }
196 }
```

¥resources¥META-INF

（1）Persistence.xml

persistence.xml　　Page 1

```
 1 <?xml version="1.0" encoding="UTF-8" ?>
 2 <persistence xmlns="http://java.sun.com/xml/ns/persistence"
 3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4     xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.c
 5 om/xml/ns/persistence/persistence_1_0.xsd"
 6     version="1.0">
 7
 8
 9   <persistence-unit name="testManager" transaction-type="RESOURCE_LOCAL">
10     <provider>org.hibernate.ejb.HibernatePersistence</provider>
11
12     <class>jp.ac.u_toyama.itc.kokushu.webapp.user.Group</class>
13     <class>jp.ac.u_toyama.itc.kokushu.webapp.user.User</class>
14
15     <properties>
16
17       <property name="hibernate.dialect"
18           value="org.hibernate.dialect.PostgreSQLDialect" />
19       <property name="hibernate.connection.driver_class"
20           value="org.postgresql.Driver" />
21       <property name="hibernate.connection.username" value="kokushu_webapp_user_
22 owner" />
23       <property name="hibernate.connection.password"
24           value="m9IGhsVj1uUu1RuLwDm5Pzkkam5mzj" />
25       <property name="hibernate.connection.url"
26           value="jdbc:postgresql://localhost/kokushu_webapp_user" />
27       <property name="hibernate.show_sql" value="true" />
28
29     </properties>
30
31   </persistence-unit>
32
33 </persistence>
```

¥resources

（1）simplelog.properties

simplelog.properties　　Page 1

```
1 org.apache.commons.logging.simplelog.defaultlog = info
```

［資料９－３］印影・原文画像データベース検索システム開発ソース・コード

［ソフトウェア構成一覧］

```
    tables.sql
  ¥db¥dao
    AbstractDAO.java
    AppendixDAO.java
    AuthorDAO.java
    CategoryDAO.java
    DocumentMetadataDAO.java
    LibraryDAO.java

  ¥db¥data
    Appendix.java
    Author.java
    DocumentMetadata.java

  ¥service¥reference
    ¥impl
      DocumentMetadataReferenceBean.java
    DocumentMetadataReference.java
    Limit.java
    ReferenceView.java
  ¥service¥storage
    ¥impl
      DocumentMetadataStorageBean.java
    DocumentMetadataStorage.java
```

DocumentMetadataStorage.java Page 1

```
1  package jp.ac.u_toyama.itc.dcdb.service.storage;
2
3  import javax.ejb.Remote;
4
5  import jp.ac.u_toyama.itc.dcdb.metadata.data.DocumentMetadata;
6
7  /**
8   * <p>文書メタデータのデータベースへの操作を行うためのビジネスインタフェースです。</p>
9   * <p>各メソッドには、永続化機構で管理されていないオブジェクトを渡すことができます。</p>
10  * <p>このクラスにはスレッドセーフでは有りません。</p>
11  *
12  * @author Keita Kita (2007/12/07 -)
13  * @since 1.0-SNAPSHOT
14  */
15 @Remote
16 public interface DocumentMetadataStorage {
17         /**
18          * <p>指定の文書メタデータを追加します。</p>
19          *
20          * @param data 追加する文書メタデータ
21          */
22         void add( DocumentMetadata data );
23
24         /**
25          * <p>指定の文書メタデータを更新します。</p>
26          *
27          * @param data 更新する文書メタデータ
28          */
29         void update( DocumentMetadata data );
30
31         /**
32          * <p>指定の文書メタデータを削除します。</p>
33          *
34          * @param data 削除する文書メタデータ
35          */
36         void delete( DocumentMetadata data );
37 }
```

tables.sql Page 1

```
1  BEGIN;
2
3  /*
4          文書メタデータ
5          （全国横断目録データベース形式にほぼ準拠）
6  */
7  CREATE TABLE document_metadata (
8          -- 文書メタデータID （帳号） （自動生成）
9          number serial PRIMARY KEY,
10         -- 分類（階） （必ず存在する）
11         four_category text NOT NULL,
12         -- 分類（類） （存在しないかもしれない）
13         classification text DEFAULT '' NOT NULL,
14         -- 分類（綱） （存在しないかもしれない）
15         tag text DEFAULT '' NOT NULL,
16         -- 分類（目） （存在しないかもしれない）
17         kind text DEFAULT '' NOT NULL,
18         -- 分類コード
19         category_code varchar( 5 ) NOT NULL REFERENCES category ( code )
20                 ON DELETE NO ACTION
21                 ON UPDATE CASCADE,
22         -- 書名
23         title text DEFAULT '' NOT NULL,
24         -- 書名（カナ）
25         title_in_kana text DEFAULT '' NOT NULL,
26         -- 書名（pinyin）
27         title_in_pinyin text DEFAULT '' NOT NULL,
28         -- 「和」書名
29         another_title text DEFAULT '' NOT NULL,
30         -- 「和」書名（カナ）
31         another_title_in_kana text DEFAULT '' NOT NULL,
32         -- 「和」書名（pinyin）
33         another_title_in_pinyin text DEFAULT '' NOT NULL,
34         -- 刊年
35         year text DEFAULT '' NOT NULL,
36         -- 出版社名
37         publisher text DEFAULT '' NOT NULL,
38         -- エディション
39         edition text DEFAULT '' NOT NULL,
40         -- 複版
41         engraving_cut text DEFAULT '' NOT NULL,
42         -- 影刻対応
43         transcript text DEFAULT '' NOT NULL,
44         -- 叢書名
45         collection text DEFAULT '' NOT NULL,
46         -- 所蔵機関コード
47         library_code varchar( 7 ) NOT NULL
48                 REFERENCES library (code)
49                         ON DELETE NO ACTION
50                         ON UPDATE CASCADE,
51         -- 登録番号
52         register_number_in_library text DEFAULT '' NOT NULL,
53         -- 請求番号
54         request_number_in_library text DEFAULT '' NOT NULL,
55         -- 文庫 名称
56         name_in_library text DEFAULT '' NOT NULL,
57         -- 文庫 コード
58         code_in_library text DEFAULT '' NOT NULL,
59         -- 冊数
60         volume text DEFAULT '' NOT NULL,
61         -- 注記
62         note text DEFAULT '' NOT NULL
```

tables.sql Page 2

```
63 ) WITHOUT OIDS;
64
65 /*
66         附録データ
67 */
68 CREATE TABLE appendix_of_document (
69         ID （自動生成）
70         id serial PRIMARY KEY,
71         -- 含まれる文書
72         included_document_number integer NOT NULL
73                 REFERENCES document_metadata ( number )
74                         ON DELETE CASCADE
75                         ON UPDATE CASCADE,
76         -- 書名
77         title text DEFAULT '' NOT NULL,
78         -- 書名（カナ）
79         title_in_kana text DEFAULT '' NOT NULL,
80         -- 書名（pinyin）
81         title_in_pinyin text DEFAULT '' NOT NULL,
82         -- 刊年
83         year text DEFAULT '' NOT NULL,
84         -- 出版社名
85         publisher text DEFAULT '' NOT NULL,
86         -- エディション
87         edition text DEFAULT '' NOT NULL
88 ) WITHOUT OIDS;
89
90 /*
91         撰者データ
92 */
93 CREATE TABLE author (
94         -- 撰者 ID
95         id serial PRIMARY KEY,
96         -- 編者の文書
97         document_number integer NOT NULL REFERENCES document_metadata ( number )
98
99                 ON DELETE CASCADE
100                ON UPDATE CASCADE,
101        -- 撰者名 （空白とかはありえない）
102        name text NOT NULL CHECK( length( ltrim( name ) ) > 0 ),
103        -- pinyin
104        name_in_pinyin text DEFAULT '' NOT NULL,
105        -- 別名等々
106        name_in_kana text DEFAULT '' NOT NULL,
107
108        UNIQUE ( document_number, name )
109 ) WITHOUT OIDS;
110
111 /*
112         子書誌データ
113 */
114 CREATE TABLE child_number_of_document (
115        -- 文書メタデータID
116        document_number integer REFERENCES document_metadata ( number )
117                ON DELETE CASCADE
118                ON UPDATE CASCADE,
119        -- 子番号
120        child_number integer REFERENCES document_metadata ( number )
121                ON DELETE CASCADE
122                ON UPDATE CASCADE,
123
124        PRIMARY KEY ( document_number, child_number )
```

tables.sql Page 3

```
125 ) WITHOUT OIDS;
126
127 /*
128         親書誌データ
129 */
130 CREATE TABLE parent_number_of_document (
131        -- 文書メタデータID
132        document_number integer REFERENCES document_metadata ( number )
133                ON DELETE CASCADE
134                ON UPDATE CASCADE,
135        -- 親書誌
136        parent_number integer REFERENCES document_metadata ( number )
137                ON DELETE CASCADE
138                ON UPDATE CASCADE,
139
140        PRIMARY KEY ( document_number, parent_number )
141 ) WITHOUT OIDS;
142
143 /*
144         原文画像メタデータ
145 */
146 CREATE TABLE document_image_metadata (
147        -- 登録番号
148        number varchar( 5 ) PRIMARY KEY,
149                文書メタデータID
150        document_number integer NOT NULL REFERENCES document_metadata ( number )
151
152                ON DELETE CASCADE
153                ON UPDATE CASCADE,
154        -- 複写部分
155        part text DEFAULT '' NOT NULL,
156        -- ページ番号 （単一ページではないものもあるため、文字列）
157        page text NOT NULL,
158        -- 印影数
159        pictures integer NOT NULL CHECK( pictures > 0 ),
160        -- 登録サイズ
161        size text DEFAULT '' NOT NULL,
162        -- 書誌情報の参照番号
163        document_reference_number varchar(8) DEFAULT '' NOT NULL,
164        -- 備考
165        note text DEFAULT '' NOT NULL,
166        -- 画像ファイルへのパス （絶対パス）
167        path text NOT NULL UNIQUE,
168
169                ページ番号は文書中で一意なはず
170        UNIQUE( document_number, page )
171 ) WITHOUT OIDS;
172
173 /*
174         印影メタデータ
175 */
176 CREATE TABLE picture_metadata (
177        -- 登録番号
178        number varchar( 5 ) PRIMARY KEY,
179        -- 原文登録番号
180        document_image_number varchar( 5 ) NOT NULL
181                REFERENCES document_image_metadata ( number )
182                        ON DELETE CASCADE
183                        ON UPDATE CASCADE,
184        -- 取転されているページ
185        page text DEFAULT '' NOT NULL,
186        -- 複写部分
```

tables.sql Page 4

```
187         part text DEFAULT '' NOT NULL,
188         -- 印影サイズ
189         size text DEFAULT '' NOT NULL,
190         -- 備考
191         note text DEFAULT '' NOT NULL,
192         -- 写像ファイルへのパス（相対パス）
193         path text NOT NULL UNIQUE
194 ) WITHOUT OIDS;
195
196 COMMIT;
```

AbstractDAO.java Page 1

```
 1 package jp.ac.u_toyama.itc.dokb.metadata.dao;
 2
 3 import java.io.Serializable;
 4
 5 import javax.persistance.EntityManager;
 6
 7 import org.apache.commons.lang.Validate;
 8
 9 /**
10  * <p>DAO に共通な機能を提供します。</p>
11  * <p>このクラスはスレッドセーフではありません。</p>
12  *
13  * @param < T > DAO の対象となるクラス
14  * @param < PK > 主キーのクラス
15  *
16  * @author Keita Kita (2007/10/05 -)
17  * @since 1.1-SNAPSHOT
18  */
19 public abstract class AbstractDAO< T, PK extends Serializable > {
20         /**
21          * <p>EntityManager です。</p>
22          */
23         private final EntityManager entityManager;
24
25         /**
26          * <p>対象となるクラスです。</p>
27          */
28         private final Class< T > targetClass;
29
30         /**
31          * <p>コンストラクタです。</p>
32          *
33          * @param entityManager EntityManager
34          * @param targetClass 対象となるクラス
35          * @throws IllegalArgumentException 引数が null の場合
36          */
37         protected AbstractDAO( EntityManager entityManager,
38                                Class< T > targetClass ) {
39                 Validate.notNull( entityManager );
40
41                 this.entityManager = entityManager;
42                 this.targetClass = targetClass;
43         }
44
45         /**
46          * <p>データを永続化します。</p>
47          *
48          * @param data 永続化する文書メタデータ
49          * @throws IllegalArgumentException 引数が null の場合
50          * @throws IllegalStateException EntityManager が終了している場合
51          */
52         public void create( T data ) {
53                 this.entityManager.persist( data );
54         }
55
56         /**
57          * <p>指定の主キーのデータを読み込みます。</p>
58          * <p>指定の主キーのデータが存在しない場合は、null をとなります。</p>
59          *
60          * @param key 主キー
61          * @return 指定の主キーのデータ、存在しない場合は null
62          * @throws IllegalStateException EntityManager が終了している場合
```

AbstractDAO.java Page 2

```
63          */
64         public T read( PK key ) {
65                 return this.entityManager.find( this.targetClass, key );
66         }
67
68         /**
69          * <p>データを永続化機構に接続しなおします。</p>
70          *
71          * @param data 接続しなおすデータ
72          * @throws IllegalArgumentException 指定のオブジェクトが不正な場合
73          * @throws IllegalStateException EntityManager が終了している場合
74          */
75         public void refresh( T data ) {
76                 this.entityManager.refresh( data );
77         }
78
79         /**
80          * <p>指定の永続化機構中のデータを更新します。</p>
81          *
82          * @param data 更新するデータ
83          * @throws IllegalArgumentException 指定のオブジェクトが不正な場合
84          * @throws IllegalStateException EntityManager が終了している場合
85          */
86         public void update( T data ) {
87                 this.entityManager.merge( data );
88         }
89
90         /**
91          * <p>指定の永続化機構中のデータを削除します。</p>
92          *
93          * @param data 削除するデータ
94          * @throws IllegalArgumentException 指定のデータが不正な場合
95          * @throws IllegalStateException EntityManager が終了している場合
96          */
97         public void delete( T data ) {
98                 this.entityManager.remove( data );
99         }
100
101         /**
102          * <p>EntityManager を返します。</p>
103          *
104          * @return EntityManager
105          */
106         protected EntityManager getEntityManager() {
107                 return this.entityManager;
108         }
109 }
```

AppendixDAO.java Page 1

```
 1 package jp.ac.u_toyama.itc.dokb.metadata.dao;
 2
 3 import javax.persistence.EntityManager;
 4
 5 import jp.ac.u_toyama.itc.dokb.metadata.data.Appendix;
 6
 7 /**
 8  * <p>永続機構中の補助データにアクセスする DAO です。</p>
 9  * <p>このクラスはスレッドセーフではありません。</p>
10  *
11  * @author Keita Kita (2007/10/05 -)
12  * @since 1.1-SNAPSHOT
13  */
14 public class AppendixDAO extends AbstractDAO< Appendix, Integer > {
15         /**
16          * <p>コンストラクタです。</p>
17          *
18          * @param entityManager EntityManager
19          * @throws IllegalArgumentException 引数が null の場合
20          */
21         public AppendixDAO( EntityManager entityManager ) {
22                 super( entityManager, Appendix.class );
23         }
24 }
```

AuthorDAO.java Page 1

```
 1 package jp.ac.u_toyama.ite.dokb.metadata.dao;
 2
 3 import javax.persistence.EntityManager;
 4 import javax.persistence.Query;
 5
 6 import jp.ac.u_toyama.ite.dokb.metadata.data.Author;
 7
 8 /**
 9  * <p>永続化機構中の投者データにアクセスする DAO です。</p>
10  * <p>このクラスはスレッドセーフではありません。</p>
11  *
12  * @author Keita Kita (2007/10/05 -)
13  * @since 1.1-SNAPSHOT
14  */
15 public class AuthorDAO extends AbstractDAO< Author, Integer > {
16         /**
17          * <p>コンストラクタです。</p>
18          *
19          * @param entityManager EntityManager
20          * @throws IllegalArgumentException 引数が null の場合
21          */
22         public AuthorDAO( EntityManager entityManager ) {
23                 super( entityManager, Author.class );
24         }
25
26         /**
27          * <p>全投者データを取得するクエリーを生成します。</p>
28          *
29          * @return 全投者データを取得するクエリー
30          * @throws IllegalStateException EntityManager が終了している場合
31          */
32         public Query getAll() {
33                 return this.getEntityManager().createQuery( "FROM Author" );
34         }
35
36         /**
37          * <p>全投者データの件数を取得します。</p>
38          *
39          * @return 全投者データの件数
40          * @throws IllegalStateException EntityManager が終了している場合
41          */
42         public long getTotal() {
43                 Query query = this.getEntityManager().createQuery(
44                                 "SELECT COUNT( a ) FROM Author AS a" );
45
46                 return ( Long ) query.getSingleResult();
47         }
48 }
```

CategoryDAO.java Page 1

```
 1 package jp.ac.u_toyama.ite.dokb.metadata.dao;
 2
 3 import java.util.List;
 4
 5 import javax.persistence.EntityManager;
 6 import javax.persistence.Query;
 7
 8 import org.apache.commons.lang.Validate;
 9
10 import jp.ac.u_toyama.ite.dokb.data.Category;
11
12 /**
13  * <p>永続機構中の分類情報にアクセスする DAO です。</p>
14  * <p>このクラスはスレッドセーフではありません。</p>
15  *
16  * @author Keita Kita (2007/12/07 -)
17  * @since 1.1-SNAPSHOT
18  */
19 public class CategoryDAO extends AbstractDAO< Category, Integer > {
20         /**
21          * <p>コンストラクタです。</p>
22          *
23          * @param entityManager EntityManager
24          * @throws IllegalArgumentException 引数が null の場合
25          */
26         public CategoryDAO( EntityManager entityManager ) {
27                 super( entityManager, Category.class );
28         }
29
30         /**
31          * <p>全分類情報を取得します。</p>
32          *
33          * @return 全ての分類情報にアクセスするクエリー
34          */
35         public Query getAll() {
36                 return this.getEntityManager().createQuery( "FROM Category" );
37         }
38
39         /**
40          * <p>全分類情報の件数を取得します。</p>
41          *
42          * @return 全分類情報の件数
43          */
44         public long getTotal() {
45                 Query query = this.getEntityManager().createQuery(
46                                 "SELECT COUNT( category ) FROM Category AS category" );
47
48                 return ( Long ) query.getSingleResult();
49         }
50
51         /**
52          * <p>指定の分類名から分類情報を探します。</p>
53          *
54          * @param name 分類名
55          * @return 指定の分類名の分類情報、見つからない場合は null
56          */
57         @SuppressWarnings("unchecked")
58         public Category get( String name ) {
59                 Validate.notNull( name );
60
61                 // 問い合わせ
62                 Query query = this.getEntityManager().createQuery(
```

CategoryDAO.java Page 2

```
63                                 "SELECT category FROM Category AS category WHERE name =
64 :name" ),
65                         setParameter( "name", name ),
66                         setMaxResults( 1 );
67                 // 結果
68                 List< Category > result = query.getResultList();
69
70                 // 情報がなければ null
71                 if ( result.isEmpty() ) {
72                         return null;
73                 }
74
75                 // 初めの情報のみを返す
76                 return result.get( 0 );
77         }
78
79         /**
80          * <p>指定の ID から分類情報を探します。</p>
81          *
82          * @param id ID
83          * @return 指定の ID の分類情報、見つからない場合は null
84          * @throws IllegalArgumentException 引数が null の場合
85          */
86         @SuppressWarnings("unchecked")
87         public Category get( Integer id ) {
88                 return this.getEntityManager().find( Category.class, id );
89
90         }
91 }
```

DocumentMetadataDAO.java Page 1

```
 1 package jp.ac.u_toyama.ite.dokb.metadata.dao;
 2
 3 import javax.persistence.EntityManager;
 4 import javax.persistence.Query;
 5
 6 import jp.ac.u_toyama.ite.dokb.metadata.data.DocumentMetadata;
 7
 8 /**
 9  * <p>永続化機構中の文書メタデータにアクセスする DAO です。</p>
10  * <p>このクラスはスレッドセーフではありません。</p>
11  *
12  * @see DocumentMetadata
13  *
14  * @author Keita Kita (2007/10/05 -)
15  * @since 1.1-SNAPSHOT
16  */
17 public class DocumentMetadataDAO extends
18                 AbstractDAO< DocumentMetadata, Integer > {
19         /**
20          * <p>コンストラクタです。</p>
21          *
22          * @param entityManager EntityManager
23          * @throws IllegalArgumentException 引数が null の場合
24          */
25         public DocumentMetadataDAO(EntityManager entityManager ) {
26                 super( entityManager, DocumentMetadata.class );
27         }
28
29         /**
30          * <p>全文書メタデータを取得するクエリーを生成します。</p>
31          *
32          * @return 全文書メタデータを取得するクエリー
33          * @throws IllegalStateException EntityManager が終了している場合
34          */
35         public Query getAll() {
36                 return this.getEntityManager().createQuery( "FROM DocumentMetada
37 ta" );
38         }
39
40         /**
41          * <p>全文書メタデータの件数を取得します。</p>
42          *
43          * @return 全文書メタデータを取得するクエリー
44          * @throws IllegalStateException EntityManager が終了している場合
45          */
46         public long getTotal() {
47                 Query query = this.getEntityManager().createQuery(
48                                 "SELECT COUNT( dm ) FROM DocumentMetadata AS dm" );
49
50                 return ( Long ) query.getSingleResult();
51         }
52 }
```

LibraryDAO.java Page 1

```
package jp.ac.u_toyama.ite.dokh.metadata.dao;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.Query;

import org.apache.commons.lang.Validate;

import jp.ac.u_toyama.ite.dokh.data.library;

/**
 * <p>本統轄機構中の図書館情報にアクセスするための DAO です。</p>
 * <p>このクラスはスレッドセーフではありません。</p>
 *
 * @author Keita Kita (2007/12/10 -)
 * @since 1.1-SNAPSHOT
 */
public class LibraryDAO extends AbstractDAO< Library, Integer > {
        /**
         * <p>コンストラクタです。</p>
         *
         * @param entityManager EntityManager
         * @throws IllegalArgumentException 引数が null の場合
         */
        public LibraryDAO( EntityManager entityManager ) {
                super( entityManager, Library.class );
        }

        /**
         * <p>全図書館情報にアクセスするクエリーを取得します。</p>
         *
         * @return 全図書館情報にアクセスするクエリー
         * @throws IllegalStateException EntityManager が閉じられている場合
         */
        public Query getAll() {
                return this.getEntityManager().createQuery( "FROM Library" );
        }

        /**
         * <p>全図書館情報の数を返します。</p>
         *
         * @return 全図書館情報の数
         * @throws IllegalStateException EntityManager が閉じられている場合
         */
        public long getTotal() {
                Query query = this.getEntityManager().createQuery(
                        "SELECT COUNT( lib ) FROM Library AS lib" );

                return ( Long ) query.getSingleResult();
        }

        /**
         * <p>指定の図書館名から、図書館情報を探します。</p>
         *
         * @param name 図書館名
         * @return 指定の図書館名の図書館情報、見つからない場合は null
         * @throws IllegalArgumentException 引数が null の場合
         */
        @SuppressWarnings("unchecked")
        public Library get( String name ) {
                Validate.notNull( name );
```

LibraryDAO.java Page 2

```

                // 問い合わせ
                Query query = this.getEntityManager().createQuery(
                        "SELECT lib FROM Library AS lib WHERE name = :name" ).
                        setParameter( "name", name ).
                        setMaxResults( 1 );
                // その結果
                List< Library > results = query.getResultList();

                // 見つからない場合は、null
                if ( results.isEmpty() ) {
                        return null;
                }

                // 初めの結果のみを返す
                return results.get( 0 );
        }

        /**
         * <p>指定の ID から、図書館情報を返します。</p>
         *
         * @param id 図書館の ID
         * @return 指定の ID を持つ図書館情報、見つからない場合は null
         * @throws IllegalArgumentException 引数が null の場合
         */
        public Library get( Integer id ) {
                return this.getEntityManager().find( Library.class, id );
        }
}
```

Appendix.java Page 1

```
package jp.ac.u_toyama.ite.dokh.metadata.data;

import java.io.Serializable;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import org.apache.commons.lang.StringUtils;
import org.apache.commons.lang.builder.HashCodeBuilder;
import org.hibernate.validator.NotNull;

/**
 * <p>雑誌データです。</p>
 * <p>このクラスはスレッドセーフではありません。</p>
 *
 * @author Keita Kita (2007/08/12 -)
 * @since 1.1 SNAPSHOT
 */
@Entity
@Table( name = "appendix_of_document" )
public class Appendix implements Serializable {
        /**
         * <p>直列化の際に使用される識別番号です。</p>
         */
        private static final long serialVersionUID = -5123809585838497018L;

        /**
         * <p>ID です。</p>
         */
        private Integer id;

        /**
         * <p>この附録が含まれる文書情報番号です。</p>
         */
        private DocumentMetadata includedDocument;

        /**
         * <p>書名です。</p>
         */
        private String title = StringUtils.EMPTY;

        /**
         * <p>書名（カナ）です。</p>
         */
        private String titleInKana = StringUtils.EMPTY;

        /**
         * <p>書名（Pinyin）です。</p>
         */
        private String titleInPinyin = StringUtils.EMPTY;

        /**
         * <p>刊年です。</p>
         */
        private String year = StringUtils.EMPTY;
```

Appendix.java Page 2

```

        /**
         * <p>出版主体です。</p>
         */
        private String publisher = StringUtils.EMPTY;

        /**
         * <p>エディションです。</p>
         */
        private String edition = StringUtils.EMPTY;


        /**
         * <p>IDを返します。</p>
         * <p>生成直後の段階では、null です。</p>
         *
         * @return ID
         */
        @Id
        @GeneratedValue( strategy = GenerationType.IDENTITY )
        public Integer getId() {
                return this.id;
        }

        /**
         * <p>IDを設定します。</p>
         *
         * @param id ID
         */
        public void setId( Integer id ) {
                this.id = id;
        }

        /**
         * <p>この附録が含まれる文書メタデータを返します。</p>
         * <p>生成直後の段階では、null です。</p>
         *
         * @return この附録が含まれる文書メタデータ
         */
        @ManyToOne( cascade =
                { CascadeType.MERGE, CascadeType.PERSIST, CascadeType.REFRESH } )
        @JoinColumn( name = "included_document_number", nullable = false )
        @NotNull
        public DocumentMetadata getIncludedDocument() {
                return this.includedDocument;
        }

        /**
         * <p>この附録が含まれる文書メタデータの番号を設定します。</p>
         *
         * @param includedDocument この附録が含まれる文書メタデータの番号
         */
        public void setIncludedDocument( DocumentMetaData includedDocument ) {
                this.includedDocument = includedDocument;
        }

        /**
         * <p>書名を返します。</p>
         *
         * @return 書名
         */
```

Appendix.java Page 3

```
125         @NotNull
126         public String getTitle() {
127                 return this.title;
128         }
129
130         /**
131          * <p>書名を設定します。</p>
132          *
133          * @param title 書名
134          */
135         public void setTitle( String title ) {
136                 this.title = title;
137         }
138
139         /**
140          * <p>書名（カナ）を返します。</p>
141          *
142          * @return 書名（カナ）
143          */
144         @Column( name = "title_in_kana" )
145         @NotNull
146         public String getTitleInKana() {
147                 return this.titleInKana;
148         }
149
150         /**
151          * <p>書名（カナ）を設定します。</p>
152          *
153          * @param titleInKana 書名（カナ）
154          */
155         public void setTitleInKana( String titleInKana ) {
156                 this.titleInKana = titleInKana;
157         }
158
159         /**
160          * <p>書名（Pinyin）を返します。</p>
161          *
162          * @return the titleInPinyin
163          */
164         @Column( name = "title_in_pinyin" )
165         @NotNull
166         public String getTitleInPinyin() {
167                 return this.titleInPinyin;
168         }
169
170         /**
171          * <p>書名（Pinyin）を設定します。</p>
172          *
173          * @param titleInPinyin 書名（Pinyin）
174          */
175         public void setTitleInPinyin( String titleInPinyin ) {
176                 this.titleInPinyin = titleInPinyin;
177         }
178
179         /**
180          * <p>刊年を返します。</p>
181          *
182          * @return 刊年
183          */
184         @NotNull
185         public String getYear() {
186                 return this.year;
```

Appendix.java Page 4

```
187         }
188
189         /**
190          * <p>刊年を設定します。</p>
191          *
192          * @param year 刊年
193          */
194         public void setYear( String year ) {
195                 this.year = year;
196         }
197
198         /**
199          * <p>出版主体を返します。</p>
200          *
201          * @return 出版主体
202          */
203         @NotNull
204         public String getPublisher() {
205                 return this.publisher;
206         }
207
208         /**
209          * <p>出版主体を設定します。</p>
210          *
211          * @param publisher 出版主体
212          */
213         public void setPublisher( String publisher ) {
214                 this.publisher = publisher;
215         }
216
217         /**
218          * <p>エディションを返します。</p>
219          *
220          * @return エディション
221          */
222         @NotNull
223         public String getEdition() {
224                 return this.edition;
225         }
226
227         /**
228          * <p>エディションを設定します。</p>
229          *
230          * @param edition エディション
231          */
232         public void setEdition( String edition ) {
233                 this.edition = edition;
234         }
235
236         @Override
237         public boolean equals( Object otherObject ) {
238                 if ( this == otherObject ) {
239                         return true;
240                 }
241                 if ( !( otherObject instanceof Appendix ) ) {
242                         return false;
243                 }
244
245                 Appendix otherAppendix = ( Appendix ) otherObject;
246
247                 // ID がなければ、全て違うデータとする
248                 if ( this.id == null || otherAppendix.getId() == null ) {
```

Appendix.java Page 5

```
249                         return false;
250                 }
251
252                 return this.id.equals( otherAppendix.getId() );
253         }
254
255         @Override
256         public int hashCode() {
257                 return ( new HashCodeBuilder().
258                         append( this.id ).
259                         append( this.includedDocument ).
260                         append( this.title ).
261                         toHashCode() );
262         }
263 }
```

Author.java Page 1

```
1  package jp.ac.u_toyama.ite.dokh.metadata.data;
2
3  import java.io.Serializable;
4
5  import javax.persistence.CascadeType;
6  import javax.persistence.Column;
7  import javax.persistence.Entity;
8  import javax.persistence.GeneratedValue;
9  import javax.persistence.GenerationType;
10 import javax.persistence.Id;
11 import javax.persistence.JoinColumn;
12 import javax.persistence.ManyToOne;
13 import javax.persistence.Table;
14
15 import org.apache.commons.lang.StringUtils;
16 import org.apache.commons.lang.Validate;
17 import org.apache.commons.lang.builder.EqualsBuilder;
18 import org.apache.commons.lang.builder.HashCodeBuilder;
19 import org.hibernate.validator.NotNull;
20
21
22 /**
23  * <p>著者データです。</p>
24  * <p>このクラスはスレッドセーフではありません。</p>
25  *
26  * @author Keita Kita (2007/08/14 -)
27  * @since 1.1-SNAPSHOT
28  */
29 @Entity
30 @Table( name = "author" )
31 public class Author implements Serializable {
32         /**
33          * <p>直列化の際に使用される識別番号です。</p>
34          */
35         private static final long serialVersionUID = -6636762938198968111L;
36
37         /**
38          * <p>ID です。</p>
39          */
40         private Integer id;
41
42         /**
43          * <p>この著者による文書メタデータです。</p>
44          */
45         private DocumentMetadata documentMetadata;
46
47         /**
48          * <p>著者名です。</p>
49          */
50         private String name = StringUtils.EMPTY;
51
52         /**
53          * <p>著者名の Pinyin 表記です。</p>
54          */
55         private String nameInPinyin = StringUtils.EMPTY;
56
57         /**
58          * <p>著者名のカナ表記です。</p>
59          */
60         private String nameInKana = StringUtils.EMPTY;
61
62
```

Author.java Page 2

```
63          /**
64           * <p>引数なしのコンストラクタです。</p>
65           */
66          public Author() {
67                  // 処理なし
68          }
69
70          /**
71           * <p>著者名を指定できるコンストラクタです。</p>
72           *
73           * @param name 著者名
74           * @throws IllegalArgumentException 引数が null の場合
75           */
76          public Author( String name ) {
77                  Validate.notNull( name );
78
79                  this.name = name;
80          }
81
82          /**
83           * <p>著者 ID を返します。</p>
84           *
85           * @return 著者 ID
86           */
87          @Id
88          @GeneratedValue( strategy = GenerationType.IDENTITY )
89          @Column( name = "id" )
90          public Integer getId() {
91                  return this.id;
92          }
93
94          /**
95           * <p>著者 ID を設定します。</p>
96           *
97           * @param id 著者 ID
98           */
99          public void setId( Integer id ) {
100                 this.id = id;
101         }
102
103
104         /**
105          * <p>この著者による文書メタデータを返します。</p>
106          * <p>年成置後の段階では、null です。</p>
107          *
108          * @return 文書メタデータ
109          */
110         @ManyToOne( cascade =
111                 { CascadeType.MERGE, CascadeType.PERSIST, CascadeType.REFRESH } )
112 )
113         @JoinColumn( name = "document_number", nullable = false )
114         public DocumentMetadata getIncludedDocumentMetadata() {
115                 return this.documentMetadata;
116         }
117
118         /**
119          * <p>この著者による文書メタデータを設定します。</p>
120          *
121          * @param data 文書メタデータ
122          */
123         public void setIncludedDocumentMetadata( DocumentMetadata data ) {
124
```

Author.java Page 3

```
125                 this.documentMetadata = data;
126         }
127
128         /**
129          * <p>著者名を返します。</p>
130          *
131          * @return 著者名
132          */
133         @NotNull
134         @Column( name = "name" )
135         public String getName() {
136                 return this.name;
137         }
138
139         /**
140          * <p>著者名を設定します。</p>
141          *
142          * @param name 編者名
143          */
144         public void setName( String name ) {
145                 this.name = name;
146         }
147
148         /**
149          * <p>著者名の Pinyin 表記を返します。</p>
150          *
151          * @return 著者名の Pinyin 表記
152          */
153         @Column( name = "name_in_pinyin" )
154         @NotNull
155         public String getNameInPinyin() {
156                 return this.nameInPinyin;
157         }
158
159         /**
160          * <p>著者名の Pinyin 表記を設定します。</p>
161          *
162          * @param nameInPinyin 著者名の Pinyin 表記
163          */
164         public void setNameInPinyin( String nameInPinyin ) {
165                 this.nameInPinyin = nameInPinyin;
166         }
167
168         /**
169          * <p>著者名のカナ表記を返します。</p>
170          *
171          * @return 著者名のカナ表記
172          */
173         @Column( name = "name_in_kana" )
174         @NotNull
175         public String getNameInKana() {
176                 return this.nameInKana;
177         }
178
179         /**
180          * <p>著者名のカナ表記を設定します。</p>
181          *
182          * @param nameInKana 著者名のカナ表記
183          */
184         public void setNameInKana( String nameInKana ) {
185                 this.nameInKana = nameInKana;
186         }
```

Author.java Page 4

```
187
188         @Override
189         public boolean equals( Object otherObject ) {
190                 if ( this == otherObject ) {
191                         return true;
192                 }
193                 if ( !( otherObject instanceof Author ) ) {
194                         return false;
195                 }
196
197                 Author otherAuthor = ( Author ) otherObject;
198
199                 Integer thisMetadataNumber = this.documentMetadata != null ?
200                         this.documentMetadata.getNumber() : null;
201                 Integer otherMetadataNumber =
202                         otherAuthor.getIncludedDocumentMetadata() != null ?
203                                 otherAuthor.getIncludedDocumentMetadata().getNur
204 ber() : null;
205
206                 return ( new EqualsBuilder().append(
207                                 thisMetadataNumber, otherMetadataNumber ).
208                         append( this.name, otherAuthor.getName() ) ).isEquals();
209
210         }
211
212         @Override
213         public int hashCode() {
214 //              return ( new AuthorKey( this.documentNumber, this.name ).hashCod
215 e() );
216
217                 return ( new HashCodeBuilder().append( this.documentMetadata ).
218                         append( this.name ) ).toHashCode();
219         }
220 }
```

DocumentMetadata.java Page 1

```
1  package jp.ac.u_toyama.ite.dakb.metadata.data;
2
3  import java.io.Serializable;
4  import java.util.Collections;
5  import java.util.List;
6
7  import javax.persistence.CascadeType;
8  import javax.persistence.Column;
9  import javax.persistence.Entity;
10 import javax.persistence.GeneratedValue;
11 import javax.persistence.GenerationType;
12 import javax.persistence.Id;
13 import javax.persistence.JoinColumn;
14 import javax.persistence.ManyToOne;
15 import javax.persistence.OneToMany;
16 import javax.persistence.Table;
17
18 import jp.ac.u_toyama.ite.dakb.data.Category;
19 import jp.ac.u_toyama.ite.dakb.data.Library;
20
21 import org.apache.commons.lang.ObjectUtils;
22 import org.apache.commons.lang.StringUtils;
23 import org.apache.commons.lang.builder.HashCodeBuilder;
24 import org.hibernate.validator.NotNull;
25
26 /**
27  * <p>文書メタデータです。</p>
28  * <p>このクラスはスレッドセーフではありません。</p>
29  *
30  * @author Keita Kita (2007/08/10 -)
31  * @since 1.1-SNAPSHOT
32  */
33 @Entity
34 @Table( name= "document_metadata" )
35 public class DocumentMetadata implements Serializable {
36         /**
37          * <p>直列化の際に使用される識別番号です。</p>
38          */
39         private static final long serialVersionUID = -4599126536833042936L;
40
41
42         /**
43          * <p>編号です。</p>
44          */
45         private Integer number;
46
47         /**
48          * <p>分類（部）です。</p>
49          */
50         private String fourCategory = StringUtils.EMPTY;
51
52         /**
53          * <p>分類（類）です。</p>
54          */
55         private String classification = StringUtils.EMPTY;
56
57         /**
58          * <p>分類（属）です。</p>
59          */
60         private String tag = StringUtils.EMPTY;
61
62         /**
```

DocumentMetadata.java Page 2

```
63          * <p>分類（H）です。</p>
64          */
65         private String kind = StringUtils.EMPTY;
66
67         /**
68          * <p>分類です。</p>
69          */
70         private Category category;
71
72         /**
73          * <p>書名です。</p>
74          */
75         private String title = StringUtils.EMPTY;
76
77         /**
78          * <p>書名（カナ）です。</p>
79          */
80         private String titleInKana = StringUtils.EMPTY;
81
82         /**
83          * <p>書名（pinyin）です。</p>
84          */
85         private String titleInPinyin = StringUtils.EMPTY;
86
87         /**
88          * <p>「即」書名です。</p>
89          */
90         private String anotherTitle = StringUtils.EMPTY;
91
92         /**
93          * <p>「即」書名（カナ）です。</p>
94          */
95         private String anotherTitleInKana = StringUtils.EMPTY;
96
97         /**
98          * <p>「即」書名（pinyin）です。</p>
99          */
100        private String anotherTitleInPinyin = StringUtils.EMPTY;
101
102        /**
103         * <p>著者です。</p>
104         */
105        private List< Author > authors = Collections.emptyList();
106
107        /**
108         * <p>附録です。</p>
109         */
110        private List< Appendix > appendixes = Collections.emptyList();
111
112        /**
113         * <p>刊年です。</p>
114         */
115        private String year = StringUtils.EMPTY;
116
117        /**
118         * <p>出版主体です。</p>
119         */
120        private String publisher = StringUtils.EMPTY;
121
122        /**
123         * <p>エディションです。</p>
124         */
```

DocumentMetadata.java Page 3

```
125        private String edition = StringUtils.EMPTY;
126
127        /**
128         * <p>蔵版です。</p>
129         */
130        private String engravingOut = StringUtils.EMPTY;
131
132        /**
133         * <p>抄刻附記です。</p>
134         */
135        private String transcript = StringUtils.EMPTY;
136
137        /**
138         * <p>叢書名です。</p>
139         */
140        private String collection = StringUtils.EMPTY;
141
142        /**
143         * <p>所蔵機関です。</p>
144         */
145        private Library library;
146
147        /**
148         * <p>登録番号です。</p>
149         */
150        private String registerNumberInLibrary = StringUtils.EMPTY;
151
152        /**
153         * <p>請求番号です。</p>
154         */
155        private String requestNumberInLibrary = StringUtils.EMPTY;
156
157        /**
158         * <p>文庫 名称です。</p>
159         */
160        private String nameInLibrary = StringUtils.EMPTY;
161
162        /**
163         * <p>文庫 コードです。</p>
164         */
165        private String codeInLibrary = StringUtils.EMPTY;
166
167        /**
168         * <p>冊数です。</p>
169         */
170        private String volume = StringUtils.EMPTY;
171
172        /**
173         * <p>注記です。</p>
174         */
175        private String note = StringUtils.EMPTY;
176
177
178        /**
179         * <p>番号を返します。</p>
180         *
181         * @return 番号
182         */
183        @Id
184        @GeneratedValue( strategy = GenerationType.IDENTITY )
185        public Integer getNumber() {
186                return this.number;
```

DocumentMetadata.java Page 4

```
187        }
188
189        /**
190         * <p>番号を設定します。</p>
191         *
192         * @param number 設定する番号
193         */
194        public void setNumber( Integer number ) {
195                this.number = number;
196        }
197
198        /**
199         * <p>分類（部）を返します。</p>
200         *
201         * @return 分類（部）
202         */
203        @Column( name = "four_category" )
204        @NotNull
205        public String getFourCategory() {
206                return this.fourCategory;
207        }
208
209        /**
210         * <p>分類（部）を設定します
211         *
212         * @param fourCategory 分類（部）
213         */
214        public void setFourCategory( String fourCategory ) {
215                this.fourCategory = fourCategory;
216        }
217
218        /**
219         * <p>分類（類）を返します。</p>
220         *
221         * @return 分類（類）
222         */
223        @NotNull
224        public String getClassification() {
225                return this.classification;
226        }
227
228        /**
229         * <p>分類（類）を設定します。</p>
230         *
231         * @param classification 分類（類）
232         */
233        public void setClassification( String classification ) {
234                this.classification = classification;
235        }
236
237        /**
238         * <p>分類（属）を返します。</p>
239         *
240         * @return the 分類（属）
241         */
242        @NotNull
243        public String getTag() {
244                return this.tag;
245        }
246
247        /**
248         * <p>分類（属）を設定します。</p>
```

DocumentMetadata.java Page 5

```
249         *
250         * @param tag 分類（属）
251         */
252        public void setTag( String tag ) {
253                this.tag = tag;
254        }
255
256        /**
257         * <p>分類（目）を返します。</p>
258         *
259         * @return 分類（目）
260         */
261        @NotNull
262        public String getKind() {
263                return this.kind;
264        }
265
266        /**
267         * <p>分類（目）を設定します。</p>
268         *
269         * @param kind 分類（目）
270         */
271        public void setKind( String kind ) {
272                this.kind = kind;
273        }
274
275        /**
276         * <p>分類を返します。</p>
277         * <p>生成直後の状態では null が返されます。</p>
278         *
279         * @return 分類
280         */
281        @ManyToOne( cascade = { CascadeType.MERGE, CascadeType.REFRESH } )
282        @JoinColumn( name = "category_code", referencedColumnName = "code" )
283        @NotNull
284        public Category getCategory() {
285                return this.category;
286        }
287
288        /**
289         * <p>分類を設定します。</p>
290         *
291         * @param category 分類
292         */
293        public void setCategory( Category category ) {
294                this.category = category;
295        }
296
297        /**
298         * <p>書名を返します。</p>
299         *
300         * @return 書名
301         */
302        public String getTitle() {
303                return this.title;
304        }
305
306        /**
307         * <p>書名を設定します。</p>
308         *
309         * @param title 書名
310         */
```

DocumentMetadata.java Page 6

```
311         public void setTitle( String title ) {
312                 this.title = title;
313         }
314
315         /**
316          * <p>書名（カナ）を返します。</p>
317          *
318          * @return 書名（カナ）
319          */
320         @Column( name = "title_in_kana" )
321         @NotNull
322         public String getTitleInKana() {
323                 return this.titleInKana;
324         }
325
326         /**
327          * <p>書名（カナ）を設定します。</p>
328          *
329          * @param titleInKana 書名（カナ）
330          */
331         public void setTitleInKana( String titleInKana ) {
332                 this.titleInKana = titleInKana;
333         }
334
335         /**
336          * <p>書名（Pinyin）を返します。</p>
337          *
338          * @return 書名（カナ）
339          */
340         @Column( name = "title_in_pinyin" )
341         @NotNull
342         public String getTitleInPinyin() {
343                 return this.titleInPinyin;
344         }
345
346         /**
347          * <p>書名（Pinyin）を設定します。</p>
348          *
349          * @param titleInPinyin 書名（Pinyin）
350          */
351         public void setTitleInPinyin( String titleInPinyin ) {
352                 this.titleInPinyin = titleInPinyin;
353         }
354
355         /**
356          * <p>「即」書名を返します。</p>
357          *
358          * @return 「即」書名
359          */
360         @Column( name = "another_title" )
361         @NotNull
362         public String getAnotherTitle() {
363                 return this.anotherTitle;
364         }
365
366         /**
367          * <p>「即」書名を設定します。</p>
368          *
369          * @param anotherTitle 「即」書名
370          */
371         public void setAnotherTitle( String anotherTitle ) {
372                 this.anotherTitle = anotherTitle;
```

DocumentMetadata.java Page 7

```
373         }
374
375         /**
376          * <p>「即」書名（カナ）を返します。</p>
377          *
378          * @return 「即」書名（カナ）
379          */
380         @Column( name = "another_title_in_kana" )
381         @NotNull
382         public String getAnotherTitleInKana() {
383                 return this.anotherTitleInKana;
384         }
385
386         /**
387          * <p>「即」書名（カナ）を設定します。</p>
388          *
389          * @param anotherTitleInKana 「即」書名（カナ）
390          */
391         public void setAnotherTitleInKana( String anotherTitleInKana ) {
392                 this.anotherTitleInKana = anotherTitleInKana;
393         }
394
395         /**
396          * <p>「即」書名（Pinyin）を返します。</p>
397          *
398          * @return 「即」書名（Pinyin）
399          */
400         @Column( name = "another_title_in_pinyin" )
401         @NotNull
402         public String getAnotherTitleInPinyin() {
403                 return this.anotherTitleInPinyin;
404         }
405
406         /**
407          * <p>「即」書名（Pinyin）を設定します。</p>
408          *
409          * @param anotherTitleInPinyin 「即」書名（Pinyin）
410          */
411         public void setAnotherTitleInPinyin( String anotherTitleInPinyin ) {
412                 this.anotherTitleInPinyin = anotherTitleInPinyin;
413         }
414
415         /**
416          * <p>著者のリストを返します。</p>
417          * <p>生成直後の状態では、null が返されます。</p>
418          *
419          * @return 著者
420          */
421         @NotNull
422         @OneToMany( cascade = CascadeType.ALL,
423                         mappedBy = "includedDocumentMetadata" )
424         public List< Author > getAuthors() {
425                 return this.authors;
426         }
427
428         /**
429          * <p>著者のリストを設定します。</p>
430          *
431          * @param authors 著者
432          */
433         public void setAuthors( List< Author > authors ) {
434                 this.authors = authors;
```

DocumentMetadata.java Page 8

```
435         }
436
437         /**
438          * <p>附録を返します。</p>
439          * <p>生成直後の状態では、null が返されます。</p>
440          *
441          * @return 附録
442          */
443         @OneToMany( cascade = CascadeType.ALL, mappedBy = "includedDocument" )
444         @NotNull
445         public List< Appendix > getAppendixies() {
446                 return this.appendixies;
447         }
448
449         /**
450          * <p>附録を設定します。</p>
451          *
452          * @param appendixies 附録
453          */
454         public void setAppendixies( List< Appendix > appendixies ) {
455                 this.appendixies = appendixies;
456         }
457
458         /**
459          * <p>刊年を返します。</p>
460          *
461          * @return 刊年
462          */
463         @NotNull
464         public String getYear() {
465                 return this.year;
466         }
467
468         /**
469          * <p>刊年を設定します。</p>
470          *
471          * @param year 刊年
472          */
473         public void setYear( String year ) {
474                 this.year = year;
475         }
476
477         /**
478          * <p>出版主体を返します。</p>
479          *
480          * @return 出版主体
481          */
482         @NotNull
483         public String getPublisher() {
484                 return this.publisher;
485         }
486
487         /**
488          * <p>出版主体を設定します。</p>
489          *
490          * @param publisher 出版主体
491          */
492         public void setPublisher( String publisher ) {
493                 this.publisher = publisher;
494         }
495
496         /**
```

DocumentMetadata.java Page 9

```
497          * <p>エディションを返します。</p>
498          *
499          * @return エディション
500          */
501         @NotNull
502         public String getEdition() {
503                 return this.edition;
504         }
505
506         /**
507          * <p>エディションを設定します。</p>
508          *
509          * @param edition エディション
510          */
511         public void setEdition( String edition ) {
512                 this.edition = edition;
513         }
514
515         /**
516          * <p>彫版を返します。</p>
517          *
518          * @return 彫版
519          */
520         @Column( name = "engraving_cut" )
521         @NotNull
522         public String getEngravingCut() {
523                 return this.engravingCut;
524         }
525
526         /**
527          * <p>彫版を設定します。</p>
528          *
529          * @param engravingCut 彫版
530          */
531         public void setEngravingCut( String engravingCut ) {
532                 this.engravingCut = engravingCut;
533         }
534
535         /**
536          * <p>鈔刻附記を返します。</p>
537          *
538          * @return 鈔刻附記
539          */
540         @NotNull
541         public String getTranscript() {
542                 return this.transcript;
543         }
544
545         /**
546          * <p>鈔刻附記を設定します。</p>
547          *
548          * @param transcript 鈔刻附記
549          */
550         public void setTranscript( String transcript ) {
551                 this.transcript = transcript;
552         }
553
554         /**
555          * <p>奥書名を返します。</p>
556          *
557          * @return 奥書名
558          */
```

DocumentMetadata.java Page 10

```
559         @NotNull
560         public String getCollection() {
561                 return this.collection;
562         }
563
564         /**
565          * <p>叢書名を設定します。</p>
566          *
567          * @param collection 叢書名
568          */
569         public void setCollection( String collection ) {
570                 this.collection = collection;
571         }
572
573         /**
574          * <p>所蔵者を返します。</p>
575          * <p>生成直後の状態では、null が返されます。</p>
576          *
577          * @return 所蔵者
578          */
579         @ManyToOne( cascade = { CascadeType.MERGE, CascadeType.REFRESH } )
580         @JoinColumn( name = "library_code", referencedColumnName = "code" )
581         @NotNull
582         public Library getLibrary() {
583                 return this.library;
584         }
585
586         /**
587          * <p>所蔵者を設定します。</p>
588          *
589          * @param library 所蔵者
590          */
591         public void setLibrary( Library library ) {
592                 this.library = library;
593         }
594
595         /**
596          * <p>登録番号を返します。</p>
597          *
598          * @return 登録番号
599          */
600         @Column( name = "register_number_in_library" )
601         @NotNull
602         public String getRegisterNumberInLibrary() {
603                 return this.registerNumberInLibrary;
604         }
605
606         /**
607          * <p>登録番号を設定します。</p>
608          *
609          * @param registerNumberInLibrary 登録番号
610          */
611         public void setRegisterNumberInLibrary( String registerNumberInLibrary )
612         {
613                 this.registerNumberInLibrary = registerNumberInLibrary;
614         }
615
616         /**
617          * <p>請求番号を返します。</p>
618          *
619          * @return 請求番号
620          */
```

DocumentMetadata.java Page 11

```
621         @Column( name = "request_number_in_library" )
622         @NotNull
623         public String getRequestNumberInLibrary() {
624                 return this.requestNumberInLibrary;
625         }
626
627         /**
628          * <p>請求番号を設定します。</p>
629          *
630          * @param requestNumberInLibrary 請求番号
631          */
632         public void setRequestNumberInLibrary( String requestNumberInLibrary ) {
633
634                 this.requestNumberInLibrary = requestNumberInLibrary;
635         }
636
637         /**
638          * <p>文庫 名称を返します。</p>
639          *
640          * @return 文庫 名称
641          */
642         @Column( name = "name_in_library" )
643         @NotNull
644         public String getNameInLibrary() {
645                 return this.nameInLibrary;
646         }
647
648         /**
649          * <p>文庫 名称を設定します。</p>
650          *
651          * @param nameInLibrary 文庫 名称
652          */
653         public void setNameInLibrary( String nameInLibrary ) {
654                 this.nameInLibrary = nameInLibrary;
655         }
656
657         /**
658          * <p>文庫 コードを返します。</p>
659          *
660          * @return 文庫 コード
661          */
662         @Column( name = "code_in_library" )
663         @NotNull
664         public String getCodeInLibrary() {
665                 return this.codeInLibrary;
666         }
667
668         /**
669          * <p>文庫 コードを設定します。</p>
670          *
671          * @param codeInLibrary 文庫 コード
672          */
673         public void setCodeInLibrary( String codeInLibrary ) {
674                 this.codeInLibrary = codeInLibrary;
675         }
676
677         /**
678          * <p>冊数を返します。</p>
679          *
680          * @return 冊数
681          */
682         @NotNull
```

DocumentMetadata.java Page 12

```
683         public String getVolume() {
684                 return this.volume;
685         }
686
687         /**
688          * <p>冊数を設定します。</p>
689          *
690          * @param volume 冊数
691          */
692         public void setVolume( String volume ) {
693                 this.volume = volume;
694         }
695
696         /**
697          * <p>注記を返します。</p>
698          *
699          * @return 注記
700          */
701         @NotNull
702         public String getNote() {
703                 return this.note;
704         }
705
706         /**
707          * <p>注記を設定します。</p>
708          *
709          * @param note 注記
710          */
711         public void setNote( String note ) {
712                 this.note = note;
713         }
714
715         @Override
716         public boolean equals( Object otherData ) {
717                 if ( this == otherData ) {
718                         return true;
719                 }
720                 if ( !( otherData instanceof DocumentMetadata ) ) {
721                         return false;
722                 }
723
724                 DocumentMetadata otherMetadata = ( DocumentMetadata ) otherData;
725
726
727                 // メタデータ番号がない場合は、違うオブジェクトとする
728                 if ( this.number == null || otherMetadata.getNumber() == null )
729                 {
730                         return false;
731                 }
732
733                 return ObjectUtils.equals( this.number, otherMetadata.getNumber(
734         ) );
735         }
736
737         @Override
738         public int hashCode() {
739                 return ( new HashCodeBuilder().
740                         append( this.number ).
741                         append( this.title ).
742                         toHashCode() );
743         }
744 }
```

DocumentMetadata.java Page 13

DocumentMetadataReferenceBean.java Page 1

```java
package jp.ac.u_toyama.ito.dokb.service.reference.impl;

import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;

import jp.ac.u_toyama.ito.dokb.metadata.dao.DocumentMetadataDAO;
import jp.ac.u_toyama.ito.dokb.metadata.data.DocumentMetadata;
import jp.ac.u_toyama.ito.dokb.service.reference.DocumentMetadataReference;
import jp.ac.u_toyama.ito.dokb.service.reference.Limit;
import jp.ac.u_toyama.ito.dokb.service.reference.ReferenceView;

/**
 * <p>DocumentMetadataReference を実装します。</p>
 *
 * @author Keita Kita (2007/10/19 -)
 * @since 1.0-SNAPSHOT
 */
@Stateless
public class DocumentMetadataReferenceBean
        implements DocumentMetadataReference {
    /**
     * <p>EntityManager です。</p>
     */
    @PersistenceContext( unitName = "dokb-db_JarAdokbManager" )
    private EntityManager manager;


    @SuppressWarnings( "unchecked" )
    public ReferenceView< DocumentMetadata > getAll( int index, Limit limit
) {
        // DAO
        DocumentMetadataDAO dao = this.createDAO();
        // クエリー
        Query query = dao.getAll();

        query.setMaxResults( limit.getLimit() );
        query.setFirstResult( limit.getStartPosition( index ) );

        return new ReferenceView< DocumentMetadata >(
                query.getResultList(), index, ( int ) dao.getTotal(), li
mit );
    }

    public long getTotal() {
        return this.createDAO().getTotal();
    }

    /**
     * <p>DocumentMetadataDAO を生成します。</p>
     *
     * @return 生成された DocumentMetadataDAO
     */
    private DocumentMetadataDAO createDAO() {
        return new DocumentMetadataDAO( this.manager );
    }
}
```

DocumentMetadataReference.java Page 1

```java
package jp.ac.u_toyama.ito.dokb.service.reference;

import javax.ejb.Remote;

import jp.ac.u_toyama.ito.dokb.metadata.data.DocumentMetadata;

/**
 * <p>文書メタデータを閲覧するためのビジネスインタフェースです。</p>
 * <p>このクラスはスレッドセーフではありません。</p>
 *
 * @author Keita Kita (2007/10/11 -)
 * @since 1.0-SNAPSHOT
 */
@Remote
public interface DocumentMetadataReference {
    /**
     * <p>文書メタデータを全て閲覧します。</p>
     * <p>最大ページ数以上のページ番号を指定した場合は、空のページが
     * 返されます。</p>
     *
     * @param index ページ番号 (0から始まる)
     * @param limit 1ページ内の最大件数
     * @return 閲覧情報
     * @throws IllegalArgumentExceptIon 引数が負の数の場合
     */
    ReferenceView< DocumentMetadata > getAll( int index, Limit limit );

    /**
     * <p>文書メタデータの全件数を取得します。</p>
     *
     * @return 全件数
     */
    long getTotal();
}
```

Limit.java Page 1

```java
package jp.ac.u_toyama.ito.dokb.service.reference;

import java.io.Serializable;

import org.apache.commons.lang.Validate;

/**
 * <p>1ページ内の最大件数を表します。</p>
 * <p>このクラスは不変オブジェクトのため、スレッドセーフです。</p>
 *
 * @author Keita Kita (2007/10/16 -)
 * @since 1.0-SNAPSHOT
 */
public class Limit implements Serializable {
    /**
     * <p>直列化の際に参照される識別番号です。</p>
     */
    private static final long serialVersionUID = 1349641446033452074L;

    /**
     * <p>1ページ内の最大件数です。</p>
     */
    private final int limit;


    /**
     * <p>コンストラクタです。</p>
     *
     * @param limit 1ページ内の最大件数
     * @throws IllegalArgumentException 引数が0以下の場合
     */
    public Limit( int limit ) {
        Validate.isTrue( limit > 0 );

        this.limit = limit;
    }

    /**
     * <p>1ページ内の最大件数を返します。</p>
     *
     * @return 1ページ内の最大件数
     */
    public int getLimit() {
        return this.limit;
    }

    /**
     * <p>全ページ数を計算します。</p>
     *
     * @param total 全件数
     * @return 全ページ数
     */
    public int getPages( int total ) {
        // 全ページ数
        int pages = total / this.limit;

        // 表示件数が端数のページがある場合
        if ( total % this.limit > 0 ) {
            return pages + 1;
        }

        return pages;
```

Limit.java Page 2

```java
    }

    /**
     * <p>結果を返し始める場所を計算します。</p>
     *
     * @param pageIndex ページ番号 (0から始まる)
     * @return 結果を返し始める位置
     * @throws IllegalArgumentException 引数が負の数の場合
     */
    public int getStartPosition( int pageIndex ) {
        Validate.isTrue( 0 <= pageIndex );

        return this.limit * pageIndex;
    }

    /**
     * <p>行番号からページ番号を計算します。</p>
     *
     * @param rowIndex 行番号
     * @return ページ番号
     * @throws IllegalArgumentException 引数が負の数の場合
     */
    public int getPageIndex( int rowIndex ) {
        Validate.isTrue( 0 <= rowIndex );

        return ( rowIndex / this.limit );
    }
}
```

ReferenceView.java Page 1

```java
package jp.ac.u_toyama.itc.dokb.service.reference;

import java.io.Serializable;
import java.util.Collections;
import java.util.List;

import org.apache.commons.lang.Validate;
import org.apache.commons.lang.math.IntRange;
import org.apache.commons.lang.math.Range;

/**
 * <p>閲覧情報を表します。</p>
 * <p>不変オブジェクトのため、このクラスはスレッドセーフです。</p>
 *
 * @param < T > 閲覧対象のデータ型
 *
 * @author Keita Kita (2007/10/18 -)
 * @since 1.0-SNAPSHOT
 */
public class ReferenceView< T extends Serializable > implements Serializable {
        /**
         * <p>直列化の際に参照される識別番号です。</p>
         */
        private static final long serialVersionUID = 2144133206678483349L;

        /**
         * <p>閲覧対象のデータのリストです。</p>
         */
        private final List< T > list;

        /**
         * <p>全件数です。</p>
         */
        private final int total;

        /**
         * <p>全ページ数です。</p>
         */
        private final int pages;

        /**
         * <p>対象となる行番号の範囲です。</p>
         */
        private final Range rowRange;


        /**
         * <p>コンストラクタです。</p>
         *
         * @param list 閲覧対象のデータのリスト
         * @param startRowIndex リストの初めの要素が格納されている行番号
         * @param total 全件数
         * @param limit 1ページ内の最大件数
         * @throws IllegalArgumentException 引数が null の場合、
         *      startRowIndex, total が0未満の場合
         */
        public ReferenceView( List< T > list, int startRowIndex,
                        int total, Limit limit ) {
                Validate.notNull( list );
                Validate.isTrue( startRowIndex >= 0 );
                Validate.isTrue( total >= 0 );
                Validate.notNull( limit );
```

ReferenceView.java Page 2

```java

                this.list = Collections.unmodifiableList( list );
                this.total = total;
                this.pages = limit.getPages( total );
                this.rowRange = calculateRowRange( startRowIndex, total, limit )
;
        }

        /**
         * <p>この閲覧情報が表す行番号の範囲を計算します。</p>
         *
         * @param startRowIndex 初めの行番号
         * @param total 全件数
         * @param limit 1ページ内の最大件数
         * @return この閲覧情報が表す行番号の範囲
         */
        private static Range calculateRowRange(
                        int startRowIndex, int total, Limit limit ) {
                assert ( startRowIndex >= 0 );
                assert ( limit != null );

                // 格納できる最大件数
                int maxResults = total - startRowIndex;

                if ( maxResults < limit.getLimit() ) {
                        return new IntRange( startRowIndex, total - 1 );
                }

                return new IntRange(
                        startRowIndex, startRowIndex + limit.getLimit() - 1 );
        }

        /**
         * <p>閲覧対象のデータのリストを返します。</p>
         *
         * @return 閲覧対象のデータのリスト
         */
        public List< T > getList() {
                return this.list;
        }

        /**
         * <p>全件数を返します。</p>
         *
         * @return 全件数
         */
        public int getTotal() {
                return this.total;
        }

        /**
         * <p>全ページ数を返します。</p>
         *
         * @return 全ページ数
         */
        public int getPages() {
                return this.pages;
        }

        /**
         * <p>この閲覧情報が表す行番号の範囲を返します。</p>
         *
```

ReferenceView.java Page 3

```java
         * @return この閲覧情報が表す行番号の範囲
         */
        public Range getRowRange() {
                return this.rowRange;
        }

        /**
         * <p>指定の行番号から、リスト内に対応するデータを返します。</p>
         *
         * @param rowIndex 行番号
         * @return リスト内に対応するインデックス
         * @throws IndexOutOfBoundsException 行番号がリストの範囲外となる場合
         */
        public T getRow( int rowIndex ) {
                return this.list.get( rowIndex - this.rowRange.getMinimumInteger
() );
        }
}
```

DocumentMetadataStorageBean.java Page 1

```java
package jp.ac.u_toyama.itc.dokb.service.storage.impl;

import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

import jp.ac.u_toyama.itc.dokb.metadata.dao.DocumentMetadataDAO;
import jp.ac.u_toyama.itc.dokb.metadata.data.DocumentMetadata;
import jp.ac.u_toyama.itc.dokb.service.storage.DocumentMetadataStorage;

/**
 * <p>DocumentMetadataStorage を実装します。</p>
 *
 * @see DocumentMetadataStorage
 *
 * @author Keita Kita (2007/12/07 -)
 * @since 1.0-SNAPSHOT
 */
@Stateless
public class DocumentMetadataStorageBean implements DocumentMetadataStorage {
        /**
         * <p>EntityManager です。</p>
         */
        @PersistenceContext( unitName = "dokb-db.jar#dokbManager" )
        private EntityManager manager;


        public void add( DocumentMetadata data ) {
                this.getDAO().create( data );
        }

        public void delete( DocumentMetadata data ) {
                this.getDAO().delete( data );
        }

        public void update( DocumentMetadata data ) {
                this.getDAO().update( data );
        }

        /**
         * <p>DocumentMetadataDAO を取得します。</p>
         *
         * @return DocumentMetadataDAO
         */
        private DocumentMetadataDAO getDAO() {
                return new DocumentMetadataDAO( this.manager );
        }
}
```

［資料９－４］古文書等印影データベースのデータ入力要領

古文書等印影データベースのデータ入力要領

2003.05.13(TUE)

１．入力項目

（１）印影（画像）－印影は実物大に切り取り，300dpi(300dots/inch)で読み取り，
画像ファイルは jpeg(.jpg)として保存する．
印影には，印文／印記とその読み（索引をとる），所有者とその読み（索引をとる），
説明文をつける．

（２）印影を含む絵画，本についての情報－品名，員数，国，時代，作者（＝印の所有者），
寸法，材質，所在地（オーナー），所有者（絵画／本のオーナー）を入力する．

２．参照番号（Reference No.）の付け方とその意味

第１桁：分類　１：絵画，２：本（文書）
第２～５桁：　文献通番（一連のユニークな番号）
第６，７桁：　画像の番号
００：全体の画像（スケール入り画像，最大 10cm）
０１～４９：印影（実物大，300dpi＝300dots/inch）
５０～９９：任意の切り出し画像（スケール入り画像，最大 10cm）
第８，９桁：　細分類番号
００：画像
０１：印文／印記　○○○・・・○【Ｓヨミ（索引用）】
×○○・・・○【Ｄヨミ（索引用）】
▽・・・▽▽▽【Ｒヨミ（索引用）】
０２：所有者　○○○・・・○【Ｎヨミ（索引用）】
０３：説明文　□□・・・・□（現代文で記入）
・・・
１１：品名　（例）鷺図
１２：員数　（例）一幅
１３：国　（例）朝鮮／中国／日本
１４：時代　（例）李朝末期
１５：作者　（例）楊基薫　（０２と同じ）
１６：作者生年（例）１８４３－１９１０
１７：寸法　（例）１２８．３×３２．３㎝

1

１８：材質　　（例）紙本墨画
１９：所在地（オーナー）　（例）富山市五福３１９０
２０：所有者（絵画／本のオーナー）　藤本幸夫
２１：注記　　貴重本

3．参照番号（Reference No.）の記入例

（1）絵画，本の全体画像　10001-00-00

1	0	0	0	1	—	0	0	—	0	0

（2）印影－1

1	0	0	0	1	—	0	1	—	0	0

（3）印影－2

1	0	0	0	1	—	0	2	—	0	0

（4）絵画の一部分

1	0	0	0	1	—	5	0	—	0	0

ファイル名＝10001-01-00

10001-02-00

10001-03-00

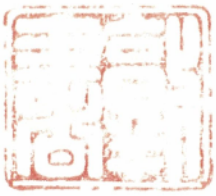
ファイル名＝10001-04-00

ファイル名＝10001-05-00

2

４．テキスト・データの入れ方

（１）例－１　青木昆陽蔵書印



ファイル名＝10001-01-00　　10001-02-00　　10001-03-00

ファイル名＝10001-04-00　　ファイル名＝10001-05-00

桁　・・・・＋・・・・１・・・・＋・・・・２・・・・＋・・・・３・・・・＋・・・・

１０００１－００－××

０１－０１　敦書【Ｓアツノリ】

０１－０２　青木昆陽【Ｎアオキコンヨウ】

０１－０３　甘藷先生の名で知られる昆陽の蔵書にみられる印である．大サイズ．

０２－０１　敦書【Ｓアツノリ】

０２－０２　青木昆陽【Ｎアオキコンヨウ】

０２－０３　甘藷先生の名で知られる昆陽の蔵書にみられる印である．中サイズ．

０３－０１　敦書【Ｓアツノリ】

０３－０２　青木昆陽【Ｎアオキコンヨウ】

０３－０３　甘藷先生の名で知られる昆陽の蔵書にみられる印である．小サイズ．

０４－０１　敦書之印【Ｓアツノリノイン】

０４－０２　青木昆陽【Ｎアオキコンヨウ】

０４－０３　甘藷先生の名で知られる昆陽の蔵書にみられる印である．

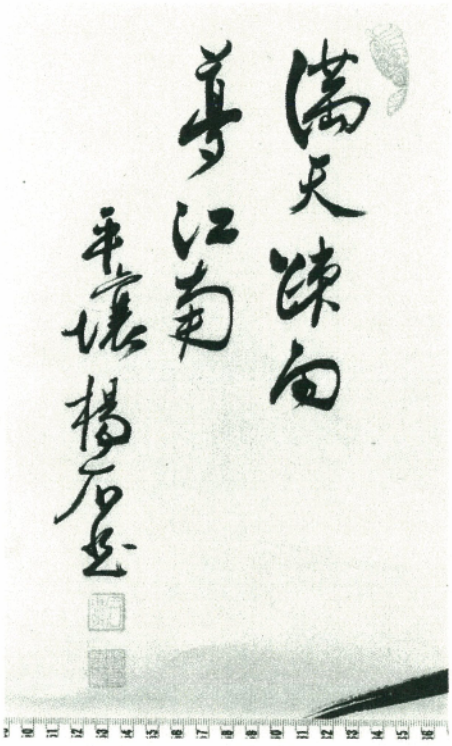
０５－０１　昆陽【Ｓコンヨウ】

０５－０２　青木昆陽【Ｎアオキコンヨウ】

０５－０３　甘藷先生の名で知られる昆陽の蔵書にみられる印である．

（2）例－２　鷺図一幅

ファイル名＝10002-00-00



ファイル名＝10002-50-00

ファイル名＝10002-01-00　　10002-02-00　　10002-03-00

桁　・・・・＋・・・・１・・・・＋・・・・２・・・・＋・・・・３・・・・＋・・・・

１０００２－００－００
００－１１　鷺図
００－１２　一幅
００－１３　朝鮮
００－１４　李朝末期
００－１５　楊基薫
００－１６　１８４３－？
００－１７　１２８．３×３２．３cm
００－１８　紙本墨画
００－１９　四条畷市
００－２０　松井　義
００－２１　コントラストと縮尺を微少変更
０１－０１　四時者【Ｓシジモノ】
０１－０２　楊基薫【Ｎヨウキクン】
０１－０３　楊基薫氏の描いた鷺の絵にみられる印である．
０２－０１　楊基印【Ｓヨウキイン】
０２－０２　楊基薫【Ｎヨウキクン】
０２－０３　楊基薫氏の描いた鷺の絵にみられる印である．
０３－０１　楊基薫印【Ｓヨウキクンノイン】
０３－０２　楊基薫【Ｎヨウキクン】
０３－０３　楊基薫氏の描いた鷺の絵にみられる印である．

５．画像データはハードディスク装置にDOKB_Imageというディレクトリを作成してデータを蓄積していくものとする．

以上．

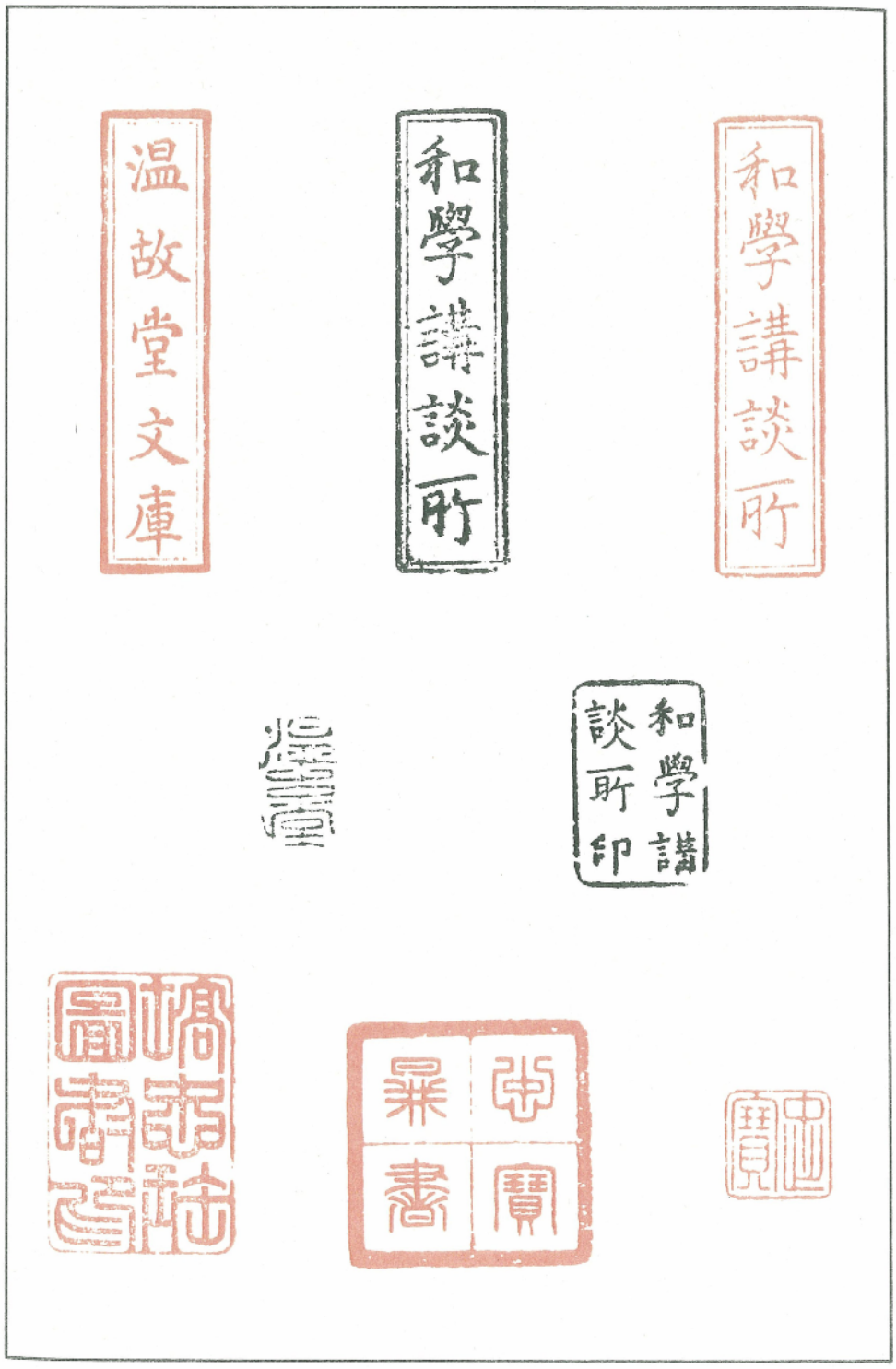
柴野栗山

柴野栗山の蔵書印は、落款類も含め二十種近くが、各種の印譜集に所収されている。その内、当文庫に蔵する旧蔵書四部に見られる蔵書印は「**柴氏家藏圖書**」「**子孫永保**」「**柴邦彦圖書**」「**後阿波國文庫別藏于江戸雀林莊之萬卷樓**」の三種のみである。

「柴氏家藏図書」の印は、四部すべてに押されているが、栗山から佐伯毛利氏に移り、のち紅葉山文庫に献納された明嘉靖二十六年序刊本「南宮奏議」の印は、大きさは他と同大であるが、印文の線が非常に細く、一見異種の如く見える。これは、この印が他の三部に押されるまで、長期に亘って使用されたことを示しているのかも知れない。又、同じく佐伯氏に帰した朝鮮写本「世紀通衢」は、昌平坂学問所を経て、共に当文庫の有となっている。清乾隆三十四年序刊本「義門読書記」は、帙の裏面に紙で網状の袋を作り、この中に煙草の葉を入れる等防虫のための配慮がなされており、栗山の愛書家の一面を今に伝えている。

栗山は、元文元年、讃岐高松に生まれた。名は邦彦、通称は彦助、字は彦輔。号は古愚軒・青悟庵・五峯山房・雙玉楼等。柴邦彦と修す。初め、後藤芝山に、後昌平黌に学び、阿波藩儒となった。東上後は昌平黌の教官となり、幕命を受けて図書の校訂・編集等に関与した。

文化十四年十二月一日没、七十四歳。


和學講談所
和學講談所
温故堂文庫
温故堂
和學講談所印

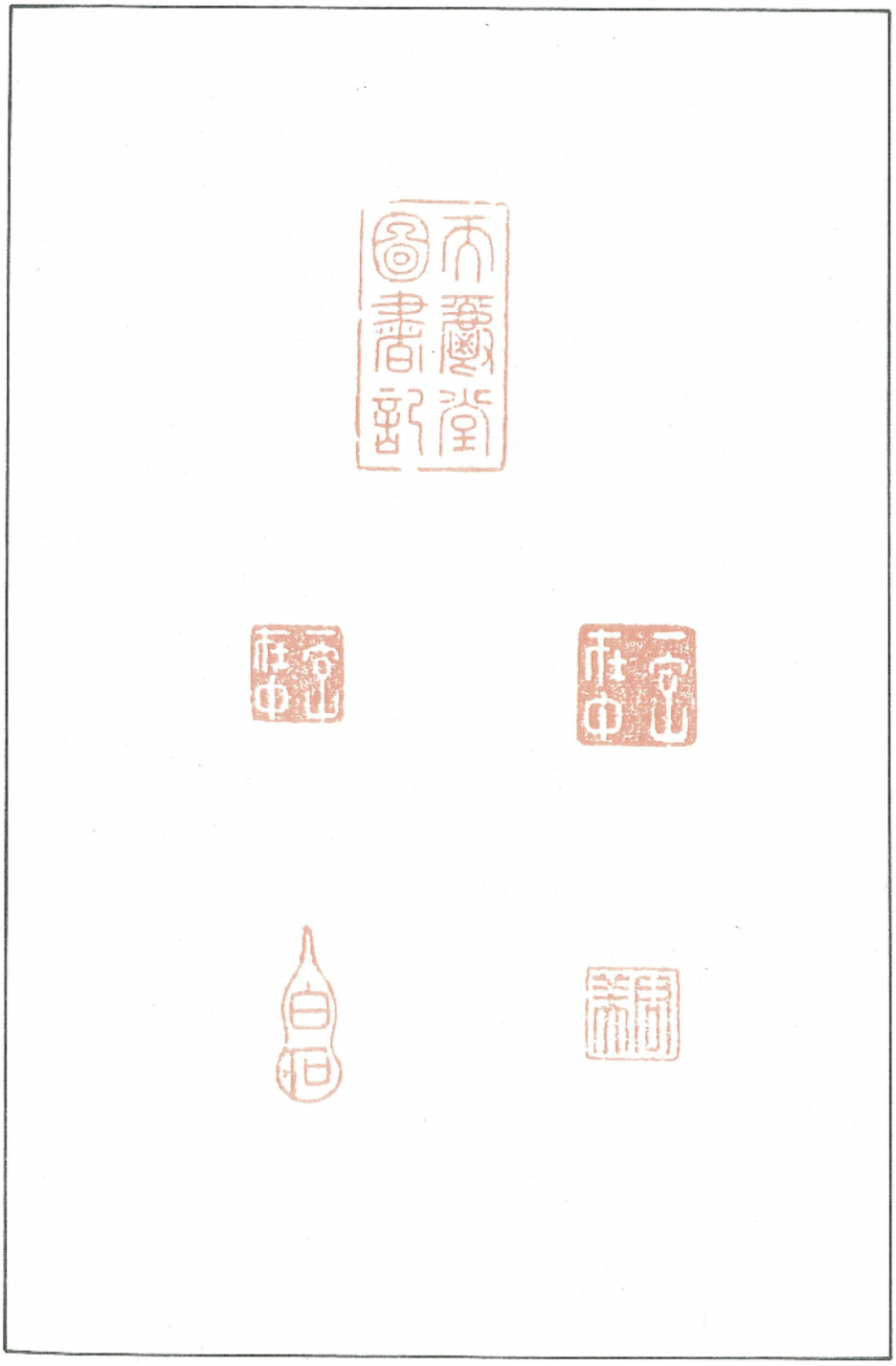
塙家・和学講談所

和学講談所は塙保己一（延享三―文政四年）が寛政五年麹町上六番町に創立した国学研究所である。同七年以後は幕府の財政的援助を受けて、講学、校訂、研究を行ない、その資料として広く古書の収集につとめた。保己一は寛政十一年に公卿日記多数を新写して紅葉山文庫に献じたほか、その後も塙家で編修、刊行したものは漏れなく幕府に進献した。さらに明治五年、保己一の孫塙忠韶から講談所蔵書を書籍館に献納し、これらがいずれもいま当文庫に伝えられている。内閣文庫の蔵書目録の源流欄に「和」と標示したものがすなわちそれである。

「**和學講談所**」印は常に巻首右下部に押されていて、朱墨の別がある。当文庫に所蔵する数は朱印約三百部、黒印約百七十部である。その使い分けの規準がどこにあるか明らかでないが、点数からいえば昌平黌とは反対で、朱印の方が多い。小字二行の「**和學講談所印**」の黒印は珍らしく、わずかに二例を見るのみである。巻首上欄に押してある。なお、和学講談所は和学所と簡称することがあり、世上には、「和学所印」なる蔵書印も存在する。

温故堂は塙保己一の書斎名である。「**温故堂文庫**」印を押したもの約七十部あり、これと「和学講談所」朱印と併用したものも多い。両者の使用上の区別は明らかでない。「温故堂文庫」印は、むしろ明治時代に押したものが多いと聞く。この印の下限は明治十五年八月塙忠韶の奥書をもつ「実豊卿記」にも押してあるから、塙家において忠韶の時代まで使用したものであることがわかる。「**温古堂**」の小形黒印はきわめて例が少なく、また「古」字も異なるので他家のものかと疑われるが、「和学講談所」朱印と併用されているから、やはり塙家の印であろうと思う。

塙忠宝は保己一の四男、通称は次郎。和学講談所御用掛として父の業をついだが、文久二年誤解によって浪士に刺された。その蔵書印「**忠寶**」「**忠寶圖書**」「**塙忠寶圖書印**」は所蔵するものあわせて十数部にすぎないが、すべて和学講談所印と併用されている。忠宝の印は巻末のみに押されている例が多い。



新井白石

八〇

新井白石

寛政五年に白石の後裔成美から、同七年に抱義から、それぞれ白石の自筆本等数種を幕府に献上した。

「**天爵堂圖書記**」「**一字在中**」「**君美**」「**白石**」の各印は、従来の印譜集に白石のものとして著録されており、当文庫においても、自筆本の「東雅」「西洋紀聞」中に見られるものである。小野則秋著「日本の蔵書印」では、「天爵堂図書記」に、二種ある旨の記載があるが、印影未載につき、文庫の印が、そのいずれにあたるか不明である。これと反対に「一字在中」印は、一種のみ知られていたが、掲出のように、大きさの違う二種が確認され、「西洋紀聞」中の「**原君美印**」も従来の印譜集未著録のものである。又、「東雅」の各冊最末丁の裏に押されている「**躬自抄録**」の印は、本書が自筆本であるところから、印文の文意も合い、これも白石の印と考えられるが、断定はできない。掲出の外、「家多賜書」「雲府図書之記」「玉繩」「賜書永宝」「紫陽太守章」「紫陽」「門天俗客」「井璵之印」「名教中自有楽地」等が、白石の印として知られている。

白石、本姓は戸板氏。初め名は璵、後君美と改めた。通称は、与次右衛門、後勘解由、字は済美、又在中。号は、白石の外、錦屏山人・紫陽・天爵堂・勿斎といい、源璵・源(原)君美と称し、新白石・井璵と修した。

木下順庵に師事、その推挙により甲府の徳川綱豊(後、将軍家宣)に仕え、次の家継との間に、いわゆる「正徳の治」を現出し、又、歴史・地理・言語等多方面の活躍をした。享保十年五月十九日没、六十九歳。

青木昆陽

「**敦書**」(三種)「**敦書之印**」「**昆陽**」は、甘藷先生の名で知られる昆陽の蔵書に見られる印である。これらの印は、当文庫に蔵する彼の自筆本八種の内の六種に押されており、オランダ語の訳本二種及び献上本三種には押印はない。

昆陽は、享保二十年から「御書物御用達」の役名を受けて古文書調査にあたり、元文五年から寛保二年までの三年間には、数回にわたり家康ゆかりの武蔵・三河等七国の古文書を採訪し、その主要なものを模写・編集して幕府に献じた。それは「諸州古文書」「判物証文写」と名付けられて当文庫に現蔵されている。外にも将軍吉宗の命を受けてオランダ語を学び、また和漢の古法制を研究する等幅広い活躍をした。それは自筆本類にもよく反映している。

昆陽、名は敦書(あつのり)、通称は文蔵、字は厚甫、青厚甫と修した。明和四年二月、書物奉行に任ぜられ、同六年十月十二日に没した。七十二歳。

研究成果報告書

Ａｊａｘによる日本現存朝鮮古書印影写真
画像データベース検索ツールの研究

課題番号：１８５０００７９
平成１８年度～平成１９年度科学研究費補助金
（ 基盤研究（Ｃ） ）研究成果報告書
平成２０年３月　発行
研究代表者　高　井　正　三
富山大学総合情報基盤センター教授