第４世代ポータルサービスのポートレット開発による機能強化

○田上 奈緒 A)、太田 芳博 B)

A) B) 共通基盤技術支援室　情報通信技術系

概要

名古屋大学ポータルサービスは、オープンソースのポータルフレームワーク uPortal を導入し運用されてきたが、2010 年度 7 月に、バージョン 3.2.1 に更新を行い第 4 世代として稼働開始した。ポータルサービスとは全学に散在する WEB サービス、個別システムへの入り口としての役割、また全学向け連絡手段としての機能を持つ。uPortal 自体は tomcat などのアプリケーションサービス配下の WEB アプリであり、導入するとポータルサービスの外枠として機能する。その内部のコンテンツは、使用技術により XML チャネル、ポートレットなどに分類されるが、名古屋大学独自の機能として、教務システムとの連携や、個人化機能強化のため各種ポートレットの開発を行っている。今回はこのポートレット開発に焦点を当て、現状と今後について報告する。

1　システム構成

現在のポータル動作環境は以下の通りである。

サーバ：IBM BladeCenter HS22

OS：Solaris コンテナ+仮想サーバ OS Solaris10

WEB サービス：Apache httpd-2.2.14

APP サービス：tomcat-6.0.24

データベース：Oracle11g RAC

認証：CAS, LDAP

シングルサインオンを実現する CAS とディレクトリサービスの LDAP は、全学向けに導入運用されているサービスを利用している。図 1 はポータルと他システム、データベースとの連携を表す。

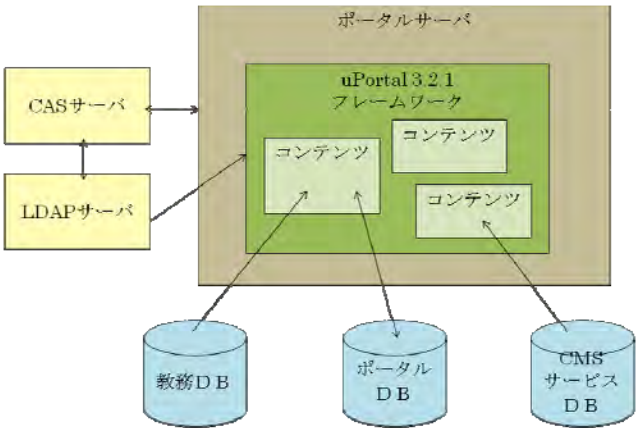

図 1. ポータルと他システムの連携

2　uPortal とポートレットの関係

ポートレットとは､図 2 にあるようにポータル WEB ページ内に複数含まれるコンテンツ枠である。外見・機能的には「iGoogle」のニュース、カレンダ等ガジェットと類似している。ポータルフレームワーク自体は uPortal の他にも多数あるが、その内部のコンテンツを表示するポートレットは JSR168 という規格に基づいて開発することで、他の JSR168 準拠ポータルフレームワーク内でも動作可能になる。名古屋大学でも JSR168 に基づき開発を行っているが、uPortal 以外で

図 2. ポータルログイン後画面

の実証は行っていない。また uPortal の場合、各コンテンツ枠はチャネルと呼ばれるが、このチャネルは作成する技術によって、「XML の内容を XSLT で形成する XML チャネル」「uPortal のインターフェースを implement して組み込むカスタムチャネル」「Java で開発しポートレットコンテナから実行されるポートレット」の３種類に分類される。

3　名古屋大学におけるポートレット開発

これら３種類のチャネルのうち、現名古屋大学ポータルでは「XML チャネル」と「ポートレット」を使用している。XML チャネルは固定化された静的ファイルであり、内容を変更したい場合は、基本的にテキストエディターを用いてファイル内のタグに囲まれた文字列を編集する。比較的変化が少ないリンク集などはこれを使用している。ポートレットは Java で開発を行いカスタマイズが自由なため、データベースから取得した情報をリアルタイムに表示するなど動的な機能を作成する場合に適している。

3.1　開発方法

ポートレット開発に用いるフレームワーク（ライブラリ）は Spring,Struts など多数あるが、その中で Spring を使用し MVC モデルで開発を行っている。C に該当するコントローラは AbstractController クラスを継承、V に該当するビューは JSP で作成する。AbstractController クラス内には handleRenderRequestInternal()メソッド、handleActionRequestInternal()メソッド（以降、前者を Render メソッド、後者を Action メソッドと呼ぶ）を実装しており、ブラウザから入力された GET データは Render メソッドに、POST データは Action メソッドに渡される。

```
●サーブレットの場合
public class TestServlet extends HttpServlet {
   protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
         // GET データの処理
         //（初期表示の出力）
   }
   protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {
         //POST データの処理
         //（ブラウザから入力されたデータの取得）
         //（ＤＢ更新処理）
         //（更新結果の出力）
   }
}
●ポートレットの場合（AbstractContoroller 使用時）
public class TestController extends AbstarctController {
      @Override
      public ModelAndView handleRenderRequestInternal(RenderRequest req, RenderResponse res) {
         //  GET データの処理
         //（初期表示の出力）
         //（更新結果の出力）・・・ここがサーブレットと異なる
      }
      @Override
      protected void handleRenderRequestInternal(RenderRequest req, RenderResponse res) throws Exception {
         //POST データの処理
         //（ブラウザから入力されたデータの取得）
         //（ＤＢ更新処理）
      }
}
```

図 3. サーブレットとポートレットの Controller 部分の比較

サーブレットを作成する際、一般的には図 3 のように HttpServlet クラスを継承し doGet()メソッドをオーバーライドして GET データの処理、doPost()メソッドをオーバーライドして POST データの処理を記述するが、ポートレットの場合も同様に、上記 Render・Action メソッド内にそれぞれ表示系処理・更新系処理を書く。

ただし Action メソッドがコールされた後は、必ず Render メソッドがコールされるため、Action メソッド内でブラウザから入力されたデータを元にＤＢ更新などを行った結果の再表示は Render メソッドで行うようにする。

3.2　作業手順

WEB アプリケーション開発環境と方法も様々であるが、実際に行っている作業手順例を表 1 に示す。

表 1. ポートレット開発手順

	作業内容	作成物
①	maven2（プログラム開発ツール）でプロジェクト作成	開発プロジェクトの雛型ワークスペース（pom.xml, web.xml）
②	pom.xml を編集（外部リポジトリサーバからネットワーク経由で自動的に指定したライブラリをダウンロード）	使用するコンパイラ、ライブラリ、リポジトリなどを指定
③	GUI の統合開発環境 eclipse にワークスペースを取り込み、プログラムの作成	Java ソース、JSP、xml 設定ファイル等
④	mvn コマンドでビルド“mvn package”	パッケージ（war ファイル）
⑤	war ファイルをアプリケーションサーバの webapps の下にコピー	パッケージが展開される
⑥	uPortal 付随のポートレットマネージャでポートレットの取り込み	ポータル内に配置可能

3.3　作成済みポートレット

上記方法で開発したポートレット、プログラム一覧を機能別で表２にリストアップする。

表 2. 作成したプログラム

機能	プログラム	内容	参照先
教務関係	時間割 休講・補講情報 学生呼出し情報	教務情報のリアルタイムな表示	教務 DB
意見収集	モニター アンケート	ユーザの意見収集ツール	独自 DB
運用・管理・コンテンツ編集	XML 編集 アナウンス登録 アナウンス表示	各部局提供コンテンツの編集ツール、必須アナウンスチェック機能	独自 DB LDAP サーバ
コース管理システム関係	受講サイトアナウンス表示	コース管理システムへの誘導	コース管理システム DB
ログイン時の警告	セキュリティ研修警告 セキュリティ自己点検警告	ネットワークセキュリティ確保の一手段である研修の受講促進	LDAP サーバ e-learning システム DB

3.4　教務連携

2010 年 7 月から運用開始した第 4 世代ポータルは個人化機能の強化をうたっており、そのためのカスタマイズを行ってきた。uPortal フレームワーク自体のインストールと環境構築に関しては本稿では述べないが、各種設定ファイルを編集し tomcat リスタートする事で動作環境を変更することが可能である。例えば名古屋大学ではデータベースを内蔵 HSQL から外部の Oracle DBMS に変更している。またユーザ管理も uPortal 独自のローカルユーザを登録可能だが、あらかじめ設置されたディレクトリサービスで組織全体の個人情報を一括管理している場合など、そのアカウントでログインできた方がシングルサインオンなど利用側・運用側ともに利便性が高くなる。本学では全学的に CAS＋LDAP による認証基盤サービスが設けられており、名古屋大学 ID で一元的にユーザ管理されている。そのため uPortal のユーザ管理も LDAP サーバに設定変更し、名古屋大学 ID によるログインが可能になっている。LDAP サーバで保存されている個人情報は、設定ファイルに記述しておけばログイン時に uPortal 内に取り込まれるため、これらの個人情報（名前、身分、所属、メールアドレスなど）をポートレット内で取得して、ログインユーザ個人に特化した情報を表示する機能を追加している。その一例として教務連携の時間割チャネルについて概要を記述する。教務システム側にも学生時間割の WEB 表示はあるが、ポータル利用促進の一環として教員時間割や履修学生への参考サイト連絡機能も付随した時間割チャネルにした。

時間割の元データは教務データベースに保管されている授業担当情報、履修情報などであるが、ポータルで扱いやすい形式するためにもポータル用のデータベースに、夜間バッチ処理でコンバートして取り込み、時間割用テーブルを作成している。教務システムではユーザアカウントが教員コード・学籍番号のため、作成されたテーブルも教員コード・学籍番号をキーとしている。そのためポータルログインユーザ（名古屋大学 ID）で LDAP 情報から教員コード、学籍番号を得て時間割テーブルを検索しポートレット内に表示している。また、休講・補講、学生呼び出しについては、リアルタイムに表示したいため教務データベースを直接参照している。図 4 は教務連携処理のデータフローである。

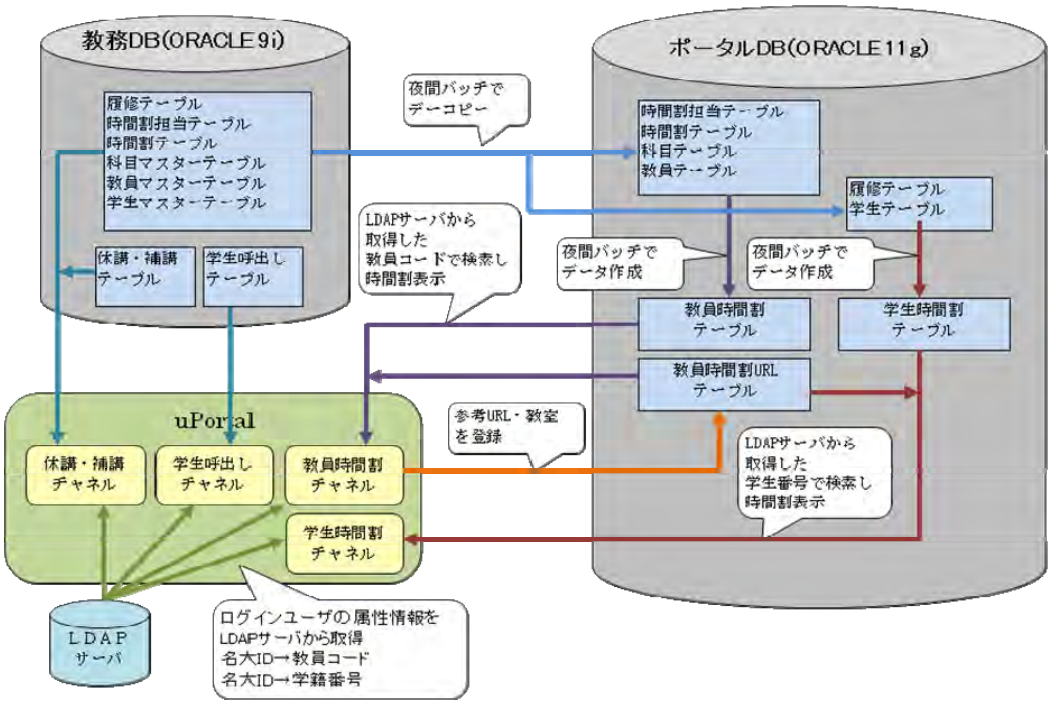

図 4. 教務連携部分データフロー

時間割ポートレットの機能についても説明する。

教員の時間割には担当する授業が表示されるが、各時間枠内に「リンク設定」ボタン、「NUCT 利用申請」ボタンが含まれる。「リンク設定」クリックで画面が切り替わり、“開講場所”、“授業の参考にするサイトへのリンク”を３つまで登録できる。この“開講場所”と“リンク先”は該当授業の履修学生時間割に表示されるので、学生はその授業の教室と参考情報を見ることできる。

また名古屋大学では“NUCT”という名称で授業コース管理システムを導入運用している。このシステムに対して教員が各授業のサイトを作成したい場合「NUCT 利用申請」クリックで、NUCT 管理者に申請のメールが送られる。そのメールを元に授業のサイト作成を担当者が行った後、教員に返信する。メール送信ではなく NUCT の API を利用して直接サイト作成することも可能だが、管理者の「人の目によるチェックを行いたい」という意見でメール送信することにした。今後、利用者が増えた場合は、サイト自動作成にする方が作業効率の軽減につながり他システムとの連携が進むと思われる。

3.5　アナウンス機能

ポータル内のコンテンツは教務連携のような動的な情報提供の他、固定の情報も多数含まれる。これら固定情報は毎回手で変更するしかないが、手間がかかるため更新を怠ってしまいがちである。その結果、情報が陳腐化して利用者も見る気がなくなるなど悪循環に陥る恐れがある。そこで固定情報を各部局の担当者がポータル上から容易に変更可能なツールを作成した。

また、全学向け・全学生向け・全教員向けなどを対象とした必須情報を、各部局の担当者が一括登録・公開できるアナウンス登録チャネル（図 5）を作成した。ここで登録された「自分宛の必須情報と参照リンク先」を、まとめて１つのチャネル内にリストアップすることで見逃しを防ぐアナウンス表示チャネル（図６）も作成した。これらのアナウンスは各ユーザがリンク先へジャンプした時や、確認したと判断される動作をトリガーとして削除される。

図 5. アナウンス登録

前述した NUCT 内でセキュリティ研修・自己点検などの WEB 研修を実施しているが、全学構成員が速やかに受講することを促進する目的もあって、この機能追加が提案された。この場合は NUCT 側で研修の答案送信クリック時に、そのユーザのアナウンスを削除したいという要望で、これを実現するには他システムに

もプログラム改修が必要になる。そこで担当者の負荷をなるべく軽減するために、組み込む JavaScript のサンプルを提供することにした。ただし却下される可能性の方が圧倒的に高いため、アナウンス登録時にその削除タイミングを選択可能にしておいた。

「アナウンス登録」ポートレットはあらかじめ担当者として申請された人だけが登録可能である。またその担当者が属する部局で作成されたものは一覧表示される。新アナウンスは１段目から入力し「登録」する。作成されたアナウンスは「公開」するまでは「修正」可能である。「公開」したら、それを「読んだ人数」を確認できる。最下段は“公開済み”アナウンスの例である。

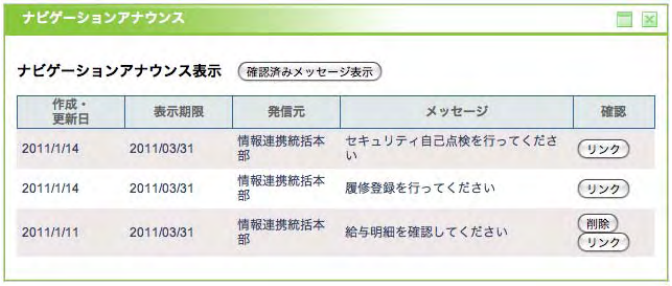

図 6. アナウンス表示

また削除のタイミングとして「ユーザ削除」「ジャンプ時削除」「他システム削除」のいずれかを選択させるが、「ユーザ削除」選択時は「アナウンス表示」ポートレットで文字通り自分宛のアナウンスをユーザが「削除」ボタンで消すことができる。「ジャンプ時削除」は「リンク」ボタンクリック時、「他システム削除」はポータルの別チャネルクリック時や他システム側で何らかの操作を行った際に削除される。

通常ポートレットは単体プログラムで独立したものが多いが、これは要望された機能を実現するため、複数のプログラムを組み合わせ小規模のシステムのような構成になった。図 7 にプログラム構成を示し、その特徴について述べる。

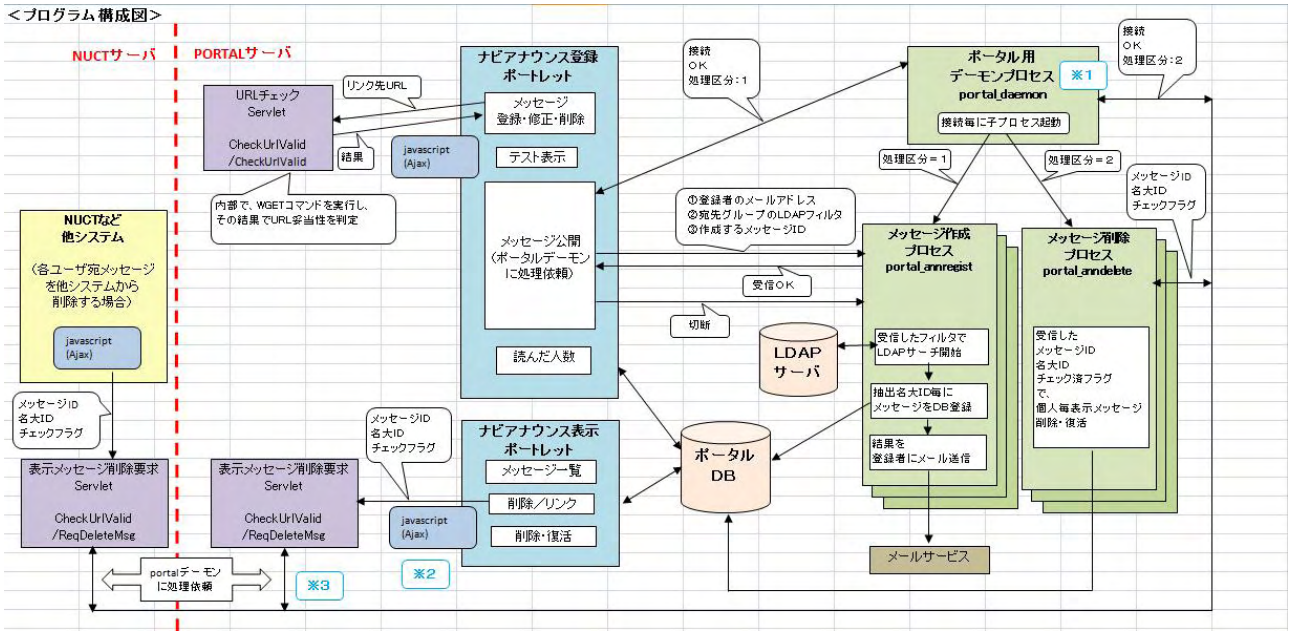

図 7. アナウンス機能プログラム構成

●特徴１

このポートレットは、C 言語でポータル用に開発したデーモンプログラムと socket で通信を行っている。このようにしたのは以下の理由からである。

複数ユーザ向けアナウンスを作成し、該当者だけに公開する場合、以下の２つの方法が考えられる。

【方法１】ユーザがログイン時にそのアナウンスが自分宛かどうか判断するため、アナウンスと共に登録された条件で LDAP 検索し、抽出されたデータ中に自分が含まれるどうかをチェック。

【方法２】アナウンス登録時に入力された条件であらかじめ LDAP 検索し、抽出されたユーザ全員分のデータを先に登録しておく。

ただし、LDAP サーバのサーチに時間がかかるため「ユーザのポータル操作時に毎回サーチ処理を行う」よりも、「一部の担当者がアナウンス登録時に LDAP サーチで時間がかかる」方がまだ良いので【方法２】を採用した。

よって「アナウンス登録」ポートレットで「公開」ボタンクリックにより、対象者全員分のデータを登録しておく事にした。

しかし作成したプログラムを動作確認したところ、全学向けアナウンス登録処理が LDAP サーチでまず抽出件数制限オーバーになり処理が失敗してしまった。この制限を増やすことは可能かもしれないが、どのみちブラウザが処理終了前にタイムアウトしてしまうであろうし、ユーザを何 10 分間も待たせる事はできない。このような理由で以下のような方法を考案した。

(1) ポートレットからは、デーモンに対して「メッセージ ID」「LDAP フィルタ」「担当者のメールアドレス」を送って登録処理を依頼し、すぐに切断する。

(2) 要求を受けたデーモンは 15 分ほどかけて LDAP サーチし、抽出された全員分のアナウンスを DB 登録する。

(3) 登録完了時にアナウンス登録担当者にメールで終了通知する。

この方法だと、ポートレットで「公開」クリックした直後、ブラウザが処理待ち状態にはならないが、実際に全員分のアナウンス登録が完了しているわけではないので、該当アナウンス欄に「処理中・・・」と表示される。担当者は完了通知メールを受信したら、ポータル画面をリロードし、公開完了状態になっているのを確認できる。

●特徴２

「ユーザ毎のアナウンス削除を他システム側で行わせる」という要望は、JavaScript と Ajax を使用して実現した。ただし、他システムが WEB アプリケーションでサーブレットコンテナを起動していることが条件になる。

(1) 他システム側にこちらから提供した JavaScipt を組み込み、同時に削除処理要求用のサーブレット（図 7 左下に示されている ReqDeleteMessage）を置いてもらう。（クロスドメインを避けるため）

(2) そのスクリプト内から Ajax で ReqDeleteMessage を実行しメッセージ ID, 名大 ID などのパラメータを渡す。

(3) サーブレット ReqDeleteMessage はパラメータを受け取り、ポータルデーモンに対して削除要求をする。（ReqDeleteMessage から直接ＤＢ削除してもよいが、他システムのサーバから直接ＤＢアクセスは避けたいため）

(4) ポータルデーモンは受信したメッセージ ID,名大 ID を元に DB 内の特定ユーザアナウンスを削除する。

4　今後の課題

以上のように名古屋大学ポータルカスタマイズを進めてきたが、今後もユーザ要望に基づく改良を続けていきたい。また 2011 年度 4 月からは国際化対応のため日本語・英語のユーザによる自動切り換えを実現する予定である。

その他、学内専用システムをポータル経由で学外からアクセスできないかという問合せが時々あるが、他システムへはリンクを張るだけなのでポータルを経由しても結局他システム側の IP 制限にひっかかり、学外アクセスは現状できない。その対応策として、ポータルとは別になるがリバースプロキシについて調査し導入を検討したい。

参考文献

[1] 長谷川裕一、麻野耕一、伊藤清人、岩永寿来、大野渉 “Spring2.0 入門”（株式会社技術評論社）

[2] 高橋登史朗 “入門 Ajax 増補改訂版”（ソフトバンク クリエイティブ株式会社）

[3] ティム・ハウズ、マーク・スミス “LDAP インターネットディレクトリアプリケーションプログラミング”（株式会社ピアソン・エデュケーション）