ホスト情報管理システムUCANN

ホスト情報管理システム UCANN

総合情報センター 情報システム部
小島 篤博
ark@center.osakafu-u.acjp

Abstract

ネットワークに接続された PC，WS などのホスト情報を一元的に管理するためには，ホスト名や IP アドレス，機種などの技術的な情報に加え，その機器の運用責任者やその所属部署などの管理組織的な情報も必要となる．大阪府立大学では，このようなホスト情報を蓄積したデータベースと，これを Web 上から参照・更新可能なインタフェースからなるホスト情報管理システム UCANN を独自に開発し，運用している．本稿ではその構築事例を紹介する．

1 まえがき

インターネットの普及に伴い，専門知識を持たないユーザによる TCP/IP ネットワーク利用の機会が増加している．

ネットワークへ新たな機器を接続したり，あるいは既存の機器を別のネットワークに移動したりする場合，ネットワーク管理者は IP アドレスの発行や DNS 登録を行い，利用者は機器ごとの FQDN（ホスト名）や IP アドレス，ゲートウェイ，DNS サーバなどを正しく設定する必要がある．この両者の作業において整合性が失われれば，接続した機器からネットワークが利用できないばかりでなく，場合によってはネットワーク全体を混乱させてしまう可能性がある．企業などでは，こういった混乱を避けるために IT 部門が一括して FQDN/IP アドレスの管理を行う傾向が強いのに対し，多くの大学では個々の構成部局ごとに独自にサブネットを構築・運営してきた経緯から，部局ごとに独立した管理組織が存在している場合が多い [1]．

大阪府立大学でも，FQDN/IP アドレスの管理権限は部局（学部や学科など）ごとに委譲する形で，キャンパスネットワーク全体のネットワーク管理を行っている．しかしながら，部局等のネットワーク管理者は一般に本来の職種と兼業している場合が多く，ネットワーク管理業務に要する作業負担が大きいという問題がある．その結果として DNS を含めたホスト情報の管理が徹底されないなどの事態の発生の一因ともなっている．また，提出された文書はファイルされてはいるものの，検索には手間がかかるなど実際の運用には十分に活用されているとはいいがたい．

そこで，大阪府立大学のキャンパスネットワークが整備されたことを期に，利用者による接続申請から IP アドレスの発行，DNS の登録までを一元的に行うことできるシステム UCANN (Utility for Campusnet Administration of Names and Numbers) を開発し，現在実務ベースでの運用を行っている．同システムは，運用ポリシーの異なる複数の構成組織からなる分散的なネットワーク管理体制に対応しており，大学のように多くの部局からなる階層的な組織に適用するメリットは大きい．

本稿では，UCANN において採用した 3 層システム構成，MVC アーキテクチャ，そして Java を効果的に活用したシステム開発の方法論について，開発事例を紹介する．

2 UCANN の概要

UCANN は，Web ベースのアプリケーションで広く採用されている 3 層システム（3-tier System）に基づいて構築されている（図 1）[2]．これは，バックエンドの RDBMS（データベース）に対するアクセスを Web サーバが中継し，クライアントの Web ブラウザから操作するというものであり，RDBMS の安全

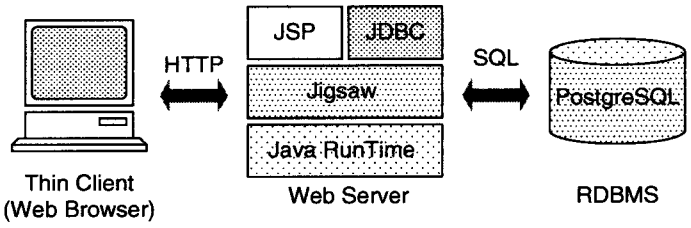

図 1: UCANN のシステム構成

性を確保するとともに，利用者に対して操作性の高いユーザインタフェースを提供することができるという利点がある．

RDBMS には PostgreSQL[3]，Web サーバには Jigsaw [4, 5] をそれぞれ採用している．Java で記述されている Jigsaw は拡張性が高く，様々な Java コンポーネントを組み込んで利用できるようになっている．例えば RDBMS と Web サーバ間の SQL(Structured Query Language) によるデータ交換を提供する JDBC(Java Database Connectivity)，Web ページの動的生成を提供する JSP(JavaServer Pages)[6, 7]，処理結果を電子メールで発信するための JavaMail は，いずれも Java で記述され容易に Jigsaw にプラグイン可能なモジュールである．

また，今回利用したソフトウェアは，PostgreSQL も含め，オープンソースあるいはフリーウェアであり，費用をかけずに構築することができるとともに，ソフトウェアライセンスに束縛されることなく他の用途への二次利用が可能である．

UCANN の開発に際して筆者らが行ったのは，主に次の 3 点である．

- RDBMS におけるスキーマ設計（ホスト，ネットワーク，管理者など）
- Web ベースのユーザインタフェース
- RDBMS とユーザの操作を仲介するロジック（JSP/Java）

以下では，UCANN の基本設計と実装について説明する．

3 基本設計

3.1 ワークフロー分析

小規模な組織では，IT 担当部門が組織全体のホスト情報管理を集中的に行う場合もあるが，総合大学のように規模が大きく，かつ部局ごとの独立性が高い組織では，サブネット/サブドメインごとに階層的なネットワーク管理組織を構成している場合が多い．大阪府立大学でも，このような複数の構成組織による分散的なホスト情報管理体制をとっており，利用者からの要求は管理組織に対する利用申請という形で行われる．ネットワーク利用申請としては以下の 3 つがある．

新規 ネットワークに新規に機器を接続する．機器の FQDN や接続サブネット，運用責任者等の情報が必要．

変更 既登録の機器の FQDN かつ/または接続サブネット等を変更する．

廃止 既登録の機器を廃止し，FQDN と IP アドレスを返還する．

ここで，新規申請，および変更申請において接続サブネットを変更する場合には，申請の時点では割り当てられる IP アドレスは未確定である．したがって，このような申請に対する処理手順は一般に次のようになると考えられる（図 2）．

1. 個々の機器の運用責任者による利用申請

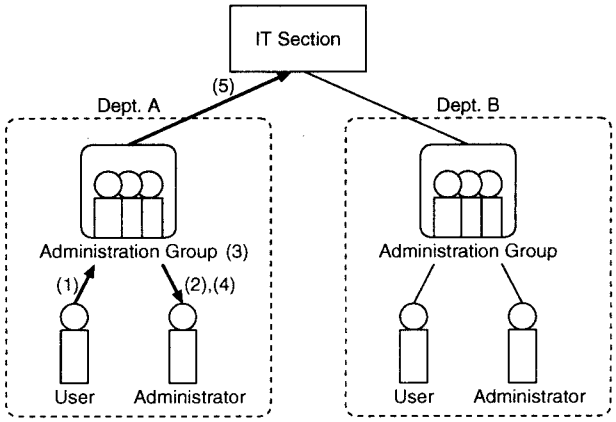

図 2: ネットワーク利用申請手順

2. ネットワーク管理者による FQDN の重複チェック，IP アドレスの割り当て

3. ネットワーク管理委員会等による利用承認

4. ネットワーク管理者による DNS 登録

5. 組織全体の IT 管理部門への利用報告

従来，このような処理は紙面や E-mail などの文書による事務手続きとして行われてきた．しかしながら，文書による手続きでは，申請中の項目について間違いや重複，不整合などの確認が人手に依らざるをえず，ネットワーク管理者の負担が大きいという問題がある．

そこで，ホストの機器情報（FQDN，IP アドレス，機種など）および管理情報（運用責任者，設置場所など）を RDBMS により電子化することで，各種申請手続きを効率化するとともに，ホスト情報の検索や閲覧など，登録情報の有効な活用を図る．

3.2 ワークフロー設計

まず，ホスト情報管理に携わる人的リソースを以下のように分類する．

利用者 個々のネットワーク機器の運用責任者であり，機器の接続や変更に際して利用申請を行い，FQDN や IP アドレスのオーソライズを受ける．

ネットワーク管理者 部局（学部や学科など）ごとのネットワーク管理担当者．担当部局内の利用申請の内容を確認する．

ネットワーク管理組織 部局内のネットワーク管理者からなる管理組織（委員会など）．ネットワーク機器の FQDN や IP アドレスの割り当てに関する承認権限を持つものとする．

DNS 作業担当者 部局内の DNS 情報を管理する作業担当者．利用承認の権限などはなく，委託業者など外部の人員を充てることも可能である．(組織によっては，ネットワーク管理者と兼任している場合もある．)

次に，機器ごとに運用責任者の所属，氏名，E-mail アドレス等が必要となるが，申請ごとに毎回同じ情報を入力するのは手間がかかる．そこで，初回に利用者の情報を登録しておき，その際に発行されるユーザ名とパスワードを利用するようにすれば，次回以降は既登録情報を再入力することなく申請を行うことができる．さらに，変更や廃止申請においては，既存の登録ホスト情報から該当ホストを検索することで，入力ミスなどを減らすことができる．

以上の事項を考慮してワークフローを見直し，図 3に示すような 3 つのセッションからなる登録手順とした．詳細は以下の通りである．

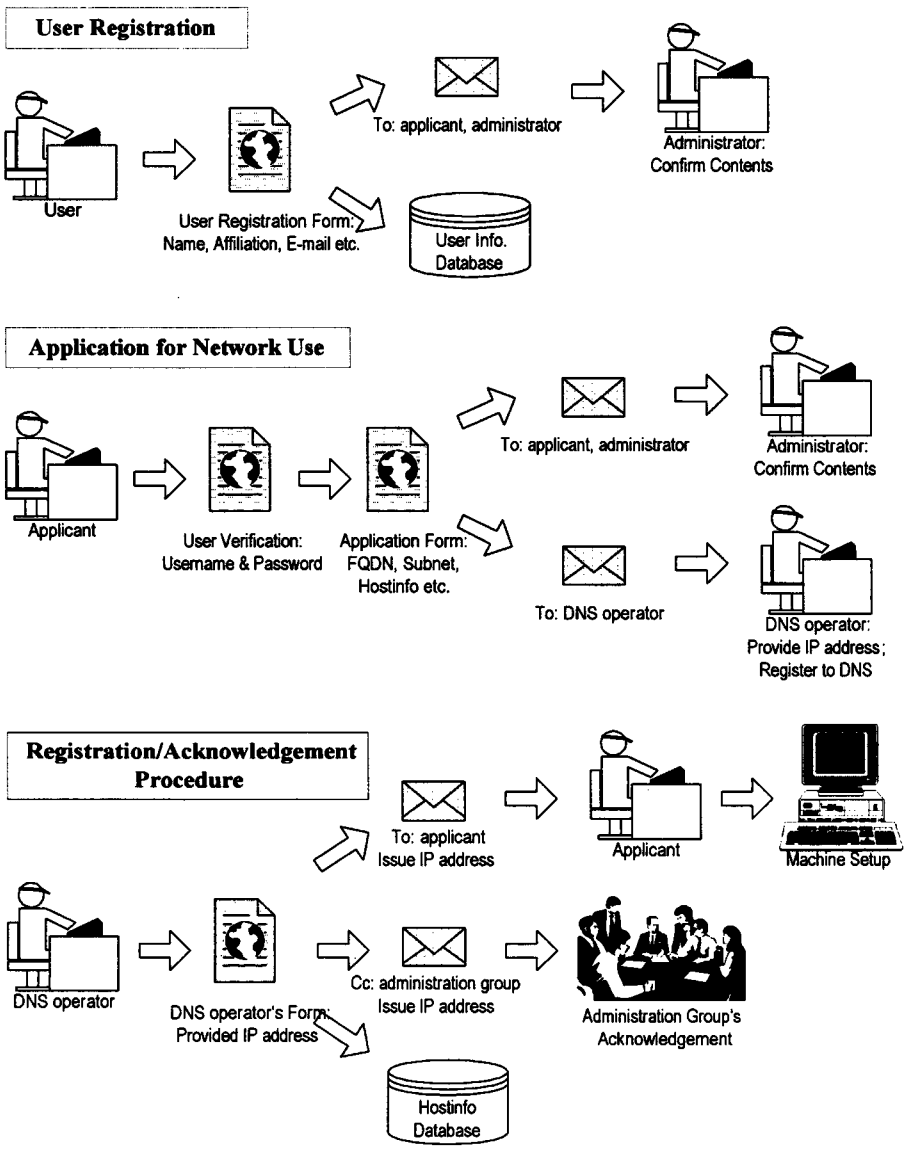

図 3: UCANN による利用申請手順

- ユーザ登録

  1. Web フォームへの必要事項（氏名，所属，E-mail 等）の記入.
  2. 所属部局等のネットワーク管理者による確認.
  3. 同時に UCANN データベースへのユーザ情報の登録.

- ネットワーク利用申請

  1. 申請者のユーザ認証.
  2. Web フォームへの必要事項（申請種別，FQDN，サブネット，機種等）の記入.
  3. 所属部局等のネットワーク管理者による確認.
  4. DNS 作業担当者への作業依頼.
  5. DNS 作業担当者による，IP アドレスの発行および DNS 登録.

- 登録/承認処理

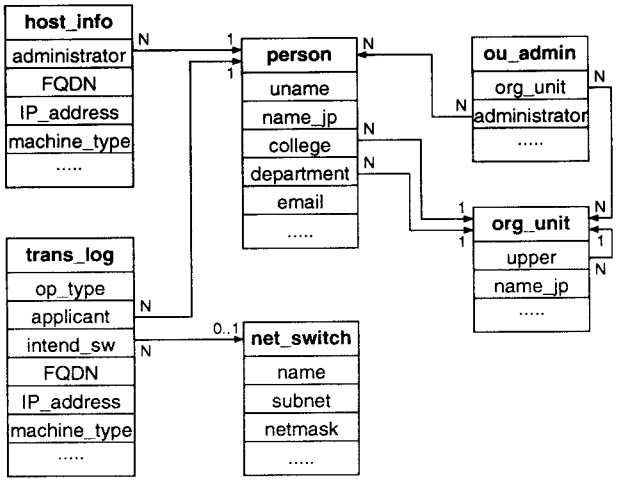

図 4: UCANN のスキーマ関連図（一部）

表 1: UCANN のスキーマ

スキーマ	説　　明
person	利用者の氏名，所属，連絡先など．
host_info	登録ホストの FQDN，IP アドレス，機種，運用責任者，設置場所など．運用責任者（administrator）フィールドは必須であり，運用責任者の person キーを保持．
trans_log	利用者からの申請レコード．ホスト情報に必要な情報のほか，登録種別（新規，変更，廃止など）を表す op_type フィールドや申請者の person キーを保持する applicant フィールドなどがある．
org_unit	学部，学科など，各階層における組織単位の情報を保持している．上位の組織へのリンク（upper）を保持．
ou_admin	組織単位と，そのネットワーク管理者との対応表．1 つの組織に複数の管理者を割当てることも可能．
net_switch	申請時点で IP アドレスが確定していない場合があるため，機器の接続先のサブネットを間接的に指定する．サブネットアドレスやネットマスクなどの情報を保持．

1. DNS 作業担当者による，必要事項（確定 IP アドレス等）の記入．
2. 申請者への IP アドレスの交付．
3. 所属部局等のネットワーク管理組織による事後承認．
4. 同時に UCANN データベースへのホスト情報の登録．

3.3 データベース・スキーマ

前述の登録手順に適合するように，データベース・スキーマの設計を行う．図 4は，UCANN のスキーマの相互関連図を示したものである．それぞれの矩形がスキーマであり，スキーマ名を太字で示している．他のスキーマへの参照は矢印で示し，“1 対 N” のような量的対応は矢印の両端に付記している．また，それぞれのスキーマの詳細は表 1に示す．

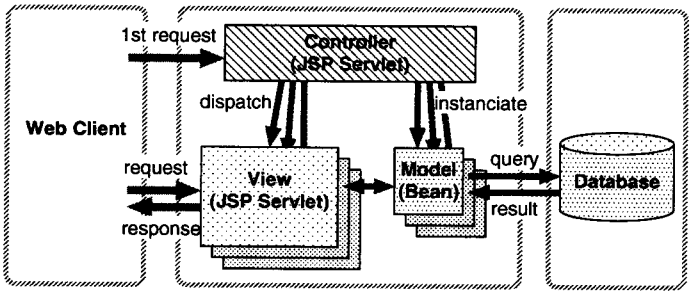

図 5: MVC アーキテクチャ

4 実装

次に，これらのスキーマを実際に入力した RDBMS を構築するとともに，これをバックエンドとするユーザインタフェースを設計・実装する．

4.1 MVC アーキテクチャ

UCANN の実装において考慮したのは，短期間で効率的にプロトタイピングを行いつつ，実務ベースに移行できる品質を確保できること，問題が生じたときにも容易に修正が可能であること，の 2 点である．これは，UCANN の開発を実際に行ったは業者やフルタイムの開発職ではなく，総合情報センターの教員が本来の業務と並行しつつ行わなければならなかったことからも特に重要であった．

このため，システムの各部をモジュール化し，それぞれの独立性を高めて品質を確保することとした．このようなアプローチの一つとして MVC(Model-View-Controller) アーキテクチャがあり，UCANN もこの設計パラダイムにしたがって実装を行った．MVC アーキテクチャは，図 5 に示すように，ユーザインタフェース（View），ビジネス・ロジック（Model），そして制御（Controller）を効果的に分離する設計パラダイムである [8]．ユーザインターフェースは，個々の Web ページのデザインなど表現方法に関わる部分である．ビジネス・ロジックは，アプリケーション本来の処理手順を，いくつかのデータ型（クラス）や手続きで定式化したものである．そして，一連の処理の流れの中でこれらの生成やデータの受け渡しを制御部が行っている．

JSP(JavaServer Pages) を利用することで，ビジネス・ロジックに豊富な Java コンポーネント・ライブラリ（JavaBeans）を適用することができ，またプレゼンテーション・ロジックには，Web ソースコード（HTML）中に Java コードを埋め込むことができるため，効率的なアプリケーション開発が可能となる [9]．

4.2 申請処理におけるデータフロー

図 6 は，新規申請を例として，一般利用者による申請手続きと，DNS 作業担当者による登録/承認処理のデータフロー図を示している．申請手続きは，ユーザ認証，ホスト情報の入力，既存の FQDN との重複チェックを順に行い，問題がなければ trans_log テーブルに新たな申請レコードを追加する．ここで受付け ID が決められ，DNS 作業担当者などにメールで送られる．

一方，DNS 作業担当者は，受付け ID をもとに申請レコードを呼び出し，空き IP アドレスを割当てて DNS に登録したのち，確定した IP アドレスを入力する．これでホスト情報は完全なものとなり，host_info テーブルに新たなレコードとして追加される．作業完了の通知は，割当てられた IP アドレスと共にメールで送られる．

申請手続きの Web ページの例を図 7に示す．

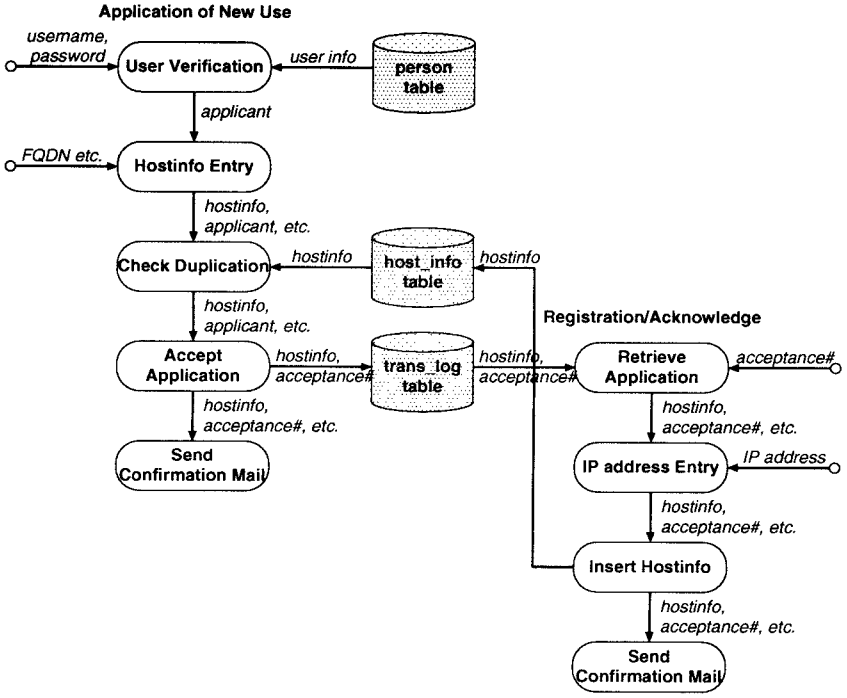

図 6: 新規申請におけるデータフロー図

5 検討

5.1 Java の採用について

UCANN の開発において Java を採用した理由は，すべて Java で記述されたオープンソースの Web サーバ（Jigsaw）が公開されており，様々な Java のモジュールを容易に結合することができること，バックエンド・データベースとの連携機能（JDBC）が標準化され，本システムで利用する PostgreSQL にも対応していること，サーバ・サイドでプログラムを実行し，Web ページを動的に生成するしくみ（JSP）が提供されていること，さらに，今後の機能拡張に対しても，同様に既存の Java コンポーネントを取り込むことで対応できると考えられることなどである．

実際に開発を終えてみると，データベースアクセスや E-mail など一般的な機能はほとんど既存のクラスライブラリを利用することができため，アプリケーション本来のロジックの開発に専念することができた．

一方，上記に挙げた項目は他のシステムの組み合わせ（たとえば Apache, PHP, Perl など）で実現することも確かに可能である．しかしながら，このようなアプローチでは，部分ごとに異なる言語を用いたり，システム全体でのリソースの共有が難しいなどの問題がある．Java を利用すれば，システムのすべてを共通のフレームワークで扱うことができ，より高度な連携も可能となる．

5.2 階層的な組織構成への対応

規模の大きな組織では，図 2に示したようにネットワーク管理組織が複数存在し，それぞれに運用方針が異なるというような複雑な管理体制となっている場合がある．UCANN では，データベース・スキーマの設計において組織単位（org_unit）の階層化をサポートしており，それぞれに応じた申請手順をビジネス・ロジックの中で実現している．

大阪府立大学では，大学全体のキャンパスネットワークの配下に学部単位，さらに学科単位という 3 階層の管理体制となっているが，それぞれの階層の役割の違いも含めて扱えるようになっている．

図 7: UCANN ページ画面

5.3 実業務への適用

UCANN は，大阪府立大学において 2000 年 4 月より試験運用を開始し，それまで文書により利用申請されたデータを移行した後，6 月より実際の業務処理に正式稼働している．2001 年 3 月現在で，ユーザ数約 750 名，登録ホスト数約 3900 件，また申請件数は累計約 900 件となっている．運用開始以来，ほぼ問題なく安定して動作している．

6 今後の課題

図 3 に示したように，現在本システムでは DNS 登録については人手で行っている．その理由として，従来から DNS 登録がなされずに機器を運用しているケースが散見され，アドレスの重複を避けるためには少なくとも UCANN による登録が徹底されるまでは自動化は難しいと判断したためである．

現在，DNS サービスは BIND を利用するのが一般的であるが [10, 11]，Java による DNS サービスの実装も開始されており，これが実用レベルの品質になれば，UCANN 自体が DNS のプライマリ・サービスを提供することも将来的には考えられる．

また，データベース化された情報の活用という点から考えると，現在はまだ利用者によるデータの検索・閲覧を提供している段階であるが，将来的には利用者の使用機器に応じた情報提供，例えば OS のバージョンアップやセキュリティ情報の E-mail による提供なども考えらえる．

いずれにしても，従来の DNS による FQDN と IP アドレスのみのホスト情報管理を，運用責任者（利用者）などを含めた統合的なデータベースに拡張していくことの意義は大きいと考える．

参考文献

[1] 秋葉泰俊, 泉裕, 上原哲太郎, 国枝義敏: "DNS サーバの管理負荷軽減システムの構築", 平 12 情処春期全大 1ZC-03 (2000).

[2] "効果的な 3 層エンジニアリング", SUN マイクロシステムズ ホワイトペーパー (1997). http://www.sun.co.jp/products/wp/

[3] 石井達夫: "PC UNIX ユーザのための PostgreSQL 完全攻略ガイド", 技術評論社 (1999).

[4] A.Baird-Smith: "Jigsaw: An Object Oriented Server", W3C Technical Report (1996).

[5] Y.Lafon, B.Mahe: "Jigsaw - The W3C's Server", W3C (2000). http://www.w3.org/Jigsaw/

[6] E.Pelegri-Llopart, L.Cable: "JavaServer Pages Specification", Java Software (1999).

[7] 原田洋子: "Java Servlet 最新サーバ・プログラミング", 秀和システム (1999).

[8] W.R.LaLonde, J.R.Pugh.: "Inside Smalltalk – v.2", Prentice-Hall, (1990).

[9] ゴヴァインド・セシャドリ: "具体例で学ぶ JSP Model 2 アプローチ", JavaWorld, Vol.4, No.7, pp.70-75 (2000).

[10] P.Albitz, C.Liu: "DNS & BIND 第 3 版", オライリー・ジャパン (1999).

[11] 西村浩二, 染川隆司, 相原玲二: "ユーザ認証機能を持つ DNS 動的更新システム", DICOMO'99 論文集, pp.49-54 (1999).