フレームワークによる3層Webアプリケーション構築法

島田　圭[†]　中所武司[†]

Webにより急速な発展を遂げたネットワークは，学校や企業だけでなく家庭にまで普及し始めている．特に，ブラウザから利用するWebアプリケーションの需要が高まっている．しかしそれをゼロから開発するのは，開発コストだけでなく開発方法を習得する時間もかかり，負担の大きな作業になる．そこで本稿では，フレームワークとDB構築ツールのプロパティ設定だけでWebアプリケーションを構築する方法について述べる．フレームワークは，人や物といったリソースのライフサイクル管理業務をドメインとし，その機能設計を再利用することでフォームデザインや検索結果表示，DBテーブル定義の設定によるアプリケーションの構築ができる．DB構築ツールは，ライフサイクル管理業務に必要なテーブルを全て用意し，テーブルの名前や列をカスタマイズすることで固有のDBを構築する．

Building of 3-tier Web Application by Framework

Kei Shimada[†] and Takeshi Chusho[†]

The demand of WWW application is rapidly increasing, because network comes into wide use in school, company, and home. This paper describes a method of building of the 3-tier WWW application by using a framework and a database building tool. A typical example of the 3-tier WWW application using life-cycle management of resouce like goods and person in small-office is taken. The framework has been extracted from the library system which was developed as an example of the life-cycle management. The tool backs up building database for the life-cycle management by providing a default on database design.

1. はじめに

1990年代後半に入ってから，コンピュータネットワークが学校や企業だけでなく家庭にまで普及し始めている．特に，Webブラウザから利用するWebアプリケーションが注目を浴びている．Webアプリケーションは，企業間取引を行なうBtoBシステムや，オンラインショッピングやチケット予約といったBtoCシステムで実用化され，その数も増える一方である．しかし，ゼロからのWebアプリケーション開発は，開発コストだけでなく，プログラムやデータベース(DB)を作成するノウハウの習得時間も必要なため，負荷が大きな作業になる．最近は学校や役所といった利益を生じない小規模なところでも，業務の効率化を図るために情報システムを導入したいというニーズがある．その上，学校や役所の事務手続きを家庭からでも可能にするために，情報システムをネットワーク上に展開しようと考えたとき，現状のWebアプリケーション開発では負荷が大きすぎて，なかなか実行に移せないと思われる．

そこで，Webアプリケーション開発の負荷を小さくするフレームワークとDB構築ツールを開発した．フレームワークは，ドメインで共通な部分を再利用するもので，各アプリケーションに依存する部分をカスタマイズするだけでアプリケーションの構築ができる．DB構築ツールは，DBテーブルの設計・作成を容易するもので，設計のデフォルトを提供したり，データ設計のプロパティを簡素化することで，設計による負荷を小さくする．

フレームワークには，すでに開発・利用されているものがあり，企業の基幹業務アプリケーションを構築するIBM社のSan Francisco[1)]が有名である．しかし，San Franciscoは基幹業務のような広い問題領域を対象にしているので，柔軟性が重視されており，問題領域の狭い業務には適用しづらい．問題領域が狭い業務のシステム構築には，フレームワークやコンポーネントを選択する必要がなく，必要最低限なものをプロパティ設定で構築できた方が良い[2)~4)]．

† 明治大学大学院理工学研究科基礎理工学専攻情報科学系
Computer Science Cource, Major in Sciences, Graduate School of Science and Technology, Meiji University.
E-MAIL:{shimada,chusho}@cs.meiji.ac.jp

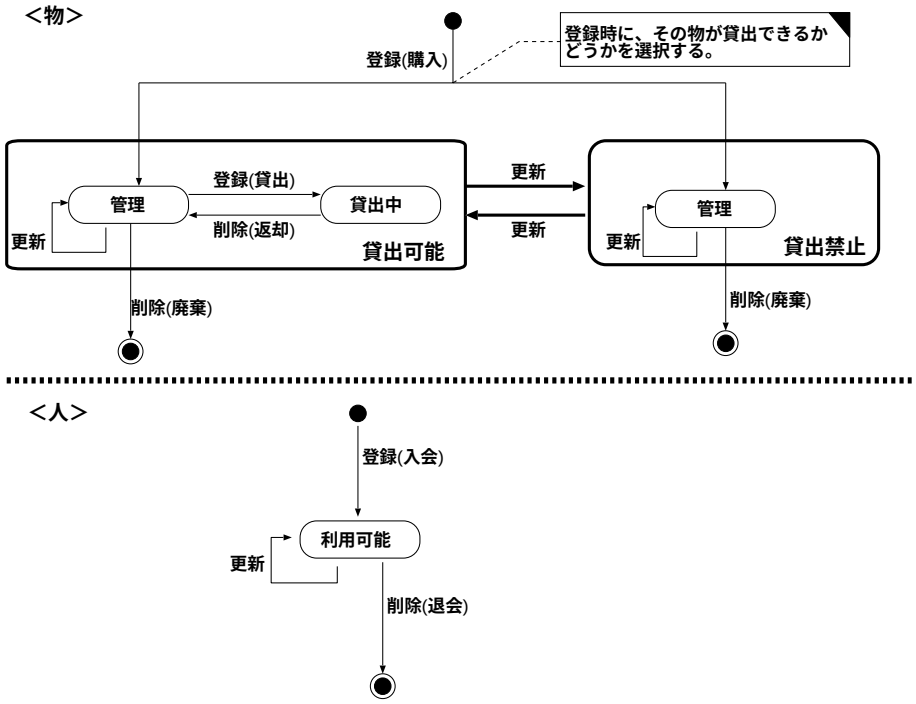

図 1 ライフサイクル管理業務のモデル
Fig. 1 Model of life-cycle management

本稿では，人や物といったリソースのライフサイクルを管理するライフサイクル管理業務 [5] をドメインとしたフレームワークと DB 構築ツールについて述べる．その例題システムとして研究室内の図書管理システムを構築し，ライフサイクル管理業務に共通な部分と図書管理業務に共通な部分を明確に分離することにより，フレームワークを開発した．また，プロパティ設定で DB の構築ができる DB 構築ツールも開発した．図書管理システムの DB をライフサイクル管理業務に抽象化した設計のデフォルトを提供することで設計の負荷を小さくした．

2. 3 層 Web アプリケーション

2.1 ドメイン

2.1.1 ライフサイクル管理業務

本研究では，ライフサイクル管理業務を 3 層 Web アプリケーションのドメインにする．ライフサイクル管理業務は，人や物といったリソースの登録（購入，入会，貸出）から削除（廃棄，退会，返却）までのライフサイクルを管理する業務である．

ライフサイクル管理業務においてリソースの状態が変化する様を，UML[6] のステートチャート図により示したのが，図 1 である．この業務には，登録や削除の他に更新や検索の機能を持つ．更新は，リソースのデータを変更したり，ある物において貸出可能から貸出禁止への変更や貸出禁止から貸出可能への変更を行なう．検索は，更新や削除でのデータ検索だけでなく，借りようとする物の有無，貸し出されているのか，あるいは貸出禁止になってるのかという貸出状態が調べられる．

2.1.2 ライフサイクル管理業務アプリケーション

ライフサイクル管理業務を Web アプリケーション化するとき，その利用者をリソース管理者とリソース利用者の 2 種類に分類した．リソース管理者は，物とリソース利用者のライフサイクルを管理する人間で，物やリソース利用者のデータの登録や更新，削除を行なう．リソース利用者は物を借りて利用する人間で，セキュリティ上，物のデータを格納する DB テーブルへの操作を禁止し，リソース利用者自らが物のデータを登録できないようにした．

しかし，この登録方法でリソース利用者が物を購入しそれをアプリケーションに登録するとき，リソース管理者に登録を依頼するので，その度に物を登録しなければならないリソース管理者にとって負荷の大きな作業になる．その負荷を小さくするために電子メイルでの登録依頼が考えられるが，DB テーブルの格納形式がリソース利用者に知られていない場合，各利用者独自の形式で物のデータを送られるので，それを解読するのに負荷がかかる．仮に，リソース利用者が DB テーブルの格納形式を知っているとしても，それをリ

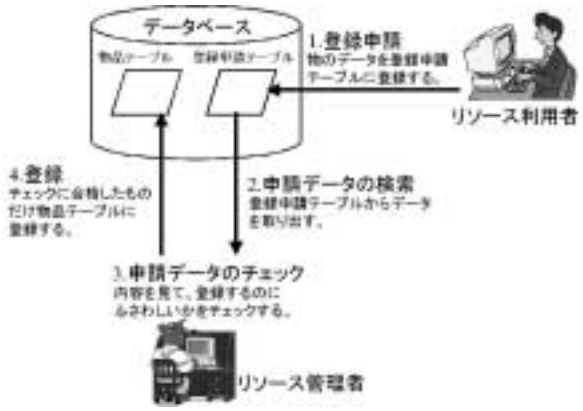

図 2 登録申請テーブルを用いたリソース利用者の物品登録
Fig. 2 Entry into the goods table by a resource user

ソース管理者自らがアプリケーションに入力するのは，リソース管理者にとって面倒な作業になる．

このような負荷を小さくするために，物の DB テーブルを複製した登録申請用の DB テーブルを用意し，リソース利用者がアプリケーションにより物のデータを登録申請用 DB テーブルに登録する機能を追加した．

リソース利用者による登録機能の手順を図 2 に示した．リソース利用者は，アプリケーションの登録申請フォームに物のデータを入力し，それを登録申請 DB テーブルに登録する．リソース管理者は登録したデータを見てチェックする．その結果，物品 DB テーブルに登録するにふさわしいと判断したときに登録する．この機能は，登録申請フォームの入力により，リソース利用者からの登録データが統一された形式でリソース管理者に送られ，リソース管理者は登録データをチェックするだけで，フォームの入力作業が無くなる．

2.2 例題アプリケーション

2.2.1 図書管理システム

ライフサイクル管理業務の例題アプリケーションとして，図書管理システムを Java で開発した．このシステムは，図書やそれを借りる人間のライフサイクルを管理するだけでなく，図書の検索や貸出・返却，図書の貸借状況を一覧表示する貸出一覧表示や，返却延滞状況を一覧表示する延滞一覧表示がある．図書検索機能により，調査したい技術に関連した本の有無がすぐ分かり，また新たに図書を購入するとき，同じものが購入されていないかを事前に調べられる．貸出一覧表示や延滞一覧表示により，貸出中の本を見たいときに誰が借りているかがすぐ分かる．

ライフサイクル管理業務において，リソース管理者とリソース利用者の立場や役割が異なるので，図書管理システムも管理者用と利用者用の 2 つのシステムを開発した．管理者用システムには，図書登録や図書削除といった図書データを管理する機能を持ち，利用者用システムは図書貸出や図書返却の機能，起動時のログインで使うパスワードの更新機能を持つ．なお，図書検索や図書貸出一覧，図書延滞一覧は，2 つのシステムで共通の機能とした．

図 3 図書管理システムの図書登録フォーム
Fig. 3 Book entry form of the library system

現在，JDK1.1 で開発したものが研究室内で利用されている．その画面例として，図書登録フォームを図 3 に示す．

2.2.2 例題アプリケーションの構成

図書管理システムは図 4 に示すクライアントとアプリケーションサーバ（AP サーバ），DB サーバの 3 層で構成している．クライアント・サーバ間の通信は JavaRMI を，サーバ内における AP サーバ・DB サーバ間の通信は JDBC ドライバ（Oracle Thin Driver）をそれぞれ用いた．

クライアントは Web ブラウザだけで，固有のクライアントソフトウェアをインストールする必要がない．サーバ側は，AP サーバと DB サーバの 2 層になっている．AP サーバには，Java アプレットをクライアントにダウンロードする Web サーバと，DB の接続ややり取りを行なうサーバプログラムがある．そして DB サーバには，RDB 管理システムの Oracle8 を用いた．

図書管理システムは，Web ブラウザに図書管理シス

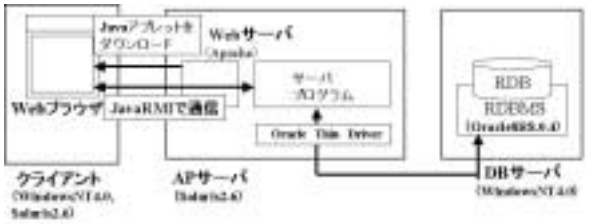

図 4 3 層 Web アプリケーションの構成
Fig. 4 Structure of 3-tier WWW application

テムの URL を入力し，Web サーバから Java アプレットをダウンロードすることで起動する．初期画面で，管理者用／利用者用を選びログインしてからはじめてシステムが利用できる．次に，図書登録フォームのような機能フォームで必要事項を入力し，検索や登録のボタンをクリックすると，入力したデータから SQL 文を自動生成し，それを AP サーバのサーバプログラムに送る．サーバプログラムでは，その SQL 文で DB を処理し，その結果を機能フォームに送り返す．

3. フレームワークの開発

3.1 フレームワーク

フレームワークとは，ある問題領域におけるソフトウェアの再利用を目的としたモジュール群で，対象となる問題領域に共通なフローズンスポットと個々のアプリケーションに依存するホットスポットの 2 つの要素からなる [7]．フレームワークを用いるとフローズンスポットが再利用できるため，ホットスポットのカスタマイズでアプリケーションを構築できる．これまで開発されたフレームワークには，Apple 社の MacApp[8] のような GUI アプリケーションフレームワークもあるが，ライフサイクル管理業務アプリケーションを構築するには，IBM 社の San Francisco のような業務アプリケーションフレームワークが適する．

San Francisco は基幹業務をドメインとし，フレームワークとコンポーネントの組合せでアプリケーションを構築するという，柔軟性を重視したものである．しかし，ライフサイクル管理業務は，人や物の登録，更新，貸出などを行なうだけの適用範囲が狭い業務なので，フレームワークとコンポーネントを選択し，それらを連携させるという煩わしい作業は必要ない．むしろ適用範囲が狭い業務には，フレームワークのみを提供し，その拡張及び部分修正によるカスタマイズでアプリケーションを構築した方が良い．

3.2 フレームワークの開発方法

本研究ではフレームワークを開発するのに，まず図書管理システムを開発した．その完成後に，もう 1 つ

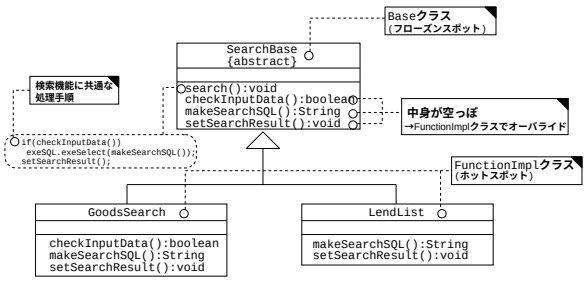

図 5 初期のフレームワーク
Fig. 5 The initial framework

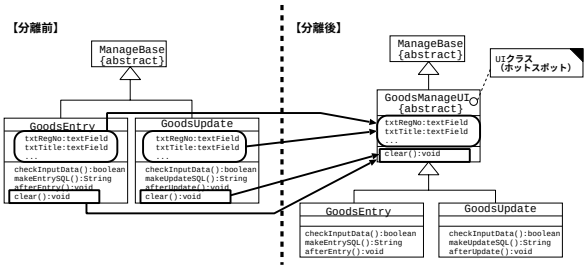

図 6 UI クラスの分離
Fig. 6 Separation of the UI class from the FunctionImpl class

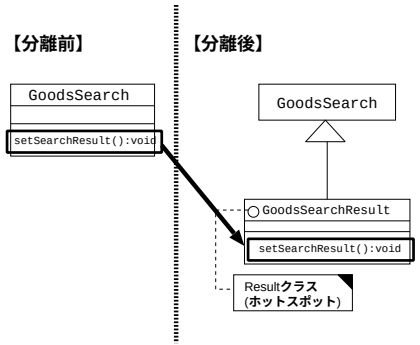

図 7 Result クラスの分離
Fig. 7 Separation of the Result class from the FunctionImpl class

の例題として考えた備品管理業務と比較することにより，異なる部分をカスタマイズ箇所，すなわちホットスポットとして特定した．このホットスポットを独立なクラスに分離することで，ホットスポットとフローズンスポットが明確に分離した形になり，フレームワークとして他の業務アプリケーションの構築に適用できるようになった．

3.3 フレームワークの開発

3.3.1 初期のフレームワーク

図書管理システムで行なわれる DB 処理は，クライアントが発行した SQL 文を AP サーバ経由で DB サーバに投げ，その結果をまた AP サーバ経由でクライアントに返すといった手順で行なう．AP サーバにあるサーバプログラムは，DB の接続と処理を行なう

プログラムの状態	フローズンスポット		ホットスポット		再利用率
	クラス数	ステップ数	クラス数	ステップ数	
初期のフレームワーク	7	749	18	2,683	22%
UI クラスの分離	26	1,657	10	1,163	59%
Result,SQL クラスの分離	35	2,115	15	818	73%

再利用率＝フローズンスポットのステップ数÷アプリケーション全体のステップ数

表 1 開発作業を通したフレームワークの再利用率の推移

Table 1 Improvement of the framework's reusablity

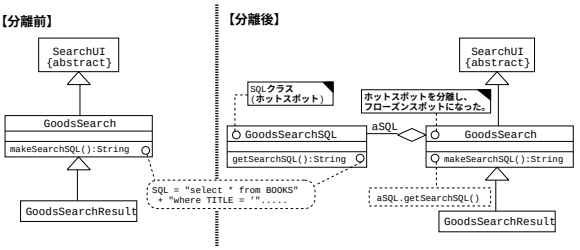

図 8 SQL クラスの分離

Fig. 8 Separation of the SQL class from the FunctionImpl class

部分で，JDK の `java.sql` パッケージが提供する API を用いて実装した．

この DB 処理を，継承機能による差分プログラミングで実装した．図書登録や，図書貸出，図書削除などの各機能が行なう処理において，フォーム入力項目のチェックから SQL 文の発行，DB からの検索結果をフォームに表示させるといった手順は，全機能共通である．そこで，処理手順を抽象クラスで実装し，それを継承する具象クラスに各機能固有の処理内容を実装した．このとき，全機能共通の処理手順を実装した抽象クラスを Base クラス（フローズンスポット），それを継承する各機能固有の処理内容を実装したクラスを FunctionImpl クラス（ホットスポット）と名付けた．例として図書検索機能に関する初期のフレームワークの構成を，図 5 に示す．

3.3.2 ホットスポットの特定

図書管理システムを開発してから，そのプログラムを備品管理システムの開発に適用することで，カスタマイズすべきホットスポットを特定した．図書管理業務と備品管理業務では，図書と備品といった物の違いだけなので，ホットスポットも物の定義に伴う箇所のみとなる．物の定義に影響する箇所は次の 3 箇所で，これらが分離すべきホットスポットである．

- フォームのデザイン設定
- 検索結果の表示設定
- DB テーブルの定義

これらは全て DB の設計に伴うものである．タイトルや出版社名などの図書データの属性が備品名や製造元といった備品データの属性に変わることで，登録フォームのような物のデータを入力するフォームにある入力欄の名前とその入力欄に検索結果を表示する部分を設定し直す必要がある．また，SQL 文にある DB テーブルの定義部分もカスタマイズしなければならない．

3.3.3 UI クラスの分離

フォームのデザインを FunctionImpl クラスで実装した場合，同じ記述が複数の FunctionImpl クラスに散在し，余分なホットスポットを作ってしまう．そこで，FunctionImpl クラスからフォームのデザイン設定部分を UI クラス（ホットスポット）なる独立したクラスに分離した．

UI クラスの分離例を図 6 に示す．図書登録と図書検索のフォームデザインが同じであるため，デザインに関する UI 変数やメソッドを GoodsManageUI という UI クラスに分離した．

3.3.4 Result クラスの分離

FunctionImpl クラスから DB 検索結果表示メソッドを，Result クラスという独立したクラスに分離した．図書管理システムが行なう検索結果の表示は，UI クラスで生成した入力 UI コンポーネントに結果を表示するので，UI クラスを継承するように Result クラスを分離した．

Result クラスの分離例を図 7 に示す．GoodsSearch にある setSearchResult メソッドを GoodsSearchResult という Result クラスに分離した．

3.3.5 SQL クラスの分離

FunctionImpl クラスにある SQL 文の発行部分を，SQL クラスなる独立したクラスに分離した．FunctionImpl クラスから SQL 文を 1 つずつ SQL 文参照メソッドに格納し，それらを SQL クラスにまとめた．SQL クラスと FunctionImpl クラスの間を has-a の関係にし，FunctionImpl クラスが SQL 文を発行する度に SQL 参照メソッドを呼ぶようにした．

SQL クラスの分離例を図 8 に示す．GoodsSearch から SQL 文を発行している部分を抽出し，それを GoodsSearchSQL なる SQL クラスに分離した．

3.3.6 再利用率の推移

フレームワークを開発する過程での再利用率の推移を表 1 に示す．再利用率とは，フレームワーク全体で占めるフローズンスポットの割合をステップ数で計算したものである．初期のフレームワークでは再利用率が 21%であったのに対し，FunctionImpl クラスからホットスポットを分離した時点では 78%になり，再利用効果が向上したと言える．

4. アプリケーションの構築方法

4.1 フォームのデザイン設定

4.1.1 UI クラスのカスタマイズ

UI クラスのカスタマイズは，図書の著者名のような物の属性名を設定するためで，UI クラス内の UI コンポーネントの生成・配置を行なうコードを書換える．UI のレイアウトは，`java.awt` パッケージが提供する GridBagLayout を簡略化した setComp メソッド [9] を用い，座標設定によってコンポーネントが配置できる．図書登録フォームにおける図書の登録番号を入力するテキストフィールドの生成・配置を以下のように記述する．

```
/* txtRegNo:登録番号を入力する TextField */
txtRegNo = new TextField(20); //生成
setComp(0, 1, 1, 1, txtRegNo);//配置
```

4.1.2 複合 UI コンポーネント

UI クラスのカスタマイズを容易にするために，ラベルやテキストフィールド，テキストエリアを組合わせたフレームワーク向けの複合 UI コンポーネントを開発した．複合 UI コンポーネントを UI クラスに適用することで，以下の点で UI のカスタマイズが容易になる．

(1) コンポーネントの数が減る．
(2) コンポーネントのサイズを設定せずに済む．

複合 UI コンポーネントは 1 つの項目に 1 つの UI コンポーネントを割り当てるので，生成するコンポーネントの数を減らせる．日付の入力で間違いを防ぐために設けた記入例を，日付の入力するコンポーネントと一緒にした日付入力コンポーネントを作成し，それを UI クラスに適用することで，作成するコンポーネントの数が少なく済む．

また，入力 UI コンポーネントの生成に必要なサイズをコンポーネントに吸収することで，サイズを設定せずに済む．先に例で示した登録番号の入力フィールドに複合コンポーネントを適用すると，次のようになる．

```
/* txtRegNo:登録番号を入力する UI */
txtRegNo = new FieldUI();
setComp(0, 1, txtRegNo);
```

4.2 検索結果表示の設定

検索結果の表示形式は，Result クラスの setSearchResult メソッドで設定する．検索結果は列単位で行なわれており，列の名前を記述するだけで検索結果表示が設定できる．例として，貸出一覧機能の Result クラスで記述した図書の名前をフォームのテキストエリア `listUI` に表示する記述を以下に示した．このとき，検索結果である `exeSQL.getResult("TITLE")` のうち列の名前である `TITLE` の部分を書換えれば良い．

```
/* listUI に検索結果を表示 */
listUI.setText(exeSQL.getResult("TITLE"));
```

4.3 DB テーブルの定義

DB テーブルは，その名前とそれが持つ列の名前を SQL クラスで設定することで定義する．SQL 文自体は参照メソッドで用意しているので，処理する対象の DB テーブルの名前とその列の名前を記述するだけで SQL 文が完成する．参照メソッドの例として，図書検索機能の SQL クラスにある図書を検索する SQL 文の参照メソッドを以下に示す．下線部はカスタマイズ箇所で，`BOOKS` がテーブルの名前，`TITLE` と `KEYWORD` はテーブル `BOOKS` が持つ列の名前である．

```
/* フォームに入力したキーワードを基に，図書の検索を行なう SQL 文 */
public String getSearchSQL(){
 String SQL = "select * from BOOKS"
  + "where TITLE = '"
   + SearchUI.key.getInputData() + "'"
  + "or KEYWORD = '"
   + SearchUI.key.getInputData() + "'";
 return SQL;
}
```

5. データベースの構築

5.1 DB 構築ツール

DB サーバの構築による負荷を小さくするために，DB 構築ツールを開発した．DB 構築ツールは，1 つのデザインフォームで 1 つのテーブルを作成する仕組みになっている．デザインフォームにテーブルの名前やその列を入力し，フォーム下部にある「テーブル作成」ボタンをクリックすると，フォームに入力したデータから SQL 文を自動生成し，それを DB サーバに投げて実行させることでテーブルを構築する．デザ

図 9 管理者テーブルのデザインフォーム
Fig. 9 Design form for an administrator table

インフォームの例として，管理者テーブルのデザインフォームを図 9 に示す．

5.2 DB 構築ツールの特徴

DB テーブルを作成するには，テーブルの名前や列だけでなく，列を設計するときにキーやデータ型も決めなければならない．DB テーブルの構築を SQL 文でする場合は SQL 文の知識を習得するのに負荷がかかり，Oracle8 のような DB 管理システムで作成するときでも，数値データにおける小数点以下の桁数「スケール」や，その列に格納するデータに対する NULL の許可や一意であるかといった，設定すべき値が多すぎる [10]．

これら SQL 文や DB 管理システムによる DB テーブル構築で生ずる負荷を小さくするのに，DB 構築ツールを開発するときに次の 2 点を考えた．

(1) DB 設計の再利用
(2) 設計の抽象化

(1) は，図書管理システムの DB を抽象化したテーブル群を用意し，各テーブルのデザインフォームを用いて DB の構築ができる．また，抽象化したテーブルの列定義を各デザインフォームのデザイン入力欄に初期設定することで，テーブルの作成手順で迷わずに済み，DB 構築での負荷が小さくなる．

(2) は，Oracle8 で図書管理システムの DB を構築するときに必要なかった値設計やチェック設計を，DB 構築ツールでは省略した．また，列はデータ型とサイズのそれぞれを設定していたのを，データ型とサイズを組合わせたデータ型を作成・提供したことで設定の手間がなくなり，設計時の負荷が小さくなる．作成したデータ型は，文字列可変長型でサイズ 80 の「フィールド型」と文字列可変長型でサイズ 200 の「エリア型」，日付型で固定サイズの「日付型」の 3 つである．

5.3 DB 設計の再利用

図書管理システムの DB を抽象化したものをデフォルトテーブルと名付け，それをデザインフォームにあるデザイン入力欄の初期値に設定した．デフォルトテーブルは次の 5 つテーブルである．

- 物品テーブル
- 管理者テーブル
- 利用者テーブル
- 貸出テーブル
- 登録申請テーブル

物品，管理者，利用者の 3 つのテーブルはリソースのデータを，貸出テーブルはリソース利用者の貸出記録を，登録申請テーブルはリソース利用者からの物品登録記録をそれぞれ格納している．DB 構築ツールはこれら 5 つテーブルのデザインフォームを用いて，DB を構築する．

5.4 テーブルの作成手順

5 つのデフォルトテーブルには，外部キーを用いた参照関係がある．例として，物品テーブルと管理者テーブルとの参照関係を UML のクラス図で表現したものを図 10 に示す [11]．これはリソース管理者が物を登録するとき，その登録した管理者を特定するための登録者 ID を物品テーブルの外部キーに設定した．そうすることで，物品テーブルは登録者 ID を通して管理者テーブルを参照する．物品テーブルを作成する SQL 文は以下のように記述する．下線部で，物品テーブルの登録者 ID が管理者テーブルの管理者 ID を参照することを定義している．

```
create table 物品 (
登録番号 verchar2(80) primary key,
登録者 ID verchar2(80)
      references 管理者 (管理者 ID),
名前 verchar2(80),
製造元 verchar2(80),
購入日 date,
貸出許可 verchar2(80)
)
```

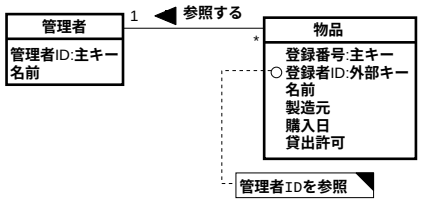

図 10 物品テーブルと管理者テーブルとの参照関係
Fig. 10 Referential relation between the goods table and the admin table

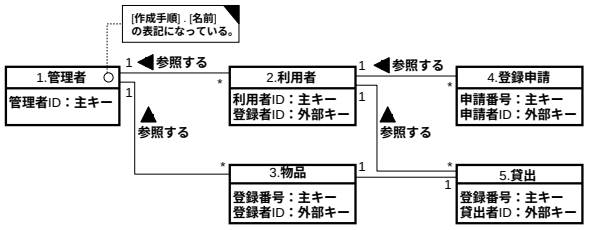

図 11 デフォルトテーブルの参照関係
Fig. 11 Referential relation in the default tables

上の SQL 文を実行するには，事前に管理者テーブルを用意する必要があるため，管理者テーブルの次に物品テーブルを作成するという手順になる．そこで，DB 構築ツールでテーブルの作成手順を決めることで，参照関係に関する知識がなくても問題なく DB の構築ができるようにした．作成手順は，全てのデフォルトテーブルから参照関係を見つけ，それぞれで参照されるテーブルから作成するように決める．

作成手順を決めるためにアプリケーションが持つ全機能におけるデフォルトテーブルの参照関係を調べあげた結果，デフォルトテーブルには図 11 のような参照関係があった．これより，デフォルトテーブルの構築手順を次のように決めた．

(1) 管理者テーブル
(2) 利用者テーブル
(3) 物品テーブル
(4) 登録申請テーブル
(5) 貸出テーブル

DB 構築ツールでは，この構築手順に従ってデザインフォームを遷移・起動させることで，順序に迷うことなくかつ一連の操作で DB の構築ができるようになった．

6. おわりに

今回は，ライフサイクル管理業務という適用範囲の狭い業務を行なう 3 層 Web アプリケーションをフレームワークと DB 構築ツールで構築する方法について述べた．例題である図書管理システムから抽出したフレームワークを用いることで，フォームのデザインや DB からの検索結果表示，DB テーブルの定義といったホットスポットのプロパティ設定だけで，Web アプリケーションの構築ができた．また，図書管理システムの DB から抽象化したデフォルトテーブルを用いた DB 構築ツールにより，DB テーブルの名前定義と列のカスタマイズのみで DB の構築ができた．

今後はフレームワークと DB 構築ツールの機能を改良しながら，それらの連携によるアプリケーション構築方法について考察していく．

参 考 文 献

1) IBM サンフランシスコ
http://www.developer.ibm.com:8080/welcome/java/sfindex.html
2) 藤原克哉，中所武司：窓口業務を例題としたエンドユーザ向き分散アプリケーションフレームワーク wwHww の開発と適用評価，情報処理学会論文誌，Vol.41，No.4，pp.1202-1211 (2000).
3) 藤原克哉，中所武司：エンドユーザ向きアプリケーションフレームワーク wwHww-知的ナビゲーション機能の XML による実現方式-，情報処理学会ソフトウェア工学研究会資料 SE-2000-128, pp.1-8(2000).
4) 南谷圭持，藤原克哉，中所武司：エンドユーザ向きアプリケーションフレームワーク wwHww - 自動記入エージェント実現方式-，情報処理学会第 61 回全国大会講演論文誌 (1)，pp.273-274(2000).
5) 島田圭，中所武司：ライフサイクル管理業務に注目したアプリケーションフレームワークの抽出と再利用効果の考察，情報処理学会第 59 回全国大会講演論文誌 (1)，pp.233-234(1999).
6) Craig, L.:Applying UML and Patterns:an introduction to object-oriented analysis and design, Prentice HallPTR(1998). (依田光江訳，今野睦，依田智夫監訳：実践 UML パターンによるオブジェクト指向開発ガイド，ピアゾン・エデュケーション社 (1998)).
7) Pree, W.:Design Patterns for Object-Oriented Software Development, ACM Press(1995). (佐藤啓太，金澤典子訳：デザインパターンプログラミング，トッパン (1996)).
8) Developer-MacApp
http://developer.apple.com/tools/macapp/
9) 菊田英明：実践 JDBC Java データベースプログラミング術，オーム社 (1998).
10) 梅田弘之，碓井満：Personal Oracle8 で学ぶ Oracle8 入門，エーアイ出版 (1998).
11) Paul, D. and Joseph R.H.:ORACLE8:Design Using UML Object Modeling, Berkeley(1999). (SE 編集部訳，日本オラクル社監修：ORACLE8 UML オブジェクトモデリング，翔泳社 (1999)).