DBPowder-web: RDBMSを用いたウェブアプリケーションの構築を支援するフレームワーク

村上　直†

† 高エネルギー加速器研究機構 計算科学センター　〒305–0801 茨城県つくば市大穂 1–1
E-mail: †tadashi.murakami@kek.jp

あらまし　本研究で提案する DBPowder-web は，RDBMS を用いたウェブアプリケーションの構築を支援するフレームワークである．ウェブアプリケーションの構築では，ウェブ技術の他に，RDBMS 技術やオブジェクト指向言語など，様々な要素技術が用いられる．これにより，開発過程の複雑化やシステムの保守性の悪さが指摘されている．特に，RDBMS における CRUD 機能 (Create,Read,Update,Delete) の開発で，その傾向は顕著である．ウェブアプリケーションの構築を簡素化するフレームワークは存在するが，フレームワークそのものが複雑であったり，機能拡張が制限されたりするケースが多い．DBPowder-web は，データスキーマを基にして CRUD 機能を持つウェブアプリケーションを自動生成する．自動生成コードはスキーマの変更に追従して何度でも再作成できる．また，開発者は DBPowder-web が指定した拡張ポイントに生成コードの機能拡張を行うことができる．これらは，ウェブアプリケーション構築の簡素化と機能拡張性を両立する．本研究では，DBPowder-web のプロトタイプシステムを開発し，その性能を評価する．
キーワード　Web とインターネット, コンテンツ処理, DB 言語, 開発手法, フレームワーク, モデル駆動型開発

DBPowder-web: Web Application Development Framework with RDBMS

Tadashi MURAKAMI†

† Computing Research Center, High Energy Accelerator Research Organization (KEK)　1–1 Oho, Tsukuba, Ibaraki, 305–0801 Japan
E-mail: †tadashi.murakami@kek.jp

Abstract　We propose *DBPowder-web*, a web application development framework with RDBMS. In the development of web applications, we need not only web techniques but also RDBMS techniques, object-oriented language techniques, and so on. These requirements bring complexity and low maintainability to the development, especially CRUD(Create, Read, Update, Delete) functions on RDBMS. There are some web application frameworks which enlighten to these problems. However, these frameworks also have problems: one of them restricts scalability, the other itself complex, and so on. *DBPowder-web* creates web application skeletons which have CRUD functions based on a data-schema. *DBPowder-web* is able to recreate skeletons when structures of the data-schema are changed. Furthermore, developers can add functions to the skeletons using the APIs of *DBPowder-web*. These simplify and bring scalability developing web applications. We developed a *DBPowder-web* prototype system and evaluated it.
Key words　Web and Internet, dealing with contents, DB modeling language, development, framework, model-driven development

1. はじめに

ウェブアプリケーションは，ユーザにとって利便性が高く，インターネット，イントラネットともに普及している．しかし，ウェブアプリケーションの適切な構築は容易ではない．

ウェブアプリケーションの構築では，ウェブ技術の他に，RDBMS 技術やオブジェクト指向言語など，様々な要素技術が用いられる．これにより，開発過程の複雑化やシステムの保守性の悪さが指摘されている．特に，RDBMS における CRUD 機能 (Create,Read,Update,Delete) の開発で，その傾向は顕著である．ウェブアプリケーションの構築を簡素化するフレームワークは存在するが，フレームワークそのものが複雑であるか，あるいは機能の拡張性が制限されるケースが多い．

本研究では，上記の問題を解決するために，RDBMS を用い

Layer 1	Customizing template web applications
Layer 2	Using Wizards to create related sets of components
Layer 3	Designing via WYSIWYG, direct manipulation, parameter forms
Layer 4	Editing layout code(similar to HTML, ASP.NET, JSF)
Layer 5	Editing high-level behavior code
Layer 6	Modifying and extending the underlying component framework
Layer 7	Editing PHP code

表 1 “Click” における，“gentle slope of complexity”

たウェブアプリケーションの構築を支援するフレームワーク DBPowder-web を提案する．

DBPowder-web に基づく開発では，はじめに DBPowder Model Description Language(DBPowder-MDL) を用いてデータスキーマを記述する．DBPowder-MDL は，簡潔な書式でデータの構造を表現できる記述言語である．DBPowder-MDL を用いてデータスキーマを記述しコンパイルすると，RDBMS のリレーショナルスキーマ及びスキーマ内のデータを操作閲覧するウェブアプリケーションが自動生成される．自動生成コードはスキーマの変更に追従して何度でも再作成できる．また，開発者が生成コードの機能拡張を行う場合，DBPowder-web の指定した拡張ポイントにコードを記述すると，再コンパイルしても機能拡張は失われない．これらの機能により，ウェブアプリケーション構築の簡素化と機能拡張性の両立を実現する．

本論文の構成は次の通りである．2 章で従来手法について述べ，3 章で DBPowder-web を提案する．4 章で評価を行い，5 章で結論とする．

2. 従 来 手 法

本章では，関連研究とウェブアプリケーション構築技術の現状を述べる．

2.1 関 連 研 究

ウェブアプリケーションの構築では，Smalltalk における Model-View-Controller programming [10] をウェブアプリケーション構築に応用した，モデル-ビュー-コントローラアーキテクチャ [9](以下 MVC) が広く用いられる．MVC では，オブジェクト間の結合度を抑えるために，各コンポーネントをモデル (M)，ビュー (V)，コントローラ (C) の 3 つに分離する．ここで，モデルはアプリケーションが扱うデータを保持し，データを扱うためのロジックを司る．ビューはモデルのデータを表示し，ユーザからの入力を受け付ける．コントローラはモデルやビューと連携し，アプリケーションの動作を定義する．

Celi らは Web Modeling Language(WebML) [1] を提唱しており，製品化を行っている [13]．WebML では，MVC のそれぞれに対して M:Structural Model, V:Composition Model, C:Navigation Model と対応付け，そのそれぞれに対して XML による記述言語を定義している．また，これらの記述言語を用いたウェブアプリケーション構築を支援する CASE ツールを，製品として提供している．

Rode らは Click [8] を提唱しており，phpClick という名称でオープンソースを公開している [15]．Click では，プログラミング作業へのツールの関与度を “gentle slope of complexity” として Layer1〜Layer7 に分類しており (**表 1**)，特にスキルの低い開発者がウェブアプリケーションを構築できるフレームワークを目指している．

遠山らは SuperSQL [14] を提唱しており，ツール一式をダウンロード可能である [18]．SuperSQL は SQL の拡張言語であり，クエリの結果である表データに対する段組などの整形を定義する．SuperSQL を用いると，表データを html, LaTeX, Excel などのデータ形式に整形して出力することができる．遠山らは，SuperSQL に基づいた様々な応用も提唱している．例えば，表データの表示レイアウトの最適化を行うもの [19] や，携帯端末のような限られた表示スペースでの表閲覧手法 [20] などがある．

2.2 ウェブアプリケーション構築技術の現状

ウェブアプリケーション構築では，様々な開発技術が用いられる．開発用プログラミング言語としては，以前は perl が主流だったが，1998 年 6 月の PHP3.0 リリースや，1999 年 12 月の tomcat3.0 リリースなどに伴い，PHP やサーバサイド Java が用いられることが多くなっている．他には，Ruby, Python, VBScript などが用いられる．

ウェブアプリケーションにおける MVC の全てを単一のプログラミング言語のみでカバーするのは，不可能ではないが適切ではない．ビューにおける html や CSS, JavaScript のコードをプログラミング言語における `echo`, `print` 文などで表示させるのは，可読性に欠ける．また，プログラミング言語を用いるようなウェブアプリケーションでは，扱うデータについて ACID 属性の保証が要請されるケースが多く，この場合 RDBMS を用いずにデータを適切に管理するのは困難である．また，フレームワークを用いないウェブアプリケーションの構築では，クロスサイトスクリプティングや SQL インジェクションなどの脆弱性に自前で対処する必要があるが，全ての html や SQL などの必要箇所にミスなく適切に処置するのは容易ではない．

従って，ウェブアプリケーションの構築では通常，MVC のコンポーネント毎に適材適所の技術を用いることが多い．以下，MVC のそれぞれで用いられる技術について述べる．

モデルの開発では，プログラミング言語と RDBMS を連携して用いることが多い．RDBMS は保持するデータの ACID 属性を実現するため，モデルの開発で RDBMS を用いるメリットは大きい．

一方で，リレーショナルモデルとプログラミング言語とのインピーダンスミスマッチが問題となる．このインピーダンスミスマッチのために，モデルの開発は難易度が高く，熟練プログラマを必要とするケースが多い．インピーダンスミスマッチ解消の手法として，O/R マッピングツールの使用が挙げられる．商用では Oracle 社の TopLink [3]，オープンソースでは Hibernate [7]，他に Enterprise JavaBeans の CMP [11] などが知られる．

ビューの開発では，プログラミング言語と併用して html テンプレートを用いることが多い．html テンプレートでは，アプリケーションによって動的に変更される要素を変数などで記述し，他の静的な要素を通常の html で記述する．

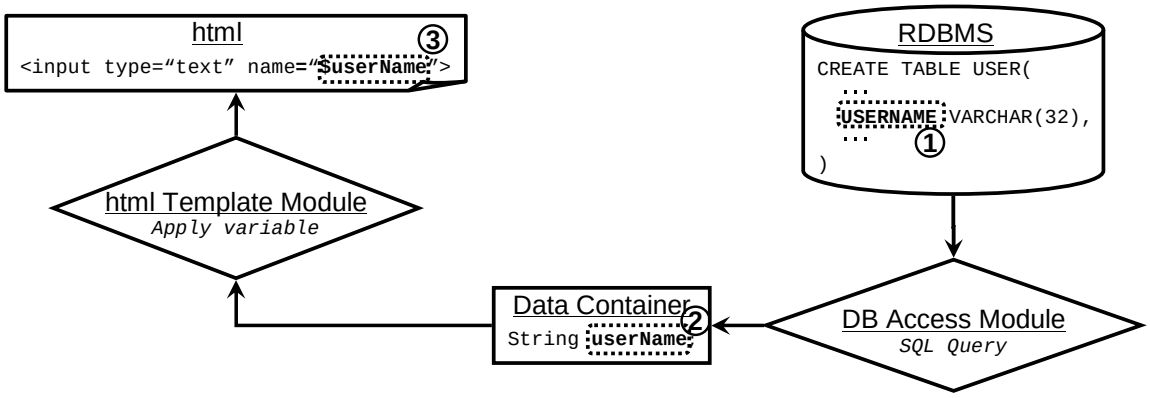

図 1　DB のデータ 1 フィールドを表示する単純な例

コントローラの開発では，主に通常のプログラミング言語が用いられるが，Struts [4]，JSF [12] などのように，ビューを含んだフレームワークを利用するケースが最近増えている．

2005 年の特徴的な動きとしては，MVC 全てを統括的にサポートするフレームワークの出現が挙げられる．一つには，Dependency Injection [2] とアスペクト指向プログラミング [2] を活用し，Struts や Hibernate などの既存フレームワークの保守性を高めることを意図した，Spring [17] や Seasar [5] が挙げられる．一方で，モデル駆動型開発を採用し，スキーマから CRUD(Create, Read, Update, Delete) 機能実現のウェブ画面を自動生成する，Ruby on Rails [6]，RIFE [16]，phpClick を挙げることができる．

以上，MVC のそれぞれで用いる技術について述べた．しかし，様々な技術を単純に組み合わせるだけでは，開発過程は複雑化し，アプリケーションの保守性も悪くなる．**図 1** に，DB のデータ 1 フィールドを表示する単純な例を示す．RDBMS の USER というテーブルにある USERNAME というフィールドの値を html に表示したいだけにも関わらず，少なくとも 5 つのコンポーネントが登場し，そのそれぞれの箇所でプログラミングが必要になる．関係するモジュールの部品点数が多いため，修正を加えた際の波及範囲が大きい．また，特にモデルではリレーショナルモデルとプログラミング言語のインピーダンスミスマッチが存在するために，開発の難易度は高くなる．

この問題については，現在でも解決を見ていない．MVC を総括的にサポートするフレームワークを用いても，これを解決することはできない．Spring や Seasar はこの問題を解決するためのアプローチを提供していない．Ruby on Rails, RIFE, phpClick は，ウェブアプリケーションの自動生成により初期開発の負担を軽減するが，その後カスタマイズを加えようとするとプログラマの負担は却って増大する．この問題については，**4.6 節**で再度議論する．

ウェブアプリケーションは，ユーザにとっては利便性が高い．ウェブアプリケーションはウェブブラウザさえあれば動作するため，プラットフォームに依存せず，追加インストール作業を必要としない．しかも，複数ユーザが同時利用できるアプリケーションとして動作する．このことから，インターネット，イントラネット共にウェブアプリケーションは普及している．しかし，本節で見たように，ウェブアプリケーションはシステムとしての複雑性と保守性の面で問題を抱えている．また，セキュリティや ACID 属性などを満たすためには，専門の知識または適切なフレームワークの使用が不可欠である．これを考慮に入れると，十分なスキルを持たない開発者が適切なウェブアプリケーションを構築するのは極めて困難であり，十分なスキルを持つ開発者にとってもウェブアプリケーションの構築は工数の多い作業になる．

3. 提 案 手 法

本章では，本研究で提案する，RDBMS を用いたウェブアプリケーションの構築を支援するフレームワーク DBPowder-web について述べる．

3.1 概　　要

我々は，データスキーマから CRUD 機能を持つウェブアプリケーションをスケルトンとして生成することで，RDBMS を用いたウェブアプリケーションの構築を支援するフレームワーク DBPowder-web を提案する．DBPowder-web の全体図を**図 2** に示す．

DBPowder-web に基づいたウェブアプリケーション開発では，まずはじめにデータスキーマを記述する．データスキーマ記述のために，モデル記述言語 DBPowder Model Description Language(DBPowder-MDL:**表 2**) が用意されている．

DBPowder-MDL を用いてデータスキーマを記述しコンパイルすることで，RDBMS のスキーマを生成するための SQL CREATE 文 (**表 3**) 及びスキーマ内のデータを操作閲覧するウェブアプリケーションを自動生成することができる．

データスキーマを変更するときは，DBPowder-MDL に変更を記述し，再コンパイルする．このとき，DBPowder-web によって書き換わる箇所 (以後 *powder-code*) と書き換わらない箇所 (以後 *user-code*) に区別される．プログラマは，*user-code* にコーディングすることで，ウェブアプリケーションをカスタマイズできる．*user-code* に追加記述したソースコードは，再コンパイルしても失われない．

powder-code は，O/R マッピングモジュールを中心とした，

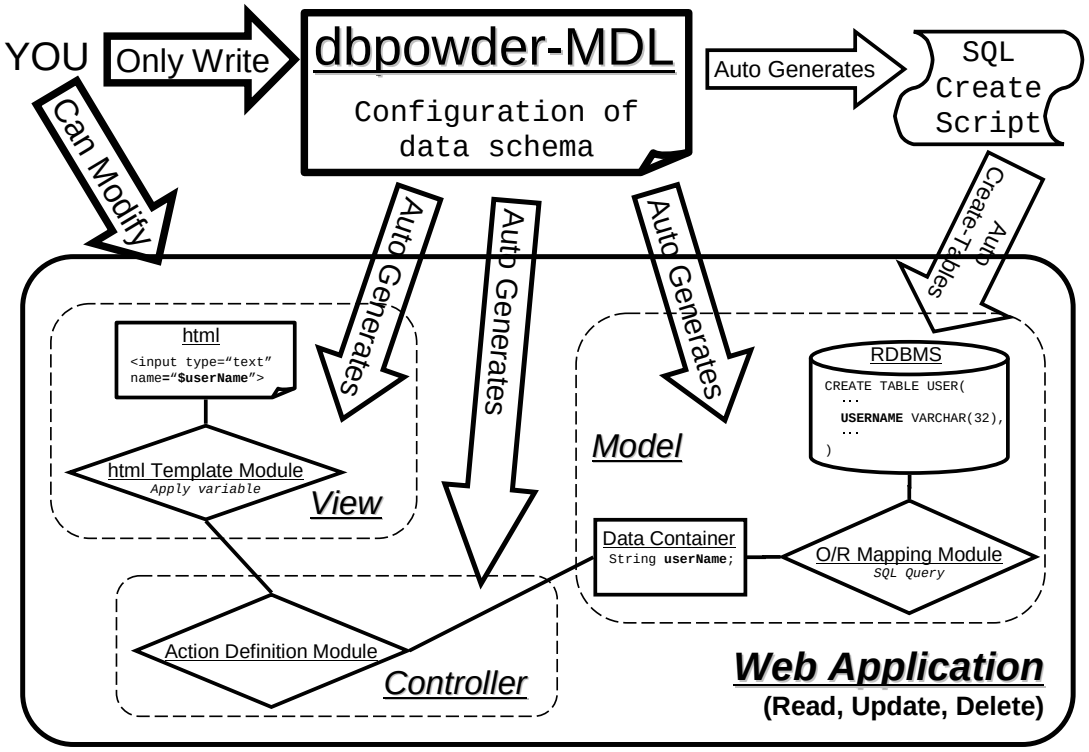

図 2　DBPowder-web 全体図

各々依存性の小さい機能別に構成される．また，これらのモジュールを利用する際に引き渡すパラメータは，スキーマの変更に対する影響が最小限にとどまるよう構成される．従ってプログラマは，*user-code* にコーディングする際に *powder-code* を再利用性の高い形で利用できる．これは，開発フェーズのみならず保守フェーズにおいても，ウェブアプリケーションが DBPowder-web の恩恵を受けられることを意味する．

DBPowder-web では，DBPowder-MDL に記述されたデータスキーマに基づき，ウェブアプリケーションの各部品を構成する．これは，**2.2 節**で述べた問題点のうち，部品点数の多さやインピーダンスミスマッチの問題を改善する．また，自動生成コードを *powder-code* と *user-code* に分離し，スキーマの変更の影響を *powder-code* に吸収する構造となっている．これは，**2.2 節**で述べた問題点のうち，保守性の改善に寄与する．

3.2　DBPowder Model Description Language(DBPowder-MDL)

本節では，DBPowder-web におけるモデル記述言語 DBPowder Model Description Language(DBPowder-MDL) について述べる．

DBPowder-MDL は，データスキーマを階層構造で記述する．( ) や [ ] でデータグループを定義し，そのデータグループが管理するデータを子要素として記述する．**表 2** に，DBPowder-MDL の例を示す．

( ) で表されるデータグループは，親要素に対して 1 対 1 又は 1 対多のリレーションシップを持つ．データグループ名の後ろに<=> を付与すると 1 対 1 を表し，データグループ名の後ろに< > による表記を付与しなければ，1 対多を表す．

[ ] で表されるデータグループは，親要素に対して多対 1 または多対多のリレーションシップを持つ．データグループ名の後ろに<+> を付与すると多対 1 を表し，データグループ名の後

```
[query_user_ans]
  number9   user_id
  text64    mail_addr
  (query)
  datetime  ans_datetime
  text64    user_agent
  text4096  freetext
  [query_user_ans_select<+>]
    (query_select)
    text512   user_select_comment
[query]
  text64    query_name
  text64    query_title
  text4096  query_text_top
  text4096  query_text_bottom
  number9   duplicate_num
  text4096  freetext_desc
  text128   mail_subject
  [query_select<+>]
    number9   select_id
    text4096  select_text
    number9   allow_text
```

表 2　DBPowder-MDL(例)

```
create table query_user_ans (
  query_user_ans_id int AI PK,
  query_id          int,
  user_id           int,
  mail_addr         varchar(64),
  ans_datetime      datetime,
  user_agent        varchar(64),
  freetext          varchar(4096)
)  ENGINE=InnoDB DEFAULT CHARSET=utf8;

create table query (
  query_id          int AI PK,
  query_name        varchar(64),
  query_title       varchar(64),
  query_text_top    varchar(4096),
  query_text_bottom varchar(4096),
  duplicate_num     int,
  freetext_desc     varchar(4096),
  mail_subject      varchar(128)
)  ENGINE=InnoDB DEFAULT CHARSET=utf8;

create table query_select (
  query_select_id   int AI PK,
  query_id          int,
  select_id         int,
  select_text       varchar(4096),
  allow_text        int
)  ENGINE=InnoDB DEFAULT CHARSET=utf8;

create table query_user_ans_select (
  query_user_ans_select_id int AI PK,
  query_user_ans_id        int,
  query_select_id          int,
  user_select_comment      varchar(512)
)  ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

(PRIMARY PEY は PK と略記)
(AUTO_INCREMENT は AI と略記)

表 3　DBPowder-MDL(表 2) から自動生成された SQL CREATE 文

ろに<*> を付与するか或いは < > による表記を付与しなければ，多対多を表す．尚，最上層のデータグループは親要素を持たないので，( ) と [ ] は同じ意味を持つ．

データグループは重複定義を許す．重複定義された全てのデータは OR 演算によってマージされ，最終的に一つのデータグループとして認識される．

データグループを構成するデータは，

データ型　データ名

という書式で定義される．

DBPowder-MDL で定義された各々のデータグループに基づいて，RDBMS におけるテーブルが生成される．テーブルには，主キーとなるシリアル ID カラムが自動的に追加される．また，リレーションシップを持つデータグループの場合，外部キーが親要素または子要素に自動的に追加される．多対 1 リレーションシップの場合，外部キーは親要素に追加される．1 対 1 又は 1 対多リレーションシップの場合，外部キーは子要素に追加される．多対多リレーションシップの場合，多対多関係を表すための結合テーブルが別途作成される．

表 3 に，**表 2** を RDBMS のテーブルに変換した例を示す．

3.3　DBPowder-web におけるモデル駆動型開発

DBPowder-web を用いたウェブアプリケーションの開発は，モデル駆動型開発手法の一種ということができる．

DBPowder-MDL でデータスキーマを記述しコンパイルすると，RDBMS スキーマを構築するための SQL CREATE 文と，データスキーマに対して CRUD 操作を行うウェブアプリケーションが自動生成される (**図 2**)．DBPowder-MDL に基づいて自動生成されたコードは，CRUD 機能を満たす *powder-code* と，プログラマの独自実装を行う *user-code* に分離される．再コンパイルすると，*powder-code* は新しくなるが *user-code* は変更されない．

DBPowder-MDL はデータスキーマを表現する一方で，RDBMS とプログラミング言語の中間に位置し，O/R マッピング記述言語としての役割を果たしている．DBPowder-MDL に基づいて生成された O/R マッピングモジュールは，複数テーブル間の外部ジョインを自動的に構成する．プログラマは *user-code* からこの O/R マッピングモジュールを独自に利用することが可能であり，その際にパラメータを渡せば，where 句を与えることができる．結果，O/R マッピングモジュールが生成する SQL の where 句は，複数テーブルを結合するために自動的に構成されたジョインに，プログラマがモジュールに与えたパラメータを加えたものになる．尚，プログラマが O/R マッピングモジュールを利用する際には，内部ジョインも利用可能である．

コントローラ，ビューに当たる Action Definition Module と html Template Module も，同じく DBPowder-MDL に基づき自動生成される．これらのモジュールは，モデル部分として生成された O/R マッピングモジュールやデータコンテナを利用しつつ，各々の役割を果たす．DBPowder-MDL で定義したスキーマの構造は，Action Definition Module と html Template Module にはあらわれないように設計されてある．つまり，DBPowder-MDL が自動生成する Action Definition Module と html Template Module は，スキーマ変更の影響を受けない．従って，Action Definition Module と html Template Module は，*user-code* として扱われる．

4.　評　　価

本研究の妥当性検証のために，DBPowder-web のプロトタイプシステムを開発し，評価した．

4.1 節では，プロトタイプシステムの開発について述べる．**4.2 節**では，このプロトタイプシステムの適用事例について述

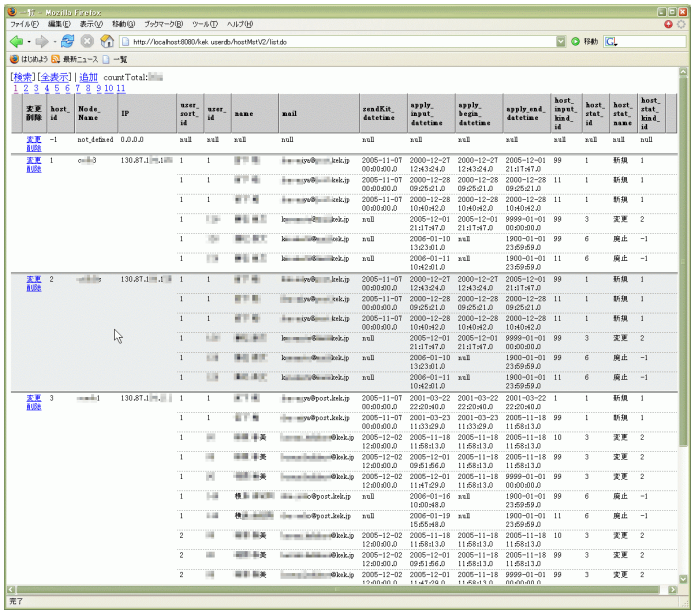

図 3　適用事例:ホスト情報管理アプリケーション

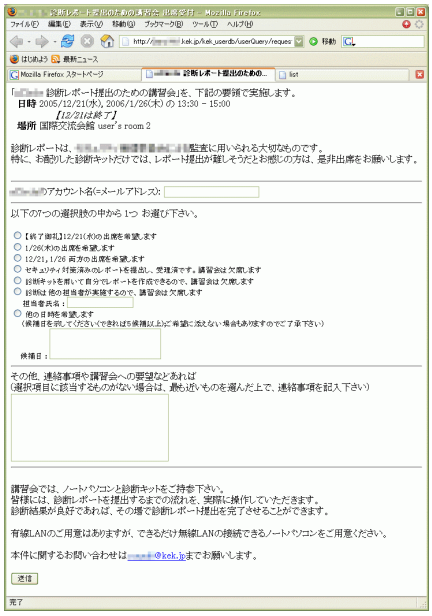

図 4　適用事例:アンケートアプリケーション

開発言語	Java 2 Platform, Standard Edition (J2SE) 5.0
RDBMS	MySQL 5.0
ウェブアプリケーションサーバ	tomcat 5.0
補助フレームワーク	Struts 1.2.8

表 4　DBPowder-web プロトタイプシステム 開発環境

べる．本研究では，実際にこのプロトタイプシステムを用いてウェブアプリケーションを構築し，実運用を行っている．**4.3 節**では，このプロトタイプシステムに基づいて，DBPowder-MDL の記述力を評価する．**4.5 節**では，このプロトタイプシステムにおけるシステム全体の複雑度と保守性を評価する．最後に**4.6 節**で，従来手法との比較を行う．

4.1　DBPowder-web プロトタイプシステムの開発

本研究では，**3 章**で提案した DBPowder-web フレームワークのプロトタイプシステム開発を行った (**表 4**)．開発言語に Java 2 Platform, Standard Edition (J2SE) 5.0 を採用した．RDBMS に MySQL 5.0，ウェブアプリケーションサーバに tomcat5.0 を採用した．また，ビュー，コントローラ用のフレークワークとして Struts1.2.8 [4] を利用した．

このプロトタイプシステムは，**3 章**で述べた各機能を実現する．DBPowder-mdl を用いてスキーマを記述し，プロトタイプシステムでコンパイルすると，このスキーマ内のデータを操作閲覧するアプリケーションが自動生成され，同時にこのスキーマを満たすリレーションが RDBMS 内に自動生成される．

4.2　DBPowder-web プロトタイプシステム適用事例

4.1 節で述べた DBPowder-web のプロトタイプシステムを用いて，ウェブアプリケーションを構築した．このウェブアプリケーションは，実際の運用で用いられている．本節では，そのうちの 2 例を示す．一例はホスト情報管理アプリケーションであり (**図 3**)，もう一例は利用ユーザへのアンケートアプリケーションである (**図 4**).

ホスト情報管理アプリケーション (**図 3**) は，高エネルギー加速器研究機構で管理するホストの情報を閲覧編集するためのものである．ホストとその所有者の情報を管理するために，5 つのテーブルを用いている．なお，このアプリケーションは，DBPowder-web を用いて自動生成されたスケルトンそのものである．**図 3** は，このうち閲覧画面である．各行の情報は，ホストごとにまとめられている．データの件数が多いときは，20 件ごとに表示される．このときは，画面左上と左下にあるページャーを使って前頁や次頁に移動できる．各行にある「変更」「削除」リンクをクリックすると，その行のホスト情報を編集または削除できる．

利用ユーザへのアンケートアプリケーション (**図 4**) は，利用ユーザにアンケートを実施し，集計結果を RDBMS に格納する．アンケートの質問文そのものも RDBMS に格納する(注1)．各ユーザは，このウェブアプリケーションで必要項目を選択し，必要に応じ文章を入力し，送信ボタンを押す．このユーザの入力は，その場で RDBMS に格納される．従ってアンケート実施者は，ユーザの最新の回答内容を，集計結果とともにいつでも閲覧できる(注2)．このアプリケーションを構築するために，**表 2** に示した DBPowder-MDL を記述し，DBPowder-web を用いてスケルトンを作成し，その生成物に対して *user-code* の修正を行った．

4.3　DBPowder-MDL の記述力の評価

本節では DBPowder-MDL の記述力を評価する．

テーブル間のリレーションシップは，1 対 1, 1 対 n, n 対 m の

(注1)：本論文では詳細は示さないが，この質問文を追加編集するためのウェブアプリケーションも，別途 DBPowder-web を用いて自動生成した
(注2)：本論文では詳細は示さないが，ユーザの回答内容を閲覧するためのウェブアプリケーションも，別途 DBPowder-web を用いて自動生成した

3 通りに分類されるが，DBPowder-MDL は，これらの全てをサポートする．リレーショナルスキーマ設計では，リレーションシップが n 対 m になる場合は結合テーブルを別途作成する必要があるが，DBPowder-MDL はこれを自動的に行う．3 テーブル以上のリレーションシップの作成が必要になる際には，結合テーブルを通常のテーブルとして定義した上で，対象テーブルへのリレーションシップをそれぞれ 1 対 1 で記述すれば良い．以上のことから，DBPowder-MDL は，テーブル間で発生するリレーションシップを記述する十分な能力があることを示せた．

本研究における DBPowder-web のプロトタイプシステムでは，各タプルの型は `number, text, datetime` のみのサポートになっているが，必要に応じてサポート範囲を広げることが可能である．

表 3 に，**表 2** に示した DBPowder-MDL から自動生成された `SQL CREATE` 文を示す．DBPowder-MDL を用いたスキーマ定義は，従来の `SQL CREATE` 文を用いたスキーマ定義と比べて記述が簡潔であり，可読性も高いと言える．このことは，プログラマへの負担を下げるばかりでなく，プログラミングに馴染みが薄い人に対する敷居を下げる効果も期待できる．

以上のことから，DBPowder-MDL はリレーショナルモデルを表現するための十分な記述力を保ちつつ，従来の `SQL CREATE` 文と比べて設計の負担を軽減できると言える．

4.4 システムとしての記述力の評価

4.1 節で示したように，本研究でのプロトタイプシステム開発は Struts [4] をベースに行った．DBPowder-web が作成するコードは全て Struts のフレームワークに差し込む形で組み込んだ．ユーザによる拡張を制限する *powder-code* には，必ず対応する形で *user-code* をラッパとして実装し，拡張ポイントを設けた．このように，DBPowder-web は，Struts での実装箇所に実質制限を加えていない．

従って，DBPowder-web をベースとしたシステムは，一般的なウェブフレームワークである Struts の記述能力を失っていない．

4.5 システム全体の複雑度と保守性の評価

図 2 に示したように，DBPowder-web では，簡潔なスキーマ記述言語 DBPowder-MDL をベースに，ウェブアプリケーションの MVC 全てを自動生成する．自動生成されるアプリケーションは，DBPowder-MDL で定義されたスキーマへの基本的な CRUD 機能を持つ．

プログラマは，DBPowder-web がウェブアプリケーションを自動生成した後に，必要に応じてビュー，コントローラに変更を加えることができる．モデルについては DBPowder-web が O/R マッピングを行っているので，SQL を記述しなくても RDBMS を扱うことができる．モデルで複雑なビジネスオブジェクトを構築したい場合は，DBPowder-web による O/R マッピングオブジェクトをベースすることができる．

3.1 節で示したように，プログラマは，独自実装を行った後でも再コンパイルを自由に実行できる．*user-code* にプログラマが独自実装したソースコードは，DBPowder-web が再コンパイルしても失われない．従って，独自実装後でも DBPowder-MDL を用いてスキーマの変更を行うことができる．

4.3 節, **4.4 節**で示したように，DBPowder-web はシステム構築のための記述力を損なわない．

以上から，DBPowder-web は，システム全体としての複雑度を下げ，保守性を高めることができると言える．

4.6 従来手法との比較

本節では，DBPowder-web を用いたウェブアプリケーション構築と，従来手法を用いたウェブアプリケーション構築を比較する．

従来手法の多くは，**2.2 節**で**図 1** を例に挙げて述べた，開発過程の複雑さと保守性の悪さを抱えている．DBPowder-web では，**4.5 節** で述べたように，これらの問題を大幅に改善する．

以下，DBPowder-web を構成する各技術要素と，そのそれぞれに対する従来手法を比較する．

インピーダンスミスマッチ解消の手法として O/R マッピングツールの使用が挙げられる [3] [7] [11]．O/R マッピングツールは，**図 1** の DB Access Module を自動生成するため，モデルの開発におけるプログラマの負担は軽減される．しかし多くの場合，RDBMS のスキーマそのものは自前で構築する必要があるばかりでなく，XML などの煩雑な設定ファイルを記述せねばならない．一部には，ツールによっては XML の設定ファイルから RDBMS のスキーマとプログラミング言語によるオブジェクトを同時に生成するものもある．DBPowder-web では，簡潔なスキーマ記述言語 DBPowder-MDL のみの記述から，ソースコードとスキーマの両方を構築する．さらに，スキーマを操作するウェブアプリケーションも，自動的に生成する．

Ruby on Rails [6] ，RIFE [16]，phpClick [8] といった統括的フレームワークは，ウェブアプリケーション構築の工数を下げるが，実装できる機能が制限されてしまう．フレームワークが生成したソースコードそのものに手を加えることは不可能ではないが，フレームワーク全体に対する深い理解が必要になる．結果として，カスタマイズの度合いが大きくなればなるほど，フレームワークから受ける恩恵は小さくなり，開発工数は大きくは下がらない．また，これらのフレームワークは，RDBMS 環境の変化などを反映させるために一旦構築したウェブアプリケーションを再構築する機能を持つものが多いが，一旦カスタマイズすると，この再構築は容易ではない．DBPowder-web では，**4.4 節**で述べたように，フレームワークとして少なくとも Struts [4] と同等の記述能力を有している．また，**4.5 節**で述べたように，再コンパイルを実行しても *user-code* は変更されない．

SuperSQL [14] を用いると，ビューを中心にしてウェブアプリケーションの使い勝手を上げることができる．DBPowder-web は，モデルを中心としつつも MVC 全体を対象としたフレームワークである．

Microsoft .NET Framework， Oracle JDeveloper, IBM Rational Web Developer for WebSphere Software, WebRatio [13] といった商用ツールは，洗練された統合開発環境を提供する．しかし，ウェブアプリケーション構築手法そのものが目まぐるしく変化している現状では，特定の商用開発環境をマスターするための

学習は必ずしも効率的ではなく，また，商用ツールそのものの統合開発環境そのものが目まぐるしく変化しているのが現状である．また，Oracle や IBM などの開発環境では，フレームワークのコア部分は Struts や Hibernate [7] などのオープンソースと連携するケースが多い．DBPowder-web では，プログラマが新規に学習するべき事項は DBPowder-MDL のみであり，生成されるコードは，Struts のオープンソースによるコードそのものである．

尚，本研究ではフレームワークとしての生産性に注力したため，パフォーマンスの評価は実施していない．但し，RDBMS を用いたシステムではパフォーマンスは重要な評価指標なので，将来的にはパフォーマンス向上も視野に入れることを構想している．なお，本節で挙げたフレームワークの中では，Hibernate は O/R マッピングツールとしてのパラメータが多く用意されており，これらのパラメータについて熟知すれば，高いパフォーマンスを発揮できると考えられる．

5. 結　　論

本研究では，RDBMS を用いたウェブアプリケーションの構築を支援するフレームワーク DBPowder-web の提案を行った．提案に基づいてプロトタイプシステム開発を行い，ウェブアプリケーションを構築した．このアプリケーションは，実運用に供している．

ウェブアプリケーションは，ユーザにとっての利便性が高い一方で，適切なアプリケーションを構築するには困難が伴う．特に，RDBMS とプログラミング言語の連携において，それが顕著である．

DBPowder-web は，簡潔なスキーマ記述言語 DBPowder-MDL からウェブアプリケーションのスケルトンを自動生成する．このスケルトンは RDBMS における CRUD 機能を有しているため，スケルトンそのものをウェブアプリケーションとして利用できる．

このスケルトンは，DBPowder-web によって変更されうる *powder-code* とプログラマが自由に追加変更できる *user-code* に分離されている．従って，スケルトン生成後も DBPowder-web を用いてスキーマの変更に追従することが可能であり，かつ，プログラマは独自の機能をアプリケーションに追加できる．

上記のことから，本研究で提案した DBPowder-web が，ウェブアプリケーションの構築の簡素化を実現しつつ，かつプログラマが最終的に構築しようとするアプリケーションの機能拡張を妨げないことを示すことができた．

6. 謝　　辞

本研究の遂行にあたり，高エネルギー加速器研究機構計算科学センター川端氏，金子氏，湯浅氏には，様々な形でご支援を頂きました．感謝致します．

文　　献

[1] Stefano Ceri, Piero Fraternali, and Aldo Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites. *Computer Networks (Amsterdam, Netherlands: 1999)*, Vol. 33, No. 1–6, pp. 137–157, 2000.

[2] Shigeru Chiba and Rei Ishikawa. Aspect-oriented programming beyond dependency injection. In *ECOOP*, pp. 121–143, 2005.

[3] Oracle Corporation(2006). TopLink. `http://otn.oracle.co.jp/products/ias/toplink/`.

[4] Apache Software Foundation(2006). Apache Struts Project. `http://struts.apache.org/`.

[5] Seasar Foundation(2006). Seasar – DI Container with AOP. `http://www.seasar.org/`.

[6] David Heinemeier and et al(2006). Ruby on Rails. `http://www.rubyonrails.org/`.

[7] JBoss Inc.(2006). Hibernate. `http://www.hibernate.org/`.

[8] J.Rode, Y.Bhardwaj, M.A.Pérez-Qui nones, M.B.Rosson, and J.Howarth. As Easy as “Click”: End-User Web Engineering. *Lecture notes in computer science, D.Lowe and M.Gaedke(Eds.): ICWE2005, LNCS 3579*, pp. 478–488, July 2005.

[9] Nicholas Kassem and Enterorise Team. *Designing Enterprise Applications: With the Java 2 Platform, Enterprise Edition (Java Series)*. Addison-Wesley Pub., Boston, MA, 2000.

[10] Glenn E. Krasner and Stephen T. Pope. A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. *J. Object Oriented Program.*, Vol. 1, No. 3, pp. 26–49, 1988.

[11] Sun Microsystems(2006). Enterprise JavaBeans Technology. `http://java.sun.com/products/ejb/`.

[12] Sun Microsystems(2006). JavaServer Faces Technology. `http://java.sun.com/j2ee/javaserverfaces/`.

[13] Web Models(2006). WebRatio. `http://www.webratio.com/`.

[14] M.Toyama. SuperSQL: An Extended SQL for Database Publishing and Presentation. *Proc. ACM SIGMOD*, pp. 584–586, June 1-4 1998.

[15] sourceforge.net(2006). phpClick. `http://phpclick.sourceforge.net/`.

[16] RIFE team(2006). RIFE. `http://rifers.org/`.

[17] www.springframework.org(2006). Spring Framework. `http://www.springframework.org/`.

[18] 遠山研究室 (2006). SuperSQL. `http://ssql.db.ics.keio.ac.jp/`.

[19] 亀岡慎平, 遠山元道. SuperSQL を用いた組版におけるレイアウト自動修正. 情報処理学会論文誌:データベース, Vol. 46, No. SIG 13, pp. 78–93, Sep 2005.

[20] 日高大輔, 遠山元道. 携帯端末でのブラウジングにおける表形式データの対話的変形. 第 16 回データ工学ワークショップ DEWS2005, pp. 3B–i8, March 2005.