B-017

データモデルと UI の相互変換を用いたウェブサービス開発環境「SoS」
SoS, a web service development environment by mutual conversion of data model and UI

中田宏昭† 深海悟†
Hiroaki Nakada Satoru Fukami

1. はじめに

SoS は我々が開発を進めているウェブサービスを開発するためのメタサービスである。

ブログや Wiki 等の CMS や、画像共有サイトなどの登場により、個人による情報発信が盛んに行われている。しかし、画像共有サイトやソーシャルブックマークといったウェブサービスそのものを発信することはそれほど一般的ではない。それは、ウェブサービスの作成にはプログラミングやデータベースの知識など必要とするためである。そこで、データモデルと UI の相互変換を行うことで、開発の敷居を下げ誰でもウェブサービスを公開・提供できるようにすることを目指している。

2. End User Developing（EUD）環境の要件

文章や動画などの静的コンテンツと違い、アプリケーションといった動的コンテンツを作成するためにはプログラミングが必要である。しかし、エンドユーザの多くはプログラミング技術を持っていない。そのため、プログラミングを必要としない EUD 環境が必要になる。ウェブサービス開発において EUD 環境が解決すべき課題として以下を考えた。

(1) データの永続化

通常、ウェブサービスではデータの永続化にデータベースが利用される。Blog、Wiki、ソーシャルブックマーク、画像共有サービスなど多くのサービスがデータベースへの追加・参照・更新・削除（CRUD）の操作を基本としている。しかし、ウェブサービスにおいてデータの永続化は最も基本的な処理であると同時に、最も煩雑な処理でもある[1]。

(2) 変更の煩雑さ

通常、UI の永続化要素に変更を加えた場合、他の UI やロジックの変更が必要になり、同じ様な変更を複数個所に施さなくてはならない。Ruby on Rails や Grails 等の scaffold も CRUD ページの自動生成により初期開発コストを軽減するが、その後のカスタマイズに関するコストは軽減しない[1]。

(3) 何が作りたいか明確になっていない

開発者が自分の作りたい物に対して漠然としたイメージしかなく、明確化できていない事がしばしばある。これは最初の作業の抽象度が高いほど問題となる。そのため、モデルやロジックから記述を始めるのは困難である。

3. 解決のアプローチ

これらの問題に対して、SoS ではデータモデルと UI の相互変換を行うことで解決する。具体的には次の様なアプローチを取る。

- UI からデータモデルへの自動変換
- データモデルから CRUD ページの自動生成
- データモデル変更の自動伝播
- 上記３つによる UI 駆動開発と進化型開発プロセスの支援

3.1 UI からデータモデルへの自動変換

多くの場合、モデリングするべきデータはシステム内のどこかで入力または出力の UI として現れている。そこで、UI 内の永続化したい要素にメタデータを埋め込むことで、UI を一種のスキーマ定義とみなしデータモデルの自動生成を行う。メタデータを埋め込んだ UI の例を以下に示す。

```
<form id="$$bookmark">
<p>名前<input id="$$名前" type="text" /></p>
<p>URL<input id="$$URL" type="text" /></p>
<p>コメント<input id="$$コメント"  type="text" /></p>
</form>
```

id 属性に埋め込まれた$$から始まる文字がモデリング用のメタデータである。モデリングに際してまずメタデータを抽出しツリー構造に変換する。このメタデータのツリーにおいて、リーフノードをプロパティ、それ以外のノードをエンティティとする仮データモデルを作成する。

この作業をすべての UI に対して行い、同一エンティティの仮モデルをマージすることで、UI からデータモデルへの変換を実現する。

UI からの自動モデリングにより、難易度の高いモデリングという作業を省くことができる。

3.2 モデルから CRUD ページの自動生成(scaffold)

データに対して行う基本的な処理として追加・参照・更新・削除（CRUD）操作がある。従来の Web アプリケーション開発ではそれぞれのデータに対してプログラミング等を用いてデータベースへの操作を行う。SoS では /persist/create, /persist/update, /persist/delete 等の URL にモデルに対応するパラメータを Post することで操作を行う。そのため、CRUD 操作に対するプログラミングは不要である。また、CRUD 用のページはモデルの種類にかかわらず、似たような構造になりやすい。そこで、データモデルから CRUD 用ページを自動生成することで、定型作業を省くことが出来る。

データモデルの作成により、この様なモデル駆動的アプローチが可能になる。同様の機能は、Ruby on Rails による scaffold をはじめ多くの Web フレームワークが提供している。

3.3 データモデル変更の自動伝播

scaffold はテンプレートによる自動生成のため、UI 生成のコストは軽減できるが、変更のコストは軽減されない。そこで、UI → モデル → UI という変換を行うことで変更

† 大阪工業大学 大学院情報科学研究科
Osaka Institute of Technology Graduate School of Information Science and Technology

の影響を自動的に解決し、変更のコストを軽減する。図 1 は new ページに URL 要素を追加したときの変更の自動伝播を示している。まず、new ページに(1)の要素が追加されると、その変更がモデルに反映され(2)が追加される。さらに bookmark を参照している UI を見つけ、モデルとメタデータの差分を調べる。差分があった場合、(3), (4), (5)の様に適切な要素を追加、または削除する。

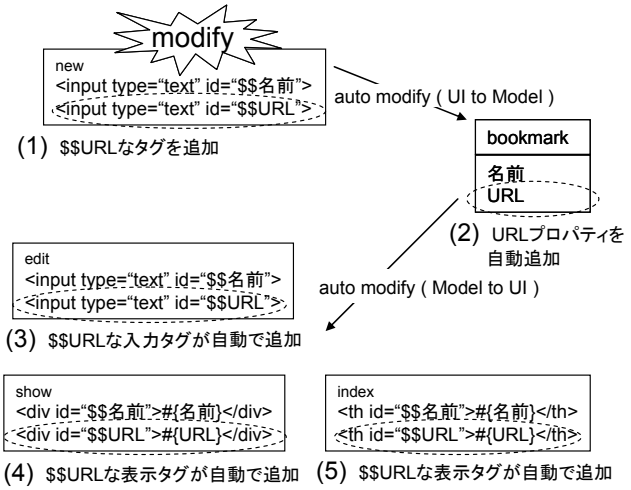

図 1. 要素追加時の変更の伝播

また、同じ要素でも UI 中で入力と表示の 2 種類の使われ方がある。そのため要素の追加時には、追加する箇所に合わせて UI コンポーネントの推論をする必要がある。

推論には同じモデルの要素は同じ UI 内で同様の操作がされやすいという点に注目する。まず、そのページ内で他の永続化要素が入力に使われているか表示に使われているかを調べる。そして、多い方の操作に合わせたコンポーネントを追加する。図 1 において(3)は入力用のコンポーネントを追加し、(4), (5) は表示用のコンポーネントを追加している。

3.4 UI 駆動開発

UI 駆動開発とは UI の記述を中心として開発を進めていく手法である。エンドユーザにとってモデルやロジックといった抽象度の高いものよりも、UI という視覚的に理解できるものの方が考えやすい。そこで、まず UI を記述し、そこからそれに付随するモデルやロジックを生成する。SoS では、UI からの自動モデリングと CRUD の自動化でこれを支援している。

3.5 進化型開発プロセス

開発の初期の段階では、システムに対する要求が明確化されておらず、開発者自身もどのようなサービスが欲しいか分かっていない事も多い。また、変化の早い Web の世界では次々と新しい要求が生まれて来るため開発に終わりは無い。この問題にはまず動くシステムを作り、イテレーションを繰り返すことで段階的に拡張していく進化型開発プロセスが有効である。これにより開発者は実際に動くシステムを触りながら要求を明確にすることができ、新しい要求への対応も容易になる。進化型の開発プロセスでは、その性質上変更が頻繁に行われる。SoS では、データモデル変更の自動伝播等によりシステムの変更に必要なコストを下げ、進化型開発プロセスを支援している。

4. 開発例

SoS を使った Web サービスの作成例として、ソーシャルブックマークの開発例を説明する。

イテレーション１ # とりあえず動くものを作る

- タイトルと URL が入力できる入力ページ(new)を作成する。この時にメタデータ [$$ブックマーク, $$タイトル, $$URL]等を適切なタグに付加する
- メタデータを元に"(ブックマーク(タイトル, URL))"というデータモデルが生成される（自動）
- scaffold で"(ブックマーク(タイトル, URL))"に対応した index, show, edit ページを生成する（自動）
- この時点でタイトルと URL の定型データに対する CRUD アプリケーションが出来上がる。これをソーシャルブックマークとしての体裁を整えるために、index ページの find タグの検索条件を重複する物を省く様に編集する

イテレーション２ # コメント機能を追加する

- show ページなどにコメント表示用のタグとメタデータ[$$コメント]を追加する
- モデルの内容が更新され"(ブックマーク(タイトル, URL,コメント))"になる（自動）
- モデルから UI の更新処理の実行により、new, edit, show にコメントに関するタグを追加する（自動）
- 細かいデザインを調整する

作成したソーシャルブックマークは図 2 の様になる。

図 2. 作成したソーシャルブックマーク

5. まとめ

SoS を用いることによって、専門的な知識を用いらずにデータベース連動型の Web アプリケーションを開発できるようになった。これにより様々なウェブサービスをユーザ自身が作り上げる土台を提供できた。今後もビジュアルエディタの開発等、より実用的な EUD 環境の実現に向けた研究を継続していく予定である。

参考文献

[1]村上 直, “DBPower-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”,第 17 回データ工学ワークショップ(DEWS2006)論文集, 4A-o4(2006).