プレゼンテーション層を 2 つに分離した Web 用超高速テンプレートシステム

1　背景

Web アプリケーションでは、動的な Web ページを生成する必要がある。例えば、ショッピングサイトならお勧め商品を毎日変えたり、ログインしたユーザごとにメニューを変えて Web ページを表示する必要がある。

動的に Web ページを生成する方法は主に 3 つある。

A)　メインプログラムの中に HTML を埋め込む（CGI プログラムや Servlet など）。

B)　HTML の中にメインプログラムを埋め込む（PHP や JSP など）。

C)　HTML ファイルをメインプログラムから読み込み、必要な部分を書き換えて出力する。

C)の方法はテンプレートシステムと呼ばれる。テンプレートシステムでは、テンプレートとなる HTML ファイルにあらかじめ「目印」を埋め込んでおく。そしてメインプログラムで HTML ファイルをテンプレートとして読み込み、「目印」がついた箇所を書き換えて出力する。

ここで大事なのは、A)や B)の方法ではメインプログラムと HTML ファイルとが一体化しているのに対し、C)のテンプレートシステムではこれらが分離しているという点である。メインプログラムと HTML ファイルとが分離していると、次のような利点がある。

- 互いに独立して変更できる。
  両者が分離されているため、一方を変更しても他方に影響を与えずにすむ。
  そのため、例えば HTML デザインだけを変更したいのに誤ってメインプログラムも変更してしまったということがなく、保守が容易になる。
- プログラマーとデザイナーの協業がしやすい。
  現在の Web アプリケーションでは、非常に複雑な HTML デザインを使用する。
  そのため、メインプログラムを担当するプログラマーとは別に、HTML デザインを担当する専門のデザイナーが必要となる。
  このような場合、メインプログラムと HTML ファイルが分離していると互いの作業がしやすくなり、開発作業が楽になる。
- 表示の切り替えが簡単にできる。
  例えばパソコンと携帯電話とで表示を切り替えたい場合、テンプレート（HTML ファイル）を切り替えるだけで済み、メインプログラムには一切変更する必要がない。

これらのような利点があるため、現在の Web アプリケーション開発ではテンプレートシステムを用いることが主流となっている。

しかし従来のテンプレートシステムには次のような問題点があった。

- テンプレートとなる HTML ファイルに対して、変更箇所を示す何らかの「目印」を埋め込むため、HTML デザインが崩れてしまう（Velocity や Template-Toolkit など）。
- 表示に必要なロジック（プレゼンテーションロジック）をメインプログラム側に記述しなくてはならない（XMLC や amrita など）。
- 内部で DOM やそれに類するツリー構造を使用するため、動作が遅く、重い。

- XMLやHTMLの生成のみを目的にしているものもあり、他のテキスト形式では使用できないことがある（XMLC や amrita など）。
- 複数のプログラミング言語で使用することを考慮していないため、開発言語が変わればテンプレートもすべて書き換える必要がある。
- セキュリティを考慮しておらず、メインプログラムでいちいちサニタイズを行わなければならない。

２　開発の内容

本システムは、従来のテンプレートシステムが持つ欠点を克服した、新しいテンプレートシステムである。具体的には、次のような特徴を持つ。

（1）テンプレートを「プレゼンテーションデータ（HTML ファイル）」と「プレゼンテーションロジック」とに分離する。

従来のテンプレートシステムでは、プレゼンテーションロジックを HTML ファイルかまたはメインプログラムの中に埋め込んでいた。

本システムでは、プレゼンテーションロジックを HTML ファイルからもメインプログラムからも分離した。これにより、プレゼンテーションロジックが HTML デザインを崩すことがなくなり、またメインプログラムの中にプレゼンテーションロジックが紛れ込むこともなくなった。

なお本システムでは、「プレゼンテーションデータ（HTML ファイル）」と「プレゼンテーションロジック」とをあわせて「テンプレート」と呼んでいる。

（2）出力用スクリプトを生成することで、高速かつ軽量に動作する。

従来のテンプレートシステムでは、内部で DOM やそれに類する木構造を使用するため、動作が遅くメモリ消費量も多かった。

本テンプレートシステムではテンプレートをあらかじめ出力用スクリプトに「コンパイル」し、実行時にはこの出力用スクリプトを実行するだけにした。これにより、高速かつ軽量な動作を可能にした。

（3）中間言語を採用することで、複数のプログラミング言語に対応する。

従来のテンプレートシステムでは、複数のプログラミング言語で使用することを考慮していなかった。そのため、複数の言語で使用するためには各言語ごとの実装を用意する必要があった。

本テンプレートシステムでは中間言語を採用することで、1 つの実装で複数の言語（Ruby、PHP、Java）に対応することができる。また Web アプリケーションの開発言語が例えば PHP から Java に変わったとしても、テンプレートはまったく変更する必要がない。

（4）テンプレートとなる HTML ファイルのデザインをまったく崩さない。

本テンプレートシステムでは、HTML ファイルの中にプレゼンテーションロジックを記述しなくてよい、つまり HTML ファイルの中に HTML 以外の要素を入れる必要がない。そのため、HTML デザインをまったく崩さない。

(5) すべてのテキストファイルで使用可能である。

従来のテンプレートシステムでは XML パーサーを使用したものが多い。しかし XML パーサーを使用すると、XML ファイルや HTML ファイルしか扱えなくなってしまい、他のテキスト形式が扱えない。

本テンプレートシステムでは、XML パーサーを使用せず独自のパーサーを使用し、任意のテキスト形式を扱えるようにする。

(6) セキュリティを考慮し、自動サニタイズ機能を備える。

従来のテンプレートシステムは、セキュリティを考慮していなかった。そのため、メインプログラムの中で明示的にサニタイズを行う必要があった。これを怠ると重大なセキュリティホールとなるが、実際にはプログラマーのスキルが低いためにサニタイズされないことも多々あった。

本テンプレートシステムでは自動的にサニタイズを行う機能を備えている。またすべてをサニタイズすることも、一部だけをサニタイズすることも可能である。

3 従来の技術(または機能)との相違

従来のテンプレートシステムは、大きく 2 種類に分類できる。

テキスト形式汎用タイプ

- テキスト形式一般を生成できるタイプのテンプレートシステム。
- プレゼンテーションロジックはテンプレート(HTML ファイル)に記述する。
- テンプレートに埋め込む「目印」は、主に独自形式。
- 代表例:Jakarta Velocity (Java)、Template-Toolkit (Perl)

XML/HTML 専用タイプ

- XML ファイルまたは HTML ファイルの生成に特化したテンプレートシステム。
- プレゼンテーションロジックはメインプログラムに記述する。
- テンプレートに埋め込む「目印」は、主に ID 属性を使用する。
- 代表例:Enhydra XMLC (Java)、amrita (Ruby)

これら従来のテンプレートシステムと本システムとを比べると、次のようになる。

	テキスト形式汎用タイプ	XML/HTML 専用タイプ	本システム
(1)プレゼンテーション層とビジネスロジック層とが分離できるか	○	×	○
(2)動作が高速か	×	×	○
(3)複数のプログラミング言語で使用できるか	△	×	○
(4)テンプレートの HTML デザインが崩れないか	△	○	○
(5)XML や HTML 以外でも使用できるか	○	×	○
(6)セキュリティを考慮しているか	×	×	○

(1) XML/HTML 専用タイプはプレゼンテーションロジックをメインプログラム中に記述しなければならない。そのためプレゼンテーション層とビジネスロジック層とが分離できない。
(2) XML/HTML 専用タイプでは内部で DOM を使用する。またテキスト形式汎用タイプも内部で DOM に類するツリー構造を使用する。そのため、どちらも動作が遅く、重い。
(3) テキスト形式汎用タイプでは、複数の言語で利用するためには各言語ごとの実装を用意する必要がある(逆にいえば、各言語ごとに実装さえすれば複数の言語で利用できる)。XML/HTML 専用タイプでは、プレゼンテーションロジックをメインプログラムと同じ言語で記述するため、そもそも複数の言語で使用することが根本的にできない。
(4) テキスト形式汎用タイプは、プレゼンテーションロジックをテンプレート中に埋め込む。つまり HTML ファイルの中に HTML ではない要素が入ってしまうため、HTML デザインが崩れてしまう。ただし、HTML コメントの中に埋め込むなど工夫すれば、HTML デザインをあまり崩さずに済む。
(5) XML/HTML 専用タイプは、そもそも XML や HTML のみを対象としているため、他のテキスト形式では使用できない。
(6) 従来のテンプレートシステムは(ごく一部を除いて)どれもセキュリティを考慮してない。

4 期待される効果

本システムは、プログラマーからみても Web デザイナーからみてもメリットが大きい。

◆ プログラマーからみたメリット
- Web デザイナーが HTML ファイルをどんなに変更しても、メインプログラムには影響がない。そのため、開発や保守が非常に容易になる。
- 実行時の動作が高速かつ軽量である。
- 自動的にサニタイズを行うことができるため、セキュリティホールを少なくできる。

◆ Web デザイナーからみたメリット
- HTML デザインがまったく崩れない。またデザインツールとも相性がよい。
- HTML デザインをどんなに変更しても、プログラマーに迷惑がかからない。
- 複数のプログラミング言語で使用できるため、開発言語ごとに異なるテンプレートシステムの使い方を覚える必要がない。
- プレゼンテーションロジックの記述が簡単なため、プログラマーに頼らなくてもデザイナー自身で変更や追加ができる。

5 普及(または活用)の見通し

現状ではリリースしたばかりであり、認知度はまだ低く、普及しているとは言いがたい。しかし海外では数名が実際のプロジェクトで利用し始めている。

6 開発者

桑田 誠 <kwa@kuwata-lab.com>

(参考)本システムのホームページ: http://www.kuwata-lab.com/webtech/kwartz/