Message from Vendors

Web開発はStrutsと自動生成ツールBlastJで

第2回：BlastJを使った実装におけるリアルなメリット

テキスト= CROSSFIRE JAPAN INC. 飯塚 友裕 IIZUKA Tomohiro http://crossfire.jp

はじめに

前回（Vol.27）は，主に設計概念を説明いたしました．設計と実装の関係について理解をしていただけたと思いますので，今回はいよいよ実装作業とテストについて書こうと思います．

Web系Javaプロジェクトの難しさ

●Web系Javaプロジェクトの実情

一般的に「Webプロジェクトは，本来はそんなに難しいものではない」ということをよく聞くのではないでしょうか？ ところが，実際にはスケジュールどおりにうまくいかないことがあったり，出戻り作業ばかりでなかなか前に進まなかったりと苦労することが多いのではないでしょうか．

一方，バッチ系のプログラムの実装ですとスケジュールどおりにいくという話もちらほら聞きます．しかし，実装の難易度的にはWebプロジェクトと比べると，バッチ系のプログラムのほうがWeb系のものよりも高いケースが多いのではないかと思われます．

これら2つの話は，一見矛盾した話ではありますが事実でもあります．では，なぜそうなってしまうかというと，「Webプロジェクトは，本来はそんなに難しいものではない」の「本来は」という部分が実は曲者で，いわゆるリアルなWebアプリケーションの実装における「難しい部分」を無視しているからだと言えるのではないでしょうか．そこで，そのWebプロジェクトの「難しい部分」とは何なのかを考えていきたいと思います．

●人間は単純作業が大の苦手

単純作業に分類される作業量の多さによる弊害は，意外とさまざまな現場で軽視されがちです．プロジェクトを成功させるには，やはり，プログラムを組むのは人間であるということを考慮していかなければなりません．

本来，人間は機械的な単純作業が苦手です．たとえば，2桁の足し算を筆算で行うのは簡単にできます．しかし，100桁の足し算となるとかなり「難しく」感じるのではないのでしょうか．この場合の「難しい」というのは，「足し算の方法」が難しいと感じるわけではありません．1桁の足し算と繰り上げがわかればよいので「足し算の方法」という意味では簡単です．つまり「人間が単純作業を淡々と続けること」が難しいと感じるわけです．

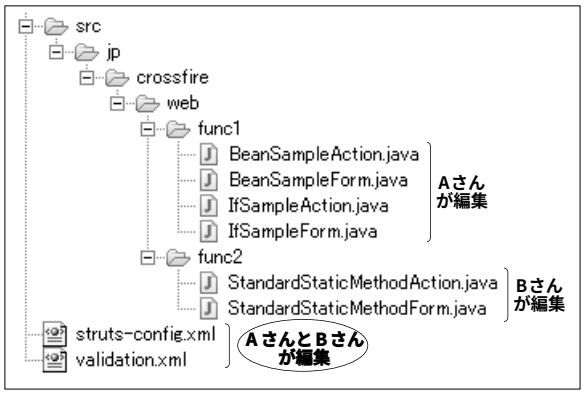
Webプロジェクトの「難しさ」というのも，実は，このような単純作業の多さを「難しい」と感じるのが多いのではないかと思います．また，人間は1つのことに気をとられるとほかのことに頭が回らないという性質があるので，初心者の方がかなり苦戦する原因にもなっているのではないかと思います．

本連載でご紹介する「BlastJ」では，この単純作業の部分をコンピュータに代りに行わせているため，単純作業を大幅にカットすることができます．

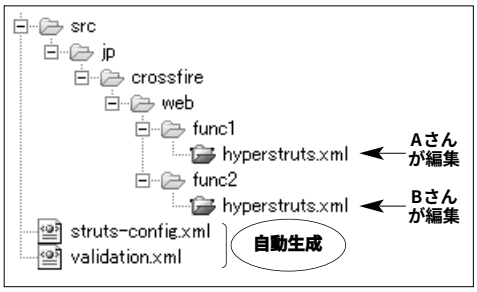
●チーム作業の難しさ

Strutsフレームワークを使ったWebプロジェクト

●図1 通常作業の場合



●図2 BlastJを利用した場合



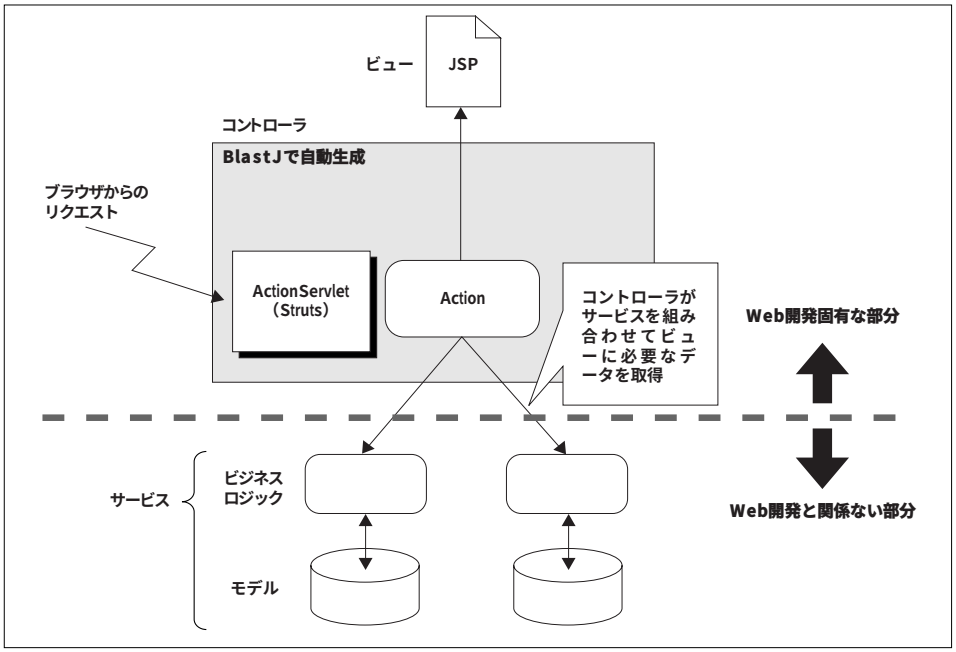
は，作業量が多くなる傾向があるため複数の人間で作業することのほうが多いのではないでしょうか．1人で作業しているときには問題にならなくても，複数の人間で作業するとさまざまな問題が発生します．

その最たるものが「struts-config.xml」や「validation.xml」などの設定ファイルの扱いです．これらの設定ファイルは，複数の人間が編集する必要があるため，1人が編集している最中にほかの人が編集することができません．また，担当者ごとに担当部分を開発して，最終的に誰かが結合するようなケースでも，結合作業でさまざまな問題が発生します．

共有しながらの編集が発生する具体的な例をとして，Aさんがパッケージfunc1，Bさんがパッケージfunc2の中のプログラムを担当する場合を考えます．通常の場合，AさんとBさんは自分が担当するパッケージ内のソースコードを個別に編集しますが，Strutsの設定ファイル（XMLファイル）は2人が共有して編集することになり，先ほど述べた問題が発生してしまいます（図1）．

一方BlastJを利用した場合には設定ファイルが自動生成されるため，AさんとBさんが共有して編集しなければならない部分はなくなります（図2）．

●図3 Web開発固有な部分と関係ない部分の境界



●つまるところStrutsでの開発の難しさとは

以上のことから，StrutsによるWeb開発特有の難しさは，理論を考えたりする難しさではなく，むしろ，設計を現場で実装する際の作業の難しさだということがわかります．また，Strutsを使ったMVCモデルのWeb開発ではWebアプリケーションに固有な部分とそうでない部分の境界がはっきりしています．ですので，逆に言ってしまえば，図3のようにWeb開発固有の難しささえ克服する方法を見つけてしまえば，Web開発は比較的簡単だということになります．

確実なWebアプリケーションの作成

●Webアプリのテスト作業の現状

どんな開発にも必ずテストは必要です．そのたびにいつも問題になるのはテストの正確さと作業量の多さだと思います．よくあるのが，全ケースごとのテストケースを作ってテストを行う場合です．それだけでもかなり多いですが，変数の境界値チェックも含めると，工夫なしではかなり膨大になってしまいます．特にWebアプリケーションではこのテストケースが膨大になるという傾向があり，厳密にやっていくと現実的には無理な量になってしまうケースも実際の現場ではあるのではないでしょうか．

●テストの効率と役割分担設計

前回，BlastJが推奨する4層設計の基本概念について述べました．その基本概念で最も重視していることは，4つの層それぞれの受け持つ役割がしっかりと決まっている役割分担設計です．実は，この役割分担設計の概念はテストの効率化および正確さにも非常に深い関係があるのです．そこで，今回はWeb開発特有の部分にフォーカスを当てますので，コントローラ部分のテストに関して述べようと思います．

●BlastJでの実装

BlastJではActionクラスの中身を自動生成する際に，基底となるクラスを指定することができます．実際にBlastJが自動生成するActionクラスは，リスト1の基底クラスを継承することになります．この基底クラス（リスト1）では，テンプレートメソッドというデザインパターンを採用しており，このクラスを継承したクラスで，このテンプレートメソッドを自動生成することになります．

これから，この2つのメソッド（execute( )，templete( )）の「役割と責任」を把握していきます．そして，その部分の責任にフォーカスをしてテストケースを作っていけば効率的で正確なテストができると言えます．

●execute( )メソッドの役割と責任

execute( )メソッドは，StrutsのActionクラスにおける処理のメインの部分を記述するメソッドです．実際のコードの中では，template( )メソッドを呼び出して処理を丸投げしています．つまり，execute( )メソッド自体はtemplate( )メソッドの実行結果の管理が主な役割になります．実行結果の管理の中でも，この部分で例外の管理を集中させることに非常にメリットがあります．template( )メソッドの代わりにexecute( )メソッドに例外処理の責任を負わせることによって，templete( )メソッドは正常ケースのロジックに対して責任を持てばよくなります．

Java言語の例外処理であるtry/catchの構文は，このような役割分担をさせる場合に非常に便利です．

●template( )メソッドの役割と責任

templete( )メソッドは，Actionの処理の本体です．また，この部分はBlastJによって自動生成される部分です．前述のようにexecute( )メソッドを工夫することによって，「正常系の操作がきちんと実行することを確認できれば大丈夫」というように設計することができます．自動生成される部分のテスト負荷が非常に小さいというのはBlastJのとても大きなメリットの1つになります．

●テストポリシーのまとめ

表1にexecute( )とtemplate( )メソッドについてまとめます．各クラスやメソッドの果たすべき役割に注目することにより，より効率良くテストができるプログラムを作成することが可能になります．特に

●リスト1　Actionクラスの基底クラス

```
public abstract class ExtendedAction extends Action {
    /**
     * executeメソッドをオーバーライドします
     */
    public ActionForward execute(ActionMapping mapping,
                                 ActionForm actionForm,
                                 HttpServletRequest request,
                                 HttpServletResponse response) throws Exception {

        Hashtable local = new Hashtable();
        try {
            // テンプレートクラスを呼びます
            templeteMethod(actionForm, request, response, local);
        } catch(Exception e) {
            // 独自のエラー処理を書きます
            ErrorHandler.handleError(this, mapping, actionForm, request, response);
        }

        return mapping.findForward((String)local.get("forward"));
    }

    /**
     * テンプレートメソッドは，BlastJでActionクラスごとに自動で実装させます
     * このクラスでは，Abstractにしておきます．
     */
    public abstract void templeteMethod(ActionForm actionForm,
                                        HttpServletRequest request,
                                        HttpServletResponse response,
                                        Hashtable local) throws Exception;
}
```

●表1　execute()メソッド／template()メソッド

メソッド	役割	前提条件	果たすべき責任
execute()	template()メソッドを呼び出し，例外を捕捉する	templete()の単体テストがきちんとなされていて，例外発生時に正しく例外を投げる	正しくtemplete()の前後の処理を行うこと．特に，例外処理をきちんと行うこと
templete()	ビジネスロジックを組み合わせて，ビューに必要なデータを取得する	ビジネスロジックの単体テストがきちんとなされていて，例外発生時に正しく例外を投げる	正常系の動作（アプリケーションロジック自体）が正しいことを保証する

execute()メソッドは基底クラスの1箇所にしか存在せず，正常の動作と例外の動作を独立して管理できるため，非常にテスト効率が良くなります．

簡単な例として，正常系のテストパターンが4パターン，異常系が5パターンあったとすると，

＜正常系と異常系とのテストパターン＞

- 非独立の場合　　4×5＝20パターン
- 独立している場合　　4＋5＝9パターン

となります．この数値を見ると，独立性のあるプログラムを書いたほうがはるかにテストが簡単になることがわかります．

まとめ

今回は，BlastJを使ってStrutsフレームワークによるWebアプリケーションを作る際の作業やテスト方針について述べました．BlastJは非常にシンプルなツールですが，「実装からテストまで非常に考えられて作られたツール」だということをわかっていただけたのではないかと思います．

次回は，これまでのまとめとして，実際に弊社が請け負ったプロジェクトに関する事例をご紹介したいと思います．

問い合わせ先

CROSSFIRE JAPAN INC.
〒160-0023
東京都新宿区西新宿7-23-9
西新宿小林ビル2F（アルディートシステム内）
TEL：03-5386-9921　　FAX：03-5386-3911
URL：http://crossfire.jp