Struts の代替としての JavaServer Faces の検証

安藤友晴●メディアとソフトウェア

●要約

Java を使った MVC モデル 2 による Web アプリケーションを開発するうえで、JavaServer Faces (JSF) は Struts の代替となりうるか検証した。検証にあたっては、Struts で作成したサンプルアプリケーションと同一の機能を持つものを JSF でも実装して、両者の実装方法を比較した。開発したアプリケーションは、JSF のタグライブラリの制約のため画面の構成が若干異なりつつも、両者ともほぼ同一の機能を持っている。Struts で作成したアプリケーションと比較すると、JSF で作成したアプリケーションは次の 3 点が優れている。まず第一に、JSF では画面の遷移を簡明に書ける。第二に、JSF では JSF EL が導入された JSP を利用できる。第三に、JSF では セッション管理に関するプログラムを記述しなくてもよい。こうした特長により、JSF を用いることで Web アプリケーションのプログラム開発効率の向上が期待できる。文字化けが発生するという問題を抱えているが、その点さえ解決できれば JSF は Struts の代替技術となりうるという結論を得た。

●キーワード

Web アプリケーション

MVC モデル 2

画面遷移

Struts

JavaServer Faces (JSF)

タグライブラリ

※2004年 1 月14日受理(紀要編集委員会)

1　はじめに

Java を使った MVC モデル 2 による Web アプリケーションの開発では、Jakarta Project で開発されている Struts(1)がよく利用されている。その一方で、Web アプリケーションのユーザインタフェースを簡単に開発するための技術として、JavaServer Faces（以下 JSF と略す）(2)がある。JSF は Java の標準化団体である Java Community Process（JCP）の JSR-127(3)で仕様策定が進んでおり(4)、今後の Java 2 Enterprise Edition（J2EE）における標準技術の一つとなる予定である。また、Sun Java Studio Creator(5)や WebSphere Studio V5.1.1 のような、GUI ベースでの Web アプリケーション開発ツールでも JSF の採用が予定されている。

Struts の作者である Craig McClanahan が JSF の開発の中心メンバーの一人でもあるためか、Struts と JSF の機能には共通点が多い。では、Web アプリケーションを開発するうえで、JSF は Struts の代替となりうるのだろうか？ もし Struts で提供されている機能が JSF でも利用でき、かつ開発効率の点で見劣りしないものならば、これまで Struts で開発していた Web アプリケーションを、今後の J2EE の標準技術となる JSF を用いたものに置き換えることができるだろう。

本研究では、Struts で作成したサンプルアプリケーションと同一の機能を持つものを JSF でも実装して、アプリケーションの構成要素ごとに両者の実装方法の違いを比較し、JSF が Struts の代替となりうるのかどうかを検証する。

2．検証を行う環境

サンプルアプリケーションの作成と検証にあたっては、以下の環境を用いた。

・JavaServer Faces	v1.0 Reference Implementation Beta
・Struts	1.1
・Web コンテナおよび Web サーバ	Tomcat 5.0.16

3．サンプルアプリケーションの概要

検証に用いるサンプルアプリケーションとして、「図書検索アプリケーション」を Struts と JSF でそれぞれ実装した。

このアプリケーションは、３つの Web ページから構成されている。Web ページは、図１から図２に、そして図２から図３へと遷移する。

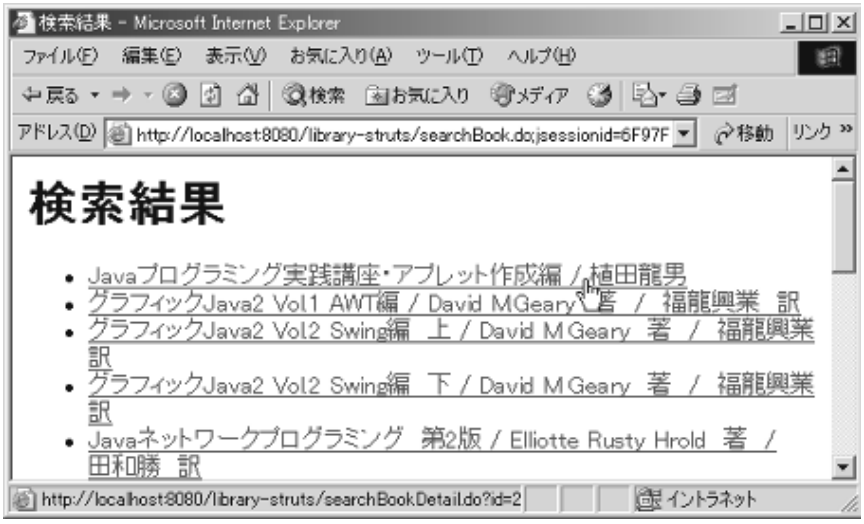
まず、「検索語入力画面」（図１）に、調べたい図書データの検索語を入力する。図１では、入力フィールドに "Java" という検索語を入力している。図１で "Go!" ボタンを押すと、入力された検索語についてリレーショナルデータベースの検索を行う。

検索の結果は、「検索結果のリスト表示画面」（図２）に表示される。図２では、１行につき１冊の図書データが表示されている。任意の行をクリックすると、その図書に関する詳細な図書データを表示させるために、再度データベースの検索を行う。

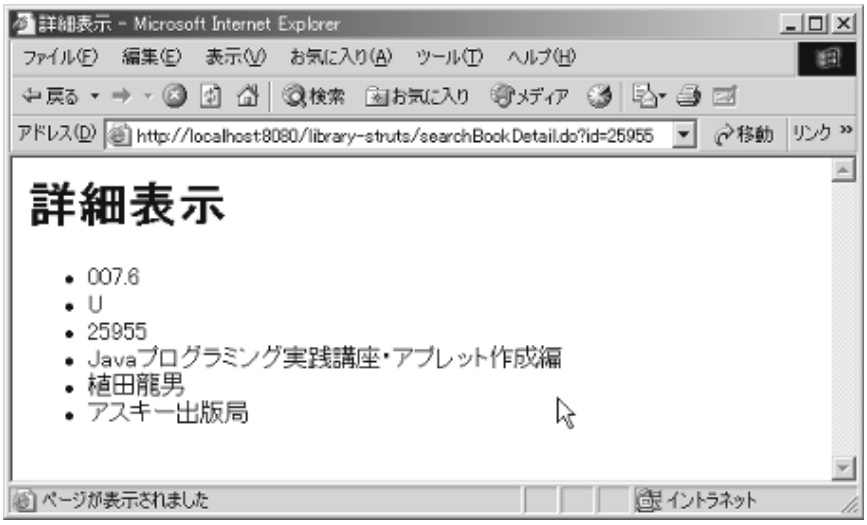
詳細な図書データは、「検索結果の詳細表示画面」（図３）に表示される。

サンプルアプリケーションの完全なソースコードは、筆者の Web ページ（http://www.wakhok.ac.jp/~tomoharu）

【図１　検索語入力画面】



【図２　検索結果のリスト表示画面】



【図３　検索結果の詳細表示画面】

から取得できる。

4．モデル

● Struts

Struts はモデルの実装には関与しない。ただし、JavaBeans の規約(6)に従ってモデルを実装することで、タグライブラリを用いた JSP からのモデルへのアクセスが容易になる。このサンプルアプリケーションでは、次のような JavaBeans をモデルとして用いている。

```
public class BookData implements Serializable {
    ......
    public String getId(){
        return id;
    }
    public void setTitle(String id){
        this.id =id;
    }
    public String getTitle(){
        return title;
    }
    public void setTitle(String title){
        this.title =title;
    }
    ......
}
```

この JavaBeans のプロパティは、リレーショナルデータベースのテーブルの項目と一致している。id プロパティは、テーブルの主キーとなっている。

● JSF

Struts と同じく JSF もモデルの実装には関与しない。タグライブラリから利用するために、JavaBeans にするのも同じである。このアプリケーションでは、先の Struts のところで紹介した JavaBeans とほぼ同じクラスをモデルとして用いている。

● 両者の比較

Struts と JSF はいずれもモデルの実装には関与しない。いずれも JavaBeans を用いている。

5．ビジネスロジック

● Struts

JSP のフォームから入力されたデータは、"Action Form Bean" という JavaBeans 形式のクラスファイルに格納される。

ビジネスロジックを実行するのは、「アクションクラス」という独自の形式を持つクラスである。このクラスは後に示す "struts.config.xml" ファイルで指定されたタイミングで呼び出され、execute メソッドが実行される。また、メソッドの返値によって次に遷移する画面が決定される。

遷移先のビューで使用されるオブジェクトは、HttpSession に格納される。

```
.....
public class SearchBookAction extends Action {

   public ActionForward execute(
      ActionMapping mapping,ActionForm f,
      HttpServletRequest request,
      HttpServletResponse response){

      SearchForm form =(SearchForm)f;
      String word =form.getWord();

      List list =search(word);

      HttpSession session =request.getSession();
      session.setAttribute("bookList",list);

      return(mapping.findForward("success"));
   }
```

● JSF

JSF では、フォームから入力されたデータとビジネスロジックは、"Backing Bean" (7) という JavaBeans で処理される。このサンプルプログラムでの Backing Bean の一部分を示す。

```
.....
public class BookSearcher {
   .....
   public void setWord(String word){
      this.word =word;
```

```
    }
    public String getWord(){
        return word;
    }
    public String searchBooks(){
        searchBooks(word);
        return "success";
    }
    .....
}
```

フォームから入力されたデータは、Backing Bean のプロパティに格納される。このクラス中の最後に示した searchBooks というメソッドがビジネスメソッドになる。このメソッドは、JSP から呼び出すことができる。

この Backing Bean は、後に示す "faces-config.xml" というファイルで指定する必要がある。そのことによって、セッションが開始されると自動的に Backing Bean が利用できるようになる。セッション管理に関するプログラムを記述する必要はない。

●両者の比較

Struts では Action Form Bean とアクションクラスの組み合わせる。一方、JSF では Backing Bean のみで処理をすることになる。

ビジネスメソッドを呼び出す設定は、Struts では struts-config.xml で指定するのに対し、JSF では JSP に記述することになる。

また、JSF ではセッション管理に関するプログラムを記述しなくても良いのは、大きな利点となる。

6．ビュー

● Struts

Struts では、JSP がビューとして標準的に用いられる。JSP では Struts 独自のタグライブラリが用いられる。このタグライブラリは、Struts の全体的な挙動との関係が深い。

図 1 に対応する JSP ファイルである search.jsp を以下に示す。

```
<%@page contentType="text/html;charset=Shift_JIS"%>
<%@taglib uri="WEB-INF/struts-html.tld" prefix="html"%>

<html:html>
<head>
<link href="style.css" type="text/css" rel="stylesheet"/>
```

```
<title> 図書検索 </title>
</head>
<body>

<h1> 図書検索 </h1>
<hr />
<html:form action="/searchBook">
<p>
<html:text property="word"/>
<html:submit property="submit" value="Go!"/>
</p>
</html:form>

</body>
</html:html>
```

● JSF

JSF でも、JSP がビューとして標準的に用いられる。Struts と同様に、JSF 独自のタグライブラリを用い、JSF の全体的な挙動との関係が深くなっている。

図 1 に対応する JSP ファイルである search.jsp を以下に示す。

```
<%@taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<%@taglib uri="http://java.sun.com/jsf/core" prefix="f"%>

<html>
<head>
<meta http-equiv="Content-Type"
      content="text/html;charset=Shift_JIS"/>
<link href="style.css" type="text/css" rel="stylesheet"/>
<title> 図書検索 </title>
</head>
<body>

<h1> 図書検索 </h1>
<hr />

<f:view>
```

```
    <h:form id="searchForm">
      <h:input_text id="searchWord" value="#{BookSearcher.word}"/>
        <h:command_button id="submit"
              action="#{BookSearcher.searchBooks}" value="Go!"/>
    </h:form>
  </f:view>

  </body>
  </html>
```

JSF のタグライブラリで特徴的なのは、JavaServer Faces expression language (JSF EL)が利用できることである。上の JSP の 2 箇所にある "#{...}" の部分が JSF EL である。前者の JSF EL では、"Backing Bean" の word プロパティを示す。また後者の JSF EL では、"Backing Bean" の searchBooks メソッドを呼び出すことを示す。

JSF EL の導入によって、"Backing Bean" のプロパティやメソッドに手軽にアクセスできるようになっている。なお、JSP 2.0 から導入された "Expression Language" は、現在の JSF の実装では利用できない。

なお図 2 では、Struts 版では箇条書きの ul タグを用いたが、JSF 版では後述する data_table タグによる、テーブルを使った画面表示をせざるを得なかった。Struts を使わずに JavaBeans のコレクションの内容を表示するには、JavaServer Pages Standard Tag Library (JSTL)(8)の forEach タグを使うのが基本的な手法である。そして JSF でリンクを貼るには、command_link タグと output_text タグを組み合わせて使う必要がある。しかし、forEach タグを使うと、各行に固有な id 属性を output_text タグに付加する必要があるとのエラーメッセージが出た。そこで JSF EL を用いて id 属性の値を生成してみたが、これも失敗した。id 属性の値には JSF EL を使えないようである。さらに、JSF のタグライブラリの中で Expression Language を使うこともできないようである。こうしたことによって、リスト表示に forEach タグを利用することは断念した。以下にエラーになった JSP を示す。

```
  <f:view>
  <ul>
    <c:forEach var="book" items="${BookSearcher.bookList}"
      <h:command_link id="detail"
                    action="#{BookSearcher.searchBookDetail}">
        <li>
          <h:output_text id="bookTitle" value="#{list.title}"/>
        </li>
      </h:command_link>
    </c:forEach>
```

```
</ul>
</f:view>
```

forEach の代わりに用いたタグについては、後述する。

●両者の比較

両者ともタグライブラリを使った JSP を用いることに違いはない。しかし、JSF では JSF EL を利用できることが大きな利点となる。なお、図 2 では、Struts 版では箇条書きを用いたのに対し、JSF 版では箇条書きを使えず、テーブルを用いたという違いが出た。

7．コントローラ

● Struts

Web ブラウザからのリクエストは、すべて「アクションサーブレット」という Servlet が処理をする。この Servlet は、Struts にはじめから用意されている。

● JSF

Struts と同様に、Web ブラウザからのリクエストは、"Faces Servlet" という Servlet が処理をする。この Servlet は、JSF にはじめから用意されている。

●両者の比較

コントローラとしての Servlet が一つだけ存在するという点では、両者とも変わりはない。

8．画面の遷移

● Struts

図 1 に対応する JSP ファイルである search.jsp では、"Go!" ボタンが押されると、「アクションサーブレット」を経由して、html:form タグの action 属性である "/searchBook" という URL に処理が渡される。

```
<html:form action="/searchBook">
<p>
<html:text property="word"/>
<html:submit property="submit" value="Go!"/>
</p>
</html:form>
```

図 2 では検索結果をリスト表示する。任意の行をクリックすると図 3 の詳細表示画面に遷移する。図 2 の表示を行う list.jsp では、一行分のデータを次のように処理している。

```
<html:link forward="search/detail" name="book" property="mapping">
   <bean:write name="book" property="title"/> /
   <bean:write name="book" property="author"/>
</html:link>
```

任意の一行のデータをクリックすると、html:link タグの forward 属性の値である "search/detail" に処理が渡されることになる。このとき、クリックされた行を識別するため、JavaBeans の id プロパティ名とその値を、JavaBeans の mapping プロパティに格納している。

```
public class BookSearcher {
   .....

   public void setMapping(){
      mapping.put("id",id);
   }
}
```

画面の遷移に関する情報は、struts-config.xml というファイルに記述する。このファイルのうち、図 1 から図 2 への遷移に関わる部分を示す。

```
<action-mappings>
   <action       path="/searchBook"    type="SearchBookAction"
                 name="searchForm"     scope="request">
      <forward name="success"          path="/list.jsp"/>
   </action>
   .....
</action-mappings>
```

まず、図 1 の "Go!" ボタンが押されると、先の JSP の記述に従い、"/searchBook" という URL に処理が渡される。この URL 名は、action タグの path 属性の値に一致している。そこで、type 属性で指定されている "SearchBookAction" に処理が移る。これは、「ビジネスロジック」の節で示したアクションクラスである。このアクションクラスの execute メソッドが実行される。このメソッドの返値となるオブジェクトには "success" という文字列が与えられており、この文字列は action 要素の子要素である forward タグの name 属性と一致している。これにより、path 属性で指定されている list.jsp（図 2）が表示されることになる。

続いて、図 2 から図 3 への遷移に関わる部分である。

```
<global-forwards>
  <forward name="search/detail"    path="/searchBookDetail.do"/>
</global-forwards>

<action-mappings>
  .....
  <action       path="/searchBookDetail"    type="SearchBookDetailAction"
                name="searchForm"    scope="request">
    <forward name="success"          path="/detail.jsp"/>
  </action>
</action-mappings>
```

この遷移は、図 1 から図 2 への遷移よりも複雑である。図 2 で、ある 1 行のデータへのリンクがクリックされると、JSP の記述に従い、struts-config.xml 中の "search/detail" という名前を探す。該当するものは、global-forwards 要素の子要素である forward タグの name 属性である。同じタグの path 属性である "searchBookDetail.do" から ".do" という文字列を削ったものは、action タグの path 属性と一致するから、後は先ほどと同様の手順によって図 3 への遷移が行われる。

● JSF

図 1 に対応する JSP ファイルである search.jsp では、"Go!" ボタンが押されると、"Faces Servlet" を経由し、JSF EL の記述に従って Backing Bean の searchBook メソッドが実行される。

```
<f:view>
  <h:form id="searchForm">
    <h:input_text id="searchWord" value="#{BookSearcher.word}"/>
      <h:command_button id="submit"
              action="#{BookSearcher.searchBooks}" value="Go!"/>
  </h:form>
</f:view>
```

次に、図 2 の表示を行う JSP ファイルを示す。

```
<f:view>
<h:form id="listForm">
<h:data_table id="table" value="#{BookSearcher.bookList}" var="list">
    <h:column>
      <f:facet name="header">
```

```
                <h:output_text value="  タイトル "/>
            </f:facet>
              <h:command_link id="detail" action="#{BookSearcher.searchBookDetail}">
                <h:output_text id="bookTitle" value="#{list.title}"/>
            </h:command_link>
        </h:column>
        <h:column>
            <f:facet name="header">
                <h:output_text value="  著者 "/>
            </f:facet>
            <h:command_link id="detail" action="#{BookSearcher.searchBookDetail}">
                <h:output_text id="bookAuthor" value="#{list.author}"/>
            </h:command_link>
        </h:column>
          </h:column>
    </h:data_table>
    </h:form>
    </f:view>
```

JSF ではイテレータを使って複数のデータを表示させるときには、data_table タグの利用が推奨されている。このタグによって、検索結果のデータは表として出力される。

任意の 1 行をクリックすると、Backing Bean の searchBookDetail メソッドが実行される。Struts の場合、どの行がクリックされたか識別するために JavaBeans の id プロパティ名とその値を格納したが、JSF の場合にはその必要はない。data_table タグを使うことで、JSF がクリックされた行の情報を持っているからである。ここでポイントとなるのは、data_table テーブルの value 属性である。この属性の値には、JSF EL で検索結果のコレクションを示している。この値には、var 属性で "list" という名前が付けられている。Backing Bean の次のメソッドでは、"list" という名前をキーにして、クリックされた行に対応する JavaBeans を取得している。

```
    public String searchBookDetail(){
        FacesContext context =
            FacesContext.getCurrentInstance();
        BookData book =(BookData)context.getExternalContext().
                            getRequestMap().get("list");
        String id =book.getId();
        searchBookById(id);
        return "success";
```

```
}
```

画面の遷移方法は、"faces-config.xml" というファイルに記述する。画面の遷移に関係するのは次の部分である。

```
<navigation-rule>
  <from-view-id>/search.jsp</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/list.jsp</to-view-id>
  </navigation-case>
</navigation-rule>

<navigation-rule>
  <from-view-id>/list.jsp</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/detail.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

navigation-rule という要素が 2 つある。

1 つめでは、search.jsp から呼ばれたビジネスメソッドが "success" という文字列を返したら、list.jsp に遷移することを示す。また 2 つめでは、list.jsp から呼ばれたビジネスメソッドが "success" という文字列を返したら、detail.jsp に遷移することを示す。次に示すプログラムは、search.jsp で "Go!" ボタンが押されたときに呼ばれる searchBooks メソッドである。このメソッドは "success" という文字列を返すので、このメソッドが実行されると次に list.jsp に遷移する。JSP から JSF EL を使って呼ばれるビジネスメソッドは、この "success" のような文字列を返値とする必要がある。

```
public String searchBooks(){
  searchBooks(word);
  return "success";
}
```

●両者の比較

画面遷移の方法を比較すると、JSF が単純な仕組みになっていることがわかる。faces-config.xml の設定では、JSP から JSP への遷移が明確になっており、わかりやすい。また、Struts にはボタンをク

リックしたときとリンクをクリックしたときに遷移方法の記述の違いがあるが、JSF にはそれがない。

ただし、図 2 で任意の行がクリックされたとき、どの行がクリックされたのか識別する方法は、両者とも若干複雑である。

9．まとめ

以上のように、同一の機能を持つサンプルアプリケーションを通して、Struts と JSF を比較してみた。

サンプルアプリケーションそのものを比較すると、JSF を用いて Struts 版とほぼ同等の機能を持つアプリケーションを作成できた。ただし、図 2 では、Struts 版では箇条書きの ul タグを用いたが、JSF 版ではテーブルにせざるを得なかった。

次に、プログラムの実装について比較してみる。Struts と比較した JSF の利点は次のように整理できる。第一に、画面の遷移を簡明に書ける点である。第二に、JSF EL が導入された JSP を利用できる点である。第三に、セッション管理に関するプログラムを記述しなくてもよい点である。

こうした利点によって、プログラムおよび設定ファイルを簡明にできる。よって、プログラムの作成や保守などの開発効率の向上が見込まれる。

一方、JSF を使ったプログラム開発において、いくつか問題点が発生した。まず第一に、JSP の節で述べたように、JSTL と JSF のタグライブラリとをうまく組み合わせられない点である。JSF のタグライブラリでは、制御構造などのロジックを扱う機能がないので、これは修正する必要があるだろう。第二に、日本語の取り扱いである。まず、JSP のファイルの先頭にディレクティヴを設定するとまったく日本語が表示されない。ディレクティブを削ってから JSP を表示させたとき、JSP に直接書かれている日本語がすべて文字化けを起こす。これは charset の取り扱いに問題があり、文字コードをすべて ISO-8859-1 と認識してしまう問題だと思われる。

以上、Struts の代替としての JSF について検証を進めてきた。結論としては、日本語環境が修正されれば、開発効率の向上を望める点から、十分に代替となりうる技術だと言える。

●参考文献

（1）Tha Apache Struts Web Application Framework.
http://jakarta.apache.org/struts/index.html

（2）JavaServer Faces.http://java.sun.com/j2ee/javaserverfaces/

（3）JSR 127 JavaServer Faces.http://www.jcp.org/en/jsr/detail?id=127

（4）Craig McClanahan and Ed Burns.
JavaServer Faces Specification -Version Proposed Final Draft (2003.12)

（5）Sun Java Studio Creator.http://wwws.sun.com/software/products/jscreator/index.html

（6）Graham Hamilton:JavaBeans Specification 1.01 Final Release.
http://java.sun.com/products/javabeans/docs/spec.html (1997.8)

（7）Eric Armstrong et al.:Java Web Services Tutorial.
http://java.sun.com/webservices/docs/1.3/tutorial/doc/index.html (2003.12)

（8）JavaServer Pages Standard Tag Library.http://java.sun.com/products/jsp/jstl/

●英文タイトル

Verification of JavaServer Faces as an alternative to Struts

●英文要約

In this paper, I try to verify whether JavaServer Faces (JSF) can serve as an alternative to Struts for developing Web applications using MVC model 2 written in Java. For this purpose, I compare two implementations: (1) the implementation of a simple application developed using Struts, and (2) that of another application (with the same functions) developed using JSF. These two applications have much the same functions, except for some slight difference in view due to the JSF tag library's restriction. The comparison shows that the JSF application is superior to the Struts application in the three respects that follow. First, I can write navigation between pages plainly by using JSF. Second, I can use JSP with JSF EL. Third, I don't need to write a program of session management. These strong points give good reason to expect that by using JSF, the efficiency of developing Web applications will be improved. Although JSF causes "moji-bake" (misdisplays of characters), I arrive at the conclusion that JSF *can* serve as an alternative to Struts, except for this "moji-bake" problem that I hope will be solved in the near future.