用 Webflow 构建 Web 程序

Brian Pontarelli and Grant Zemont
翻译 Rainman
原文
http://www.javaworld.com/javaworld/jw-02-2003/jw-0214-webflow.html?

Webflow 是 BEA WebLogic Portal 框架的核心部分。BEA WebLogic Portal 包含很多组件，包括用户管理，商业，权限和活动流程。用 BEA WebLogic Portal 构建的 Web 程序利用 Webflow 系统，从而有助于分离表示逻辑和商业流程。

Model-View-Controller 2 (MVC2)是 Webflow 体系架构的基础。Webflow 大部分都是控制器功能，但也提供基本的 model 支持和用于 view 的标记库。不同于其它大多数的 Controller 系统，Webflow 更像是一个工作流系统。它可以把不同的商业逻辑类链接在一起，同时支持分支。不同于 Struts 等系统，Webflow 以事件和处理器为基础，且通过基于事件的链和分支提供更多的功能。

Web 程序可利用 Webflow 的功能去设置事件，逻辑和网站流程。Webflow 的“流”以表示节点为起点（JSP），通过零或多个类（名为 InputProcessor 的事件处理器），直至遇到另一个表示节点。这样的设计令到 Webflow 成为构建 Web 程序的有力工具。

Webflow 的关键特性包括：

1. 基于组件的结构利于代码重用；前端，中层，后端组件全部分工清楚。
2. 遵从 J2EE 规范。
3. 内置表格验证。
4. 前端和后端开发者分工明确。

以下我们研究 Webflow 的每种组件和讨论一些例子。

Webflow components

一个 Webflow 全部由节点（nodes）组成。在 Web 程序中，一个节点是一个抽象的组件，它表现为一个对象。节点是由事件互相连接的。一个简单的 Webflow 可表示如下：

源 JSP -> InputProcessor -> 目标 JSP

在图中的每个箭头都是被 Webflow 系统处理的事件。因此，一个 Webflow 是一串被连接的节点，它描述了 Web 程序的一部分逻辑流程。本节简要的介绍节点的类型和组成 Webflow 框架的概念。在这里，我们不会重复已存在的 BEA 文档。关于此框架的更进一步的信息请浏览 BEA 网站。

Presentation nodes（表示节点）

表示节点是用户与 Web 程序交流的组件，例如 JSP,HTML,WML 和 Servlet。

InputProcessor nodes（输入处理器）

这些节点是实现 InputProcessor 接口的 java 类。因此，InputProcessor 必须实现以下方法：

public Object process(HttpSerlvetRequest request, Object requestContext) throws ProcessingException

Copyright © Java开发者(www.ChinaJavaWorld.com), All Rights Reserved.

HttpServletRequest 对象是从客户端发送给 Weblogic 应用服务器的请求。这个请求来自 InputProcessor 前的表示节点。在上述的简单例子中，在源 JSP 中的 form 或 anchor 标记会创建请求传递给 InputProcessor。RequestContext 是请求的唯一标识符。

Pipeline 和 PipelineComponent 节点（管道和管道组件节点）

PipelineComponent 可以事务性或非事务性执行有关商业过程的特殊任务。多个 PipelineComponent 可连接到单一的 Pipeline。Pipeline 的执行是线性的，因为它不支持分支。事务性 Pipeline 确保所有成功执行或事务回滚。包含商业逻辑的 PipelineComponent 可应用到大型商业程序中的多个 Web 程序，因此可装载到 BEA Weblogic 的 EJB 容器。PipelineComponent 实现 PipelineComponent 接口。所以，PipelineComponent 必须实现以下方法：

public Object process(PipelineSession pipelineSession, Object requestContext) throws PipelineException, RemoteException

PipelineSession 允许在 Web 程序中的 InputProcessor 和在企业程序中的 PipelineComponent 互相通信。RequestContext 依然是传递给 InputProcessor 的唯一标识符。

名字空间

名字空间允许开发者把 Web 程序分割成多个 Webflow。每个 Webflow 有它自己的名字空间，而且是不可嵌套。通常，名字空间是伴随逻辑线路创建的。例如，在典型的商业网站，一个开发者可为在线目录，记录管理，核对和订单查找等模块创建互相独立的名字空间。

事件

事件是从一个节点到另一节点以及从 Webflow 到 Pipeline 的传换。当一个节点完成任务，它返回一个事件，系统以此决定调用 Webflow 中下一个节点。表示节点的 form 发送，超链接或其它向服务器的请求都会产生事件。InputProcessor 节点当 process 方法返回时产生事件。它们可通过抛出异常产生异常事件。InputProcessor 中的所有事件类型告诉 Webflow 系统下一个节点是哪个。

E－Business 控制中心

E－Business Control Center(EBCC)是图形界面应用程序，用以创建 Webflow 和管理 BEA WebLogic Portal 应用程序。在 EBCC，开发者能够把 JSP 页面，PipelineComponent 和 InputProcessor 以节点的形式关联起来。它们也可以执行其它与 BEA WebLogic Portal 相关联的任务，例如用户管理和活动管理。本文只关注 EBCC 有关 Webflow 创建和管理的方面。

在 EBCC 中，一些 XML 文件存储了有关 Webflw 的信息。通常开发者不需要直接编辑这些文件。以下是一些重要的 Webflow 文件：

- 工程文件，<project name>.eaprj，是最主要的 Webflow 程序工程文件。
- .wf 文件，<namespace>.wf，包含 Webflow 的名字空间信息，包括事务和事件信息。

Copyright © Java开发者(www.ChinaJavaWorld.com), All Rights Reserved.

- .wf.ui 和.ui 文件用来显示 Webflow 组件在 EBCC 中的最后组织。
- .wfx 文件允许 Webflow 扩展，它超越了本文的范围。

Webflow 程序例子

在本节，我们用 Webflow 创建一个模拟运输过程跟踪程序。此应用程序的概念是用户可通过 email 进行信息跟踪。然而实际上，此程序除去了找回跟踪信息和通过 email 发送跟踪信息的代码，以保持应用程序简单。

注意：Pipeline 是线性的，因此分支应用在 Pipeline 中是困难的，循环就更困难了；所以，Pipeline 应放置在流程是线性且没有分支和循环发生的地方。除非你需要事务，否则 Pipeline 的应用一般都有很多限制。

Webflow in EBCC

建立 Webflow 程序的第一步是抽象地把程序分解成块。在这里可应用一些好的实践。例如，把 form 的验证与其它逻辑分离就是好的想法。在 Webflow 的世界里，这意味着把验证的逻辑放在它自己的 InputProcessor 中。

另一实践就是决定是否用 Pipeline 和 PipelineComponent。就像上述的那样，当商业逻辑不存在分支和循环时，Pipeline 可以理想地把商业逻辑聚合在一起。如果流程执行到一个点，而它后面跟着许多不同分支流程时，你就应该用 InputProcessor。同样地，要根据条件重复地执行同一流程时，也应该用 InputProcessor。然而，若以上两种情况你都没有遇到，Pipeline 就可能会适合你。

最后，要考虑流程是否需要事务。例如，如果一个流程包含两个步骤，要不这两个步骤都执行成功且提交，要不失败回滚，你就会想到把 PipelineComponent 作为 EJB 组件，这样就可以支持事务。

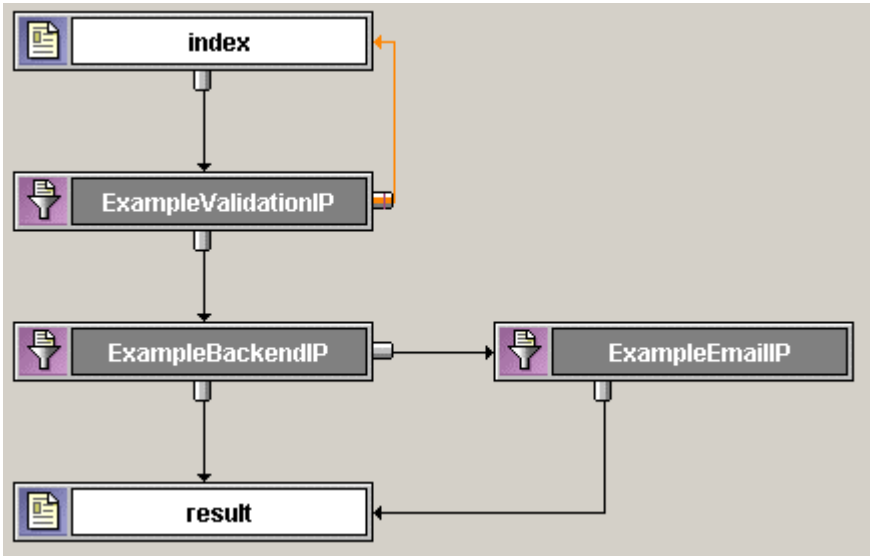
在我们的程序中，不管用户是否选择通过 email 接受跟踪信息，我们都会有分支的情况。因此，我们把所有逻辑都放在 InputProcessor 中。程序中会有三个逻辑步骤，相对地，会有三个 InputProcessor。这三个逻辑步骤为：

1. 从用户得到信息并验证它。
2. 从后台系统得到跟踪信息。
3. 选择性地把跟踪信息通过 email 发送给用户。

一个隐藏的第四步是向用户显示跟踪信息。这一步不在我们的商业逻辑中，而在表示逻辑中，所以属于 JSP 范围。

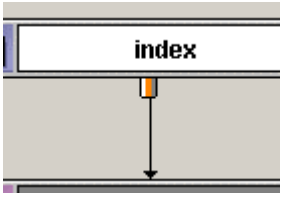
图一描述的是整个 Webflow，当中包括 InputProcessor 和 JSP。

Copyright © Java开发者(www.ChinaJavaWorld.com), All Rights Reserved.



图表 1 完整的 Webflow

你可以看到开始节点是 Index 节点。此节点包含单一名为 submit 的事件，它指向后续的 ExampleValidationIP。



图表 2 Webflow 事件

此节点有两个事件。第一个在图三显示的异常事件，名为 com.bea.p13n.appflow.Webflow.forms.InvalidFormDataException 。如果 ExampleValidationIP 抛出此异常，Webflow 就会跟随此事件返回到 Index。

图表 3 Webflow 异常

图 2 和图 3 的细节分别显示在图 4 和图 5。

图表 4 Webflow 事件细节

Copyright © Java开发者(www.ChinaJavaWorld.com), All Rights Reserved.

图表 5 Webflow 异常细节

JSP 页面

现在我们编写验证和抛出异常的代码。在解决这模块之前，我们要构造自己的 JSP 或者至少在脑中构思要包含哪些 JSP。我们用单一表格给用户输入信息，例如跟踪号码。用户有权选择 email 通知功能和填写 email 地址。

作为 Webflow 的一部分，BEA 提供标记库，其中的标记使开发者更容易进行表格值验证。这些标记及其 API 被称作 validated form。validated form 是 Webflow 的内建属性。粗略来说，validated form 使开发者通过表格中域的名字（HttpServletRequest 的参数名）来验证表格和封装成简单 JavaBean 属性。此 API 不支持复杂的 JavaBean 操作。

首先我们要在 JSP 中 include 标记库：

```
<%@ taglib uri="WebFlow.tld" prefix="wf" %>
```

在我们的 JSP validated form 如下：

```
<wf:validatedForm event="submit" namespace="Example"
applyStyle="field" invalidStyle="color: red;">
  Tracking number <wf:text name="trackingNumber"/><br>
  Email? <wf:checkbox name="email" value="true"/><br>
  Email address <wf:text name="emailAddress"/><br>
  <input type="submit" value="Submit" name="submit"/>
</wf:validatedForm>
```

表格验证的代码

现在我们已设置了 JSP，接下来是表格验证的代码。我们使用一些 ValidatedForm API 中的类：ValidatedFormFactory 创建 ValidatedForm 类，ValidatedValuesFactory 创建 ValidatedValues 类。ValidatedForm 和 ValidatedValues 追踪用户在每个表格域中提供的值以及确定特定的值是否有效。所以当遇到验证错误时，Webflow 就会重指向到输入页面，ValidatedForm API 会重新显示先前用户输入的值和显示错误信息。

验证表格的代码如下：

```
// Create the request JavaBean and store it in the request
TrackingRequest trackingRequest = new TrackingRequest();
request.setAttribute("trackingRequest", trackingRequest);

ValidatedForm vForm = ValidatedFormFactory.getValidatedForm();
ValidatedValues vValues =
ValidatedValuesFactory.getValidatedValues(request);

// Set the JavaBean values, except the Boolean for the check box, which
// we will have to set automatically, because the ValidatedForm API from
// BEA doesn't handle it
```

Copyright © Java开发者(www.ChinaJavaWorld.com), All Rights Reserved.

```
vForm.validate(vValues, trackingRequest);
...
// Validate the tracking number
 vForm.validate(vValues, STRING_VALIDATOR, "trackingNumber", new MinMaxExpression(10, 15),"The tracking number must be between 10 and 15 characters");

// If the user selected email, validate the email address
if (trackingRequest.isEmail()) {
 vForm.validate(vValues, STRING_VALIDATOR, "emailAddress",
    new MinMaxExpression(10, 100),
    "The email address must be between 10 and 100 characters");
}

if (vValues.getInvalidFieldCount() > 0) {
    throw new InvalidFormDataException("Please correct the invalid   fields");
}
```

这段代码使用 validate()方法。此方法获得需要验证的请求参数和要用到的验证器名字（这里我们使用 BEA 提供的 STRING_VALIDATOR），以及访问验证器的表达式。

另一个重要的地方是 validated form 封装 JavaBean。在代码中的 validate()方法同时获得 ValidatedValues 对象和 JavaBean 对象，再根据对应的值封装 JavaBean 属性。

若验证过程中有错误产生，抛出 InvalidFormDataException，同时返回 Index 页面。如果用户输入了不恰当的跟踪号码，他会得到错误信息“The tracking number must be between 10 and 15 characters”。

如果验证成功，ExampleValidationIP 返回字符串“success”，Webflow 指向流程下一个 ExampleBackendIP（参考图 1，2，4）。

InputProcessor 分支

ExampleBackendIP InputProcessor 的代码故意省略掉。相对于研究复杂的商业逻辑，我们更重要的是看 Webflow 如何以流程系统运作。从图 1 可看到，这个 InputProcessor 引出两个事件分支：一个是指向表示节点，另一个是指向 ExampleEmailIP。下面的代码展示如何选择从 ExampleBackendIP 返回的事件：

```
// Retrieve the TrackingRequest out of the request
  TrackingRequest trackingRequest = (TrackingRequest) request.getAttribute("trackingRequest");
...
// Check what to forward to
return trackingRequest.isEmail() ? "email" : "complete";
```

此 InputProcessor 返回适当的值给 Webflow 系统，系统决定从哪条路径继续执行。在我们的例子中，如果用户选择接受 email，字符串“email”返回。否则，返回字符串“complete”。尽管图 1 没有显示哪一个事件指向流程的哪一部分，但容易看出从 ExampleBackendIP 指向 ExampleEmailIP 的事件是”email”，从 ExampleBackendIP 指向 result.jsp 的事件是“complete”。

Copyright © Java开发者(www.ChinaJavaWorld.com), All Rights Reserved.

一般而言，在 Webflow 系统中的事件只是简单的字符串。如果你重看上面，从 InputProcessor 的 process 方法返回的是 Object。Webflow 系统会执行此 Object 的 toString()方法来决定后续的执行。

在图 1，ExampleEmailIP 有一个连接到 result.jsp。此连接依然是“complete“，但可以随我们的意愿来命名。这是 ExampleEmailIP process()返回的字符串。这个流程不只是适应 Webflow。系统有足够的扩展性可允许分支到其它 InputProcessor 和表示节点。你可以分支和连接到 Webflow 的其它类型节点，包括客户端节点。

我们省略了 ExampleEmailIP 和商业逻辑的代码，使你更容易学习 Webflow 技术。你可以从 http://www.javaworld.com/javaworld/jw-02-2003/webflow/jw-0214-webflow.zip 下载源代码。

结论

在我们的例子中，最后的 JSP 页面包含简单的逻辑来显示跟踪信息,此信息从 ExampleBackendIP 得到并存储在 HttpServletRequest 中：

```
<jsp:useBean id="trackingData" scope="request" type="com.inversoft.javaworld.TrackingData"/>
<jsp:useBean id="trackingRequest" scope="request" type="com.inversoft.javaworld.TrackingRequest"/>
<p>
The value of your shipment:
<jsp:getProperty name="trackingData" property="value"/>
</p>
<p>
The description of your shipment:
<jsp:getProperty name="trackingData" property="description"/>
</p>

<%
if (trackingRequest.isEmail()) {
%>
Your shipment information has been emailed to:
<jsp:getProperty name="trackingRequest" property="emailAddress"/>
<%
}
%>
```

结果是从 HttpServletRequest 得到的，因为 ExampleBackendIP 或 ExampleEmailIP 都会把 request 传到这一节点。这个通用机制使数据从 InputProcessor 传递到 JSP 页面。开发者也可以把数据存到用户 session 或是应用程序 context，但在决定之前，一定要衡量这样做的价值和各种可能的结果。

Copyright © Java开发者(www.ChinaJavaWorld.com), All Rights Reserved.