第5章　页面导航

本章主题：

- 导航处理及相关组件
- 隐式导航
- 导航规则及基于导航规则的导航
- 重定向
- h:link 与 h:button 标记
- 视图参数与可书签化 URL

对于一个 Web 应用来说，页面导航的设计好坏直接影响用户对应用的访问效率。从 JSF 1.2 到 JSF 2.0，JSF 提供的页面导航技术有了很多的扩充。从原先根据用户动作和业务逻辑确定目标视图的 POST 请求，到新的面向视图参数和目标视图的 GET 请求，本章将对它们做详细介绍。

5.1　导航概述

在 Web 应用中，所谓导航是指通过在页面上放置超链接或递交按钮，使访问者可以快速地从一个页面（视图）转至另一个页面（视图）。合适的导航能够方便访问者访问其所需的信息或请求相关的业务处理。

导航处理是指根据当前请求的上下文状态信息确定目标页面（视图）的过程。在 JSF 中，导航处理由导航处理器（NavigationHandler）完成，下面组件涉及导航处理：

- h:commandButton（动作按钮）；
- h:commandLink（动作超链接）；
- h:button（结果按钮）；
- h:link（结果超链接）。

对于动作类组件（h:commandButton 和 h:commandLink），导航处理发生在用户发出请求之后。当用户单击组件呈现的按钮或超链接提交请求时，将触发动作事件、调用动作方法，之后就由导航处理器确定要呈现的目标视图。这种导航处理也称为后置式导航处理。

对于结果类组件（h:button 和 h:link），导航处理发生在页面呈现前，即在组件呈现时，导航处理器就会确定目标视图。当用户单击组件呈现的按钮或超链接时，将直接转至之前已确定的目标视图。这种导航处理也称为前置式导航处理。

说明：h:outputLink 组件呈现的普通超链接是一种基本的导航技术，但其目标页面是在标记中直接指定的，所以并不需要 JSF 导航处理器进行导航处理。

无论是后置式导航处理还是前置式导航处理，其处理算法是一致的。导航处理器根据结果值、并结合其他上下文状态信息进行导航处理。下面是导航处理中会涉及的几个主要术语。

1. 动作值(action value)

动作组件的 action 属性值本身,可以是一个 EL 方法表达式,也可以是一个简单的字符串文字。

2. 结果值(outcome value)

是导航处理器进行导航处理的主要依据。结果值的可能情况包括:

- 动作组件的 action 属性指定的字符串文字。
- 动作组件的 action 属性指定的 EL 方法表达式的计算结果。
- 结果组件的 outcome 属性指定的字符串文字。
- 结果组件的 outcome 属性指定的值表达式的计算结果。

3. 视图 ID(view ID)

以"/"开头、相对于应用上下文路径的视图(页面)URL。

说明:

(1) 这里,action 是 h:commandButton 组件标记和 h:commandLink 组件标记的一个属性,而 outcome 是 h:button 组件标记和 h:link 组件标记的一个属性。

(2) 对于后置式导航处理,若 action 属性指定的是简单的字符串文字,则其动作值与结果值是相同的。

(3) 对于前置式导航处理,不牵涉动作值。

5.2 隐式导航

隐式导航是 JSF 2.0 新引入的特性,是指在开发人员没有指定相应的导航规则时,导航处理器所进行的导航处理。

当未定义导航规则或没有匹配的导航规则时,JSF 框架的导航处理器将把结果值看作是目标视图的视图 ID。如果结果值不以"/"开头,则在其前添加源视图相对于应用上下文路径的路径。如果结果值不包含文件扩展名,则在其后添加源视图的文件扩展名。

例如,源视图(/login.xhtml)的表单中包含如下的命令按钮标记:

```
<h:commandButton id="cm" value="OK" action="response"/>
```

当用户单击该按钮时,将会导航至视图 ID 为/response.xhtml 的目标视图。这里,虽然导航处理器参与了导航处理,但导航处理确定的目标页面视图总是某个固定的页面视图。这种导航称为静态导航。

又例如,源视图(/login.xhtml)的表单中包含如下的命令按钮标记:

```
<h:commandButton id="cm" value="OK" action="#{login.isUser}"/>
```

login 引用一个受管 bean,bean 类必须包含一个 isUser 方法。

```
public String isUser(){
    if(...){                          //验证成功
        return "success";
    } else {                          //验证失败
        return "failure";
```

```
    }
}
```

若验证成功，将会导航至视图 ID 为/success. xhtml 的目标视图。若验证失败，将会导航至视图 ID 为/failure. xhtml 的目标视图。这里，导航处理确定的目标页面视图不仅取决于用户单击了哪个按钮或超链接，也取决于当前请求的其他上下文状态信息，从而会导航至不同的目标页面视图。这种导航称为动态导航。

隐式导航的特点是不需要指定导航规则，目标视图 ID 由结果值直接指定。这种导航方式的优点是简单、直接，易于理解；缺点是不能对整个应用的导航模型进行集中管理，不便于维护。

隐式导航不仅可以实现静态导航，也可以实现动态导航；不仅适用于后置式导航，也适用于前置式导航。

5.3 基于导航规则的导航

当导航处理器进行导航处理时，首先寻找匹配的导航规则。如果存在匹配的导航规则，将依据导航规则确定目标页面视图。

5.3.1 导航规则

导航规则由 navigation-rule 元素及其子元素在/WEB-INF/faces-config. xml 或其他自定义的 Faces 配置文件中声明。下面是一个导航规则的声明示例。

```
<navigation-rule>
    <from-view-id>/index.jsp</from-view-id>
    <navigation-case>
        <from-action>#{login.isUser}</from-action>
        <from-outcome>ok</from-outcome>
        <if>#{login.valid}</if>
        <to-view-id>/success.xhtml</to-view-id>
    </navigation-case>
    …
</navigation-rule>
```

其中：

(1) 每一个 Faces 配置文件可以包含多个 navigation-rule 元素，每个 navigation-rule 元素声明一个导航规则。

(2) 每一个 navigation-rule 元素可以包含一个可选的 from-view-id 子元素，另外可以包含多个 navigation-case 子元素，每个 navigation-case 元素声明一个导航案例。

(3) 每一个 navigation-case 元素可以包含一个可选的 from-action 子元素、一个可选的 from-outcome 子元素、一个可选的 if 子元素，另外需要包含一个必选的 to-view-id 子元素。

下面是上述一些元素的含义及作用。

from-view-id：用于指定适用该导航规则的源页面视图的视图 ID。如果未指明该元素，

一个导航规则将适用于所有页面视图。一个页面视图的所有导航案例可以定义在一个导航规则里，也可以分散定义在多个导航规则里。

from-outcome：用于指定适用该导航案例的结果值。

from-action：用于指定动作值(即动作方法表达式本身)。源于同一个页面视图的多个请求，在调用各自的动作方法后可能会产生相同的结果值，此时可以通过动作值区分它们，以便匹配不同的导航案例。

if：该元素值应该是一个布尔型的值表达式，用于指定该导航案例是否匹配的动态条件。

to-view-id：用于指定目标页面视图的视图 ID。

5.3.2 导航算法

导航处理器以结果值为参数、导航规则和导航案例为依据，结合当前请求的其他上下文状态信息进行导航处理。下面是导航处理算法一个大致描述。

(1) 导航处理器首先会依次(在配置文件中从上到下)检查各导航规则，寻找相匹配的导航规则。相匹配的导航规则通常是指其 from-view-id 元素内容为源视图 ID 的导航规则。

(2) 当找到一个相匹配的导航规则时，导航处理器将依次检查其中的各导航案例，寻找相匹配的导航案例。相匹配的导航案例是指满足下面条件的导航案例：

- 如果包含 from-action 元素，其内容应等于动作值；
- 如果包含 from-outcome 元素，其内容应等于结果值；
- 如果包含 if 元素，其指定的值表达式的计算结果应为 true。

(3) 若发现一个相匹配的导航案例，其中不包含 if 元素，则：

- 若结果值为非 null，则其 to-view-id 元素的值即为目标视图 ID；
- 若结果值为 null，重新显示源视图。

(4) 若发现一个相匹配的导航案例，且其中包含 if 元素，则不管结果值是否为 null，其 to-view-id 元素的值即为目标视图 ID。

(5) 若未发现任何匹配的导航规则及导航案例，而结果值为 null，则重新显示源视图。

(6) 若未发现任何匹配的导航规则及导航案例，而结果值为非 null，则进行隐式导航。

说明：只有当结果值为 null 且不包含 if 元素时，当前视图作用域才会延续；否则在导航处理后，一个新的视图作用域将产生。

也可以在 WEB-INF 目录下创建其他的 Faces 配置文件(非 faces-config. xml)，然后在其中声明导航规则。这些配置文件的相对于应用上下文路径的路径列表(各路径间以逗号分隔)应作为名为 javax. faces. CONFIG_FILES 的上下文参数的初始值在 web. xml 文件中指定。下面是设置该上下文参数的一个示例。

```
<context-param>
    <param-name>javax.faces.CONFIG_FILES</param-name>
    <param-value>/WEB-INF/faces-config_1.xml,/WEB-INF/faces-config_2.xml
    </param-value>
</context-param>
```

当存在多个 Faces 配置文件时，导航处理器按以下顺序检查各 Faces 配置文件中的导航规则和导航案例：

- 上下文参数 javax. faces. CONFIG_FILES 的值中指定的顺序。如对于上面示例，导航处理器将先检查 faces-config_1. xml 配置文件中的导航规则和导航案例，然后再检查 faces-config_2. xml 配置文件中的导航规则和导航案例。
- /WEB-INF/faces-config. xml 文件。该配置文件中的导航规则和导航案例会被最后检查，即使该配置文件出现在 javax. faces. CONFIG_FILES 上下文参数值中也是如此。

5.3.3 导航规则的进一步说明

这里对导航规则的一些特殊情况作进一步的说明。

1. 通配符

一个导航规则的 from-view-id 元素的内容可以以通配符 * 结尾。此时，判断该导航规则是否匹配的条件是指该元素内容的前缀（* 之前的内容）与源视图 ID 的前部是否相同。若存在多个这样的匹配导航规则，选择匹配前缀最长的一个。

假设源页面视图的视图 ID 是/subsys/group/page. xhtml，Faces 配置文件中含以下两个导航规则：

```
<navigation-rule>
    <from-view-id>/subsys/*</from-view-id>
    <navigation-case>
        ......
    </navigation-case>
</navigation-rule>
<navigation-rule>
    <from-view-id>/subsys/group/*</from-view-id>
    <navigation-case>
        ......
    </navigation-case>
</navigation-rule>
```

那么这两个导航规则都适用于上述源页面视图引发的导航，但后面一个导航规则更恰当，因为其匹配的前缀更长。

若一个导航规则的 from-view-id 元素的内容仅为一个 *，或者根本不包含该元素，则该导航规则适用于任何源视图引起的导航处理。

在这里，一个导航规则可能适用于多个页面视图，反过来，一个源页面视图也可能存在多个相匹配的导航规则。对于一个具体的源页面视图，到底采用哪个导航规则，一要看哪个导航规则更恰当（视图 ID 匹配前缀较长），二还要看该导航规则中是否有相匹配的导航案例。也就是说，如果最恰当的导航规则中没有相匹配的导航案例，那么将检查另一个相匹配的导航规则中是否有相匹配的导航案例。

2. 动态目标视图 ID

每个 navigation-case 元素必须包含一个 to-view-id 子元素。该子元素不仅可以直接指定目标视图 ID，也可以指定一个 EL 值表达式。后种情况下，具体的目标视图 ID 通过计算 EL 值表达式获得。例如：

```
<navigation-rule>
    <from-view-id>/main.xhtml</from-view-id>
    <navigation-case>
        <to-view-id>#{myBean.nextViewID}</to-view-id>
    </navigation-case>
</navigation-rule>
```

该导航规则适用于源自视图/main.xhtml的导航。当导航处理时,名为myBean的受管Bean的getNextViewID()方法被调用,以获取目标视图ID。

5.4 重 定 向

当用户单击动作类按钮(h:commandButton)或超链接(h:commandLink)时会产生回送(postback)请求、引发相应组件的动作事件。在处理回送请求的"调用应用"阶段,动作方法被调用、并产生结果值,然后导航处理器以结果值为参数进行导航处理、确定目标页面视图,最后再由JSF框架将目标视图呈现给用户。这一过程的直接结果是:浏览器窗口中显示的是目标视图的内容,而浏览器的地址栏中仍显示源视图的URL。在很多情况下,这种不一致是不适当的。利用重定向技术可以消除这种不一致。

重定向是由服务器和客户端浏览器共同完成的。首先,服务器并不直接将目标视图的内容呈现给客户端,而是先向客户端发送一个HTTP重定向响应。该响应仅包含状态行和响应头,其中状态码为302,表示重定向;响应头包括Location域,指定重定向的目标视图的URL。其次,浏览器接收到重定向响应后,会自动向目标页面发出请求,这一过程并不需要用户的介入。最终,浏览器地址栏中显示目标视图的URL,浏览器窗口中显示目标视图的内容。

可以看出,重定向比原来的情况多了一个"请求—响应"过程,所以响应速度会变慢,但它可使浏览器上显示的页面内容与其地址栏中的URL保持一致。

当采用规则导航时,可以在navigation-case元素中插入redirect子元素,要求JSF框架重定向至目标视图。例如:

```
<navigation-rule>
    <from-view-id>/login.xhtml</from-view-id>
    <navigation-case>
        <from-outcome>ok</from-outcome>
        <to-view-id>/index.xhtml</to-view-id>
        <redirect/>
    </navigation-case>
</navigation-rule>
```

如果采用隐式导航,可以在结果值中添加一个名为faces-redirect、值为true的请求参数,以便实现重定向。下面是代码示例:

```
<h:commandButton value="确认" action="index?faces-redirect=true"/>
```

此时,问号(?)前面的index被取出以确定目标视图ID,而faces-redirect=true则指

示 JSF 框架重定向至目标视图。

5.5 h:link 与 h:button 标记

h:link 标记与 h:button 标记统称为结果类标记，其相应组件类有共同的超类 UIOutcomeTarget，在该超类中定义了两个组件共同的组件族名 OutcomeTarget。

结果类标记组件在呈现时，都会先根据 outcome 属性计算结果值，然后由导航处理器根据结果值确定目标视图。outcome 属性值可以是表示结果值的字符串，也可以是一个 EL 值表达式，结果值通过计算该值表达式获得。

与 h:commandButton 和 h:commandLink 标记不同，结果类标记 h:link 和 h:button 不需要置于 h:form 标记内，它们不会收集表单数据产生回送请求。结果类标记产生直接请求。

5.5.1 h:link

该标记在服务器端表示为一个 HtmlOutcomeTargetLink 组件实例。组件呈现为 HTML a 元素，其 href 属性值为为导航处理确定的目标视图的 URL。

单击由该标记呈现的超链接将产生一个对目标视图的直接请求。

标记组件呈现的超链接的文本既可以由该标记组件 value 属性指定，也可以由嵌套的文本或 h:outputText 标记指定，或者由嵌套的 h:graphicImage 标记指定超链接图像。

5.5.2 h:button

该标记在服务器端表示为一个 HtmlOutcomeTargetButton 组件实例，组件呈现为 HTML input 元素。默认情况下，input 元素的 type 属性值为“button”，即产生一个文本按钮，标记的 value 属性值指定按钮标题。若为标记指定了 image 属性，则 input 元素的 type 属性值为“image”，即产生一个图像按钮，input 元素的 src 属性值为标记的 image 属性指定的图像资源的 URL。无论哪种情况，input 元素都会包含一个 onclick 属性，其中的 JavaScript 代码可以产生超链接导航行为。

单击由该标记呈现的按钮将产生一个对目标视图的直接请求。

5.5.3 常用属性

除了 outcome、value 属性以及 id、render、binding、title、style、styleClass 等通用属性外，结果类标记还包含下面一些常用属性。

(1) fragment：用于指定目标页面某个位置(锚点)的名称，使得当导航到目标页面时能直接定位于该属性指定的页面位置。

(2) disabled：该属性适用于 h:link 标记，但不适用 h:button 标记，默认值为 false。若设置为 true，h:link 标记组件将呈现为 HTML span 元素，而不是 HTML a 元素。

(3) target：该属性适用于 h:link 标记，但不适用 h:button 标记，用于指定目标资源打开的位置。默认情况下，目标资源会在当前窗口打开。

5.6 规则导航应用示例

本应用项目(ch5_navigation)包含一个 Faces 配置文件、一个受管 bean 类和三个 JSF 页面。应用的运行效果如图 5-1 所示。

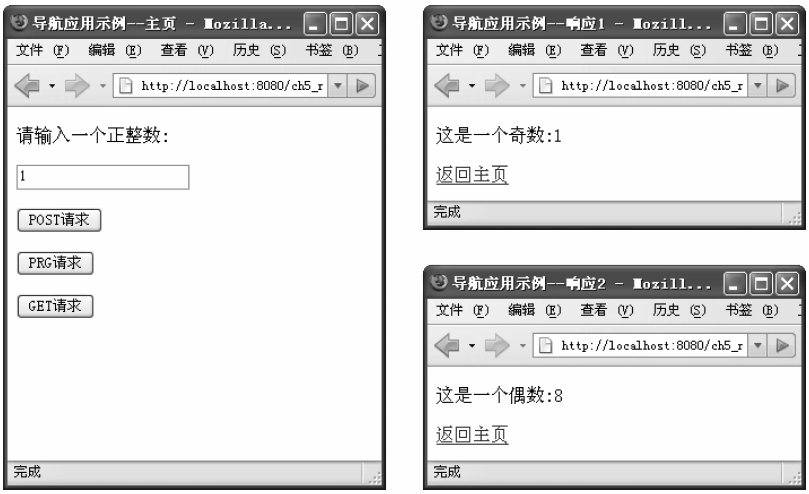

图 5-1 应用 ch5_navigation 运行效果

1. JSF 页面

该应用共有三个 JSF 页面。主页 index.xhtml(代码清单 5-1)主要包含一文本域和三个按钮。“POST 请求”和“PRG 请求”两个按钮都是由相应的 h:commandButton 标记产生的,“GET 请求”则是由 h:button 标记产生的。

代码清单 5-1 主页(index.xhtml)

```
1. <?xml version='1.0' encoding='UTF-8' ?>
2. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3.   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4. <html xmlns="http://www.w3.org/1999/xhtml"
5.     xmlns:h="http://java.sun.com/jsf/html">
6.   <h:head>
7.     <title>导航应用示例--主页</title>
8.   </h:head>
9.   <h:body>
10.    <h:form id="f">
11.      <p><h:outputLabel for="i" value="请输入一个正整数:"/></p>
12.      <p><h:inputText id="i" value="#{myBean.num}"/></p>
13.      <p><h:commandButton value="POST 请求"/></p>
14.      <p><h:commandButton value="PRG 请求" action="#{myBean.action}"/></p>
15.    </h:form>
16.    <p><h:button value="GET 请求"/></p>
```

```
17.   </h:body>
18. </html>
```

应用的另外两个页面分别是 response1. xhtml 和 response2. xhtml。这两个页面的功能都是显示受管 bean 中 num 属性的值，其中 response1. xhtml(代码清单 5-2)只是当 num 属性值为奇数时才被呈现，而 response2. xhtml(代码清单 5-3)则是当 num 属性值为偶数时才被呈现。之所以引入两个响应页面而非一个响应页面，完全是为了说明导航过程的需要。

代码清单 5-2　response1. xhtml

```
 1. <?xml version='1.0' encoding='UTF-8' ?>
 2. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 3.   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
 4. <html xmlns="http://www.w3.org/1999/xhtml"
 5.     xmlns:h="http://java.sun.com/jsf/html">
 6.   <h:head>
 7.     <title>导航应用示例--响应 1</title>
 8.   </h:head>
 9.   <h:body>
10.     <p>这是一个奇数:#{myBean.num}</p>
11.     <h:link value="返回主页" outcome="index"/>
12.   </h:body>
13. </html>
```

代码清单 5-3　response2. xhtml

```
 1. <?xml version='1.0' encoding='UTF-8' ?>
 2. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 3.   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
 4. <html xmlns="http://www.w3.org/1999/xhtml"
 5.     xmlns:h="http://java.sun.com/jsf/html">
 6.   <h:head>
 7.     <title>导航应用示例--响应 2</title>
 8.   </h:head>
 9.   <h:body>
10.     <p>这是一个偶数:#{myBean.num}</p>
11.     <h:link value="返回主页" outcome="index"/>
12.   </h:body>
13. </html>
```

两个响应页面中的“返回主页”超链接都由 h:link 标记产生，且都采用静态的隐式导航。下面主要讨论由主页到响应页面的导航过程。

2. 导航过程分析

由主页到响应页面的导航采用动态的规则导航。

首先给出应用中唯一的受管 bean 类的代码(代码清单 5-4)，因为其中的一些方法和属性会在导航规则中被引用。其中，getValid()方法判断 num 属性值是否有效，即若属性值为正整数，返回 true，否则返回 false；getNextViewID()方法用于动态计算目标视图 ID。

代码清单 5-4　受管 bean(MyBean. java)

```
1. package bean;
2.
3. public class MyBean {
4.
5.   private int num=1;
6.   public int getNum(){
7.     return num;
8.   }
9.   public void setNum(int num){
10.    this.num=num;
11.  }
12.
13.  public String getNextViewID(){
14.    System.out.println(num);
15.    if(num%2==0){
16.      return "/response2.xhtml";
17.    } else {
18.      return "/response1.xhtml";
19.    }
20.  }
21.  public boolean getValid(){
22.    return num>=1;
23.  }
24.  public String action(){
25.    return null;
26.  }
27. }
```

然后给出应用的 Faces 配置文件内容(代码清单 5-5),其中包含受管 bean 的声明和导航规则的声明。

代码清单 5-5　faces-config. xml

```
1. <?xml version='1.0' encoding='UTF-8'?>
2. <faces-config version="2.0"
3.   xmlns="http://java.sun.com/xml/ns/javaee"
4.   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5.   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
6.         http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd">
7.   <managed-bean>
8.     <managed-bean-name>myBean</managed-bean-name>
9.     <managed-bean-class>bean.MyBean</managed-bean-class>
10.    <managed-bean-scope>session</managed-bean-scope>
11.  </managed-bean>
12.  <navigation-rule>
13.    <from-view-id>/index.xhtml</from-view-id>
```