Stripes快速入门

白 汉奇

Stripes快速入门

白 汉奇

目录

第 1 章 Stripes 框架简介

简单的介绍 *Stripes* 框架

本章会对 Stripes 框架作一个简短的介绍。为什么我们需要Stripes？与我们所熟悉的 Struts 1 和Struts 2 相比，它有哪些优势？

1.1. 初识 Stripes

和我们熟悉 Struts 1 和 Struts 2 类似，Stripes 同样是一种展示层框架，用于快速构建web程序。在使用Struts 1，WebWork 和 Struts 2 等框架的时候，通常需要大量额外的 XML 配置，当一个项目达到一定规模的的时候，编写 Java 代码的同时还要维护大量的 XML 配置，这的确是件费神的事。Stripes 完全抛弃了这些框架的弊病，使用了最新的 Java 5 带来的技术，遵循 “Convention over Configuration” 理念，只需要在 Java 代码中加入少量的 Annotation，就可以完成配置，大量减少了代码的维护工作。

从 Stripes 网站，可以了解到 Stripes 框架的目标：

- 简化 Java web 开发。
- 针对一些常见问题，提供简单而强大的解决方案。
- 容易上手，你很难想像让一个新 Stripes 的用户在30分钟内就能很快进入状态。
- 容易扩展。

Stripes 提供的特性。

- 零配置，不需要外部配置文件，这是 Stripes 最引人注目的特性。
- 强大的binding引擎，足以应对复杂的对象。
- 验证和类型转换机制非常容易使用和本地化。
- 良好的本地化支持，甚至在 JSP 页面之间跳转时仍然生效。
- 能够复用 ActionBean，将它作 view helper 使用。
- 简易的 indexed property 支持。
- 内置支持同一个 form 触发多个事件。
- 具备透明的文件上传能力。
- 支持增量开发。
- 相当灵活，易于扩展。

1.2. Stripes vs Struts 1

作为一种经典的 MVC 框架，Struts 1 曾经在 Java 开发人员心目中占据了最高点。但是随着时间的推移，Struts 1 基础构架上先天性的缺陷显得越来越明显。同时，JSF 作为标准框架成为 Java EE 标准的一部分，Struts 1 面临着前所未有的挑战。一些开发人员开始尝试使用其它 web 框架，来替代 Struts 1。Struts 1 的作者也曾尝试对 Struts 1 彻底改造，并提出了雄心勃勃的 “Struts Ti” 计划，

但那就意味着 Struts 1 良好的向前兼容性将被打破，升级带来的代价非常大。所幸的是，一次机缘，让 Struts 1 和 WebWork 的开发人员能够坐下来，一起探讨 web 框架的未来。很快他们达成了共识，于是全新的 Struts 2 应运而生，它是基于 WebWork 的全新的 web 框架，新的 WebWork的开发工作也转移到了 Struts 2上。

和 WebWork 一样，Struts 1 作为一个遗留项目被保留下来。Struts 的另一个分支 Shale 从Struts 项目中剥离了出去，成为 Apache 下一个独立的项目。

在 web 开发中，与 Struts 1 相比，Stripes 有很多闪亮的新特性。

- 大大减少了开发工作量

  在 Struts 1 中，开发一个带有form的页面，你需要开发一个 JSP 页面文件，一个 `ActionForm` 类，一个 `Action` 类，并要在 `struts-config.xml`中注册你新建的 ActionForm 和 Action。

  而 Stripes 就要简单得多，你只需要创建一个 JSP 文件和 一个 `ActionBean` ，在 `ActionBean` 使用工 Annotation `UrlBinding` 指定你要响应的 URL ，使用 Annotation `HandlesEvent` 将事件映射到方法上。

- 增量开发

  在 Struts 1 中，当你写发一个 JSP 页面文件，其中有一个 `<html:form>` 标签，这时你还没有开发 `FormBean` ，当你把它部署到服务器，比如tomcat。打开浏览器，查看页面，它会抛出异常，提示找不相应的 `FormBean`。

  而 Stripes 中，完全不用担心，即使没有创建 `ActionBean`，你一样可以预览 JSP 页面的效果，然后再创建 `ActionBean`。

- 属性绑定

  在 Struts 1 中，你的 `Form` 有一个嵌套(nested)属性 `person.address.line1` ，如果 `person` 或 `address` 为 `null`，Struts 1 就会抛出异常，你必须在使用嵌套属性的对象中对其一一对预先初始化，这就是意味需要一些额外的工作。

  而 Stripes 中，完全不用担心没有初始化的问题，Stripes 会自动进行初始化，只要定义一个不带参数的修饰符为 `public` 的构造器。

- Indexed 属性

  在 Struts 1 中，完全屈从于JavaBean 规范，但对 web 模型不是很友好。你必须实现indexecd getter 和 setter 方法。

  而 Stripes 中，你只要提供你所用的 List 或 Map 的 getter 和 setter 方法。使用泛型告诉 Stripes List 和 Map 中对象的类型。Stripes 会在必要的创建一个 List 或 Map 实例，需要进行扩展，产生新的对象，并将他们放入到 List 或 Map 中，设置好 List 中对象的属性。

- Null的处理

  在 Struts 1 中，`ActionForm` 有一个 int 的属性，当时你保留页面字段为空时，Struts 1 会自动将转换成 0 ，更糟糕的，如果验证失败，回到 Form 页面，它会在输入框填充一个 0 。

  而 Stripes 中，如果页面输入框为空，那么这个字段不提交，`ActionBean` 不会填充字段值。

- 格式化输出

  在 Struts 1 中，你只有使用 JSTL 提供的格式化功能 。

而 Stripes 中，和 JSTL类似，提供了格式化标签。

- 多事件支持的 `Action`

  在 Struts 1 中，你必须使用 `DispatcherAction` 来实现，并且所有 button 必须使用同一名称，不同的值，它会根据值的不同来调用的不同的方法，如果button 要进行国际化时，又是一件痛苦的事。

  而 Stripes 中，使用 button 的 name 属性本身来映射相应的方法。

- JSP/View Helper

  在 Struts 1 中，没有什么好的方法将动态数据载入到 JSP 页面之中。

  而 Stripes 中，`ActionBean` 可以作为一个 View Hepler使用。

- HTML 标签

  在 Struts 1 中， Struts 1 标签中 input 使用 `property`而不是 `name` ，`class` 被替换成 `styleCalss`。

  而 Stripes 中，尽可能保持这些属性的名称与 HTML 一致。

1.3. Stripes vs Struts 2

前不久 Apache 官方刚刚发布了 Struts 2 的最新稳定版 2.1.6.GA，在这个版本中，官方首次提供了一个 Convention Plugin 插件，以代替 2.0.x 中的 Codebehide，Zero Config 等 Annotation 插件。从名称可以看出其一个重要的特色就是 Convention 。不过，感觉 Struts 2 有“复制” Stripes 之嫌。

在 Stripes 中，可在 `StripesFilter` 中指定一个 `init-param` 参数 `ActionResolver.Packages` ，指定从哪些包中扫描 `ActionBean`。

```
<init-param>
          <param-name>ActionResolver.Packages</param-name>
          <param-value>tutorial.action</param-value>
</init-param>
```

在 Struts 2 中，有相似的概念。

你可以在 Struts 2 配置文件中指定一个 `struts.convention.action.packages` 常量，其值为 Struts 2 中 `Action` 所在的包。

第 2 章 Stripes 快速入门

创建你的第一个 *Stripes* 项目

本章介绍如何准备一个 Stripes 开发环境，并创建第一个 Stripes 项目。

2.1. 安装 Sun JDK

既然 Stripes 是一个基于 Java 5 的 web 框架，你必须拥有一个 Java 5 版本以上的开发环境。

请从 Sun官方网站下载最新的 JDK 5 或者 6，安装到你的系统上，并设置好 `JAVA_HOME` 环境变量，指向 JDK 的安装位置，不要忘记把 `JAVA_HOME/bin` 加入到系统的 Path 变量中。

> **注意**
>
> 现在主流的 Linux 的发行版本的软件库中都已经包含了开源的Java 运行环境 OpenJDK，你可以直接使用系统内置的安装工具安装即可。如在 Ubuntu 下，你可以通过 **apt-get install openjdk-*** 来安装，Ubuntu 还带有 Sun 官方的 JDK。在 Fedora 系统中，可以通过 **yum install java-1.6.0-openjdk java-1.6.0-openjdk-devel** 来安装 Java 开发环境。

打开命令行工具，输入 **java -version** 检测你的 Java 环境是否安装正确并且生效。

```
[hantsy@localhost ~]$ java -version
java version "1.6.0_12"
Java(TM) SE Runtime Environment (build 1.6.0_12-b04)
Java HotSpot(TM) Client VM (build 11.2-b01, mixed mode)
```

2.2. 使用 Maven 创建项目

首先你要安装 Maven 环境。

2.2.1. 安装 Maven

Maven 是 Apache 旗下一个 Java 项目构建工具，它除内置了一套完整的构建生命周期外，还有完善的包依赖管理功能。

从 Apache 网站下载最新的 Maven，解压到硬盘上。设置环境变量 `M2_HOME`，并将 `M2_HOME/bin` 加入到系统的path中。

在 Linux 系统下，你可以直接将其写入你的用户配置文件中。

打开 `.bashrc`，在文件末尾追加以下几行代码。

```
export M2_HOME=/opt/build/maven
export PATH=$PATH:$M2_HOME/bin
```

在命令行中输入**mvn -v** 来验证 Maven 安装。

```
[hantsy@localhost ~]$ mvn -v
 Maven version: 2.0.10
 Java version: 1.6.0_12
```

```
OS name: "linux" version: "2.6.27.19-170.2.35.fc10.i686" arch: "i386" Family: "unix"
```

2.2.2. 创建 Stripes 项目

在确定你已经安装好Maven后，就可以开始创建 Stripes 项目了。

mvnstripes 提供一个 Stripes 的 Maven artifact 。但这个 artifact 目前还没有进入 Maven 中心资源库，你可以sf.net下载，然后安装到本地资源库中。

> **提示**
>
> Maven 中，artifact 提供了一个简单项目模板，通过它可以很快的构建项目。在第 2.3 节 “使用 NetBeans 创建项目”中，将会介绍完全以手动方式创建项目，这时你就会体会到 Maven 带来的便利。

转到你下载的文件即artifact所在的目录，执行下面命令。

```
mvn install:install-file \
-Dfile=stripes-archetype-quickstart-1.0.jar \
-DgroupId=net.sourceforge \
-DartifactId=stripes-archetype-quickstart \
-Dversion=1.0 -Dpackaging=jar
```

现在你可以通过这个 artifact 创建一个 Stripes 项目了。

```
mvn archetype:generate \
-DarchetypeArtifactId=stripes-archetype-quickstart \
-DarchetypeGroupId=net.sourceforge \
-DarchetypeVersion=1.0 \
-DgroupId=tutorial \
-DartifactId=helloworld
```

稍等片刻，一个名为 helloworld 项目就创建好了。

> **注意**
>
> 如果你是初次运行 Maven ，这一过程可能要花几分钟时间，Maven 会自动从远程 Maven 中心资源库中下载所这个 artifact 的所有依赖到本地资源库，默认情况下，本地资源库的目录是 `~/.m2/repository` 。

进入到 helloworld 项目目录，运行 **mvn jetty:run** ，从终端中看到 Jetty 运行完毕之后，就可以打开浏览器，输入地址 http://localhost/helloworld 。如果看到如下类似的信息，恭喜你，你已经成功的创建了第一个 Stripes 项目，并且已经在服务器运行起来了。

```
Congratulations, you've set up a Stripes project!
```

2.3. 使用 NetBeans 创建项目

NetBeans 是 Sun 主导开发一款开源的开发环境。你可以从 NetBeans 的官方网站下载最新 6.5 稳定版本。针对不同的开发人员，NetBeans 提供了多种特性组合，针对 Java web 开发，你可以选择下

载 Java 组合包，它已经包含 Java EE 和 web 开发所需要的特性，并包含了流行的开源应用服务器 Glassfish 和 Tomcat。

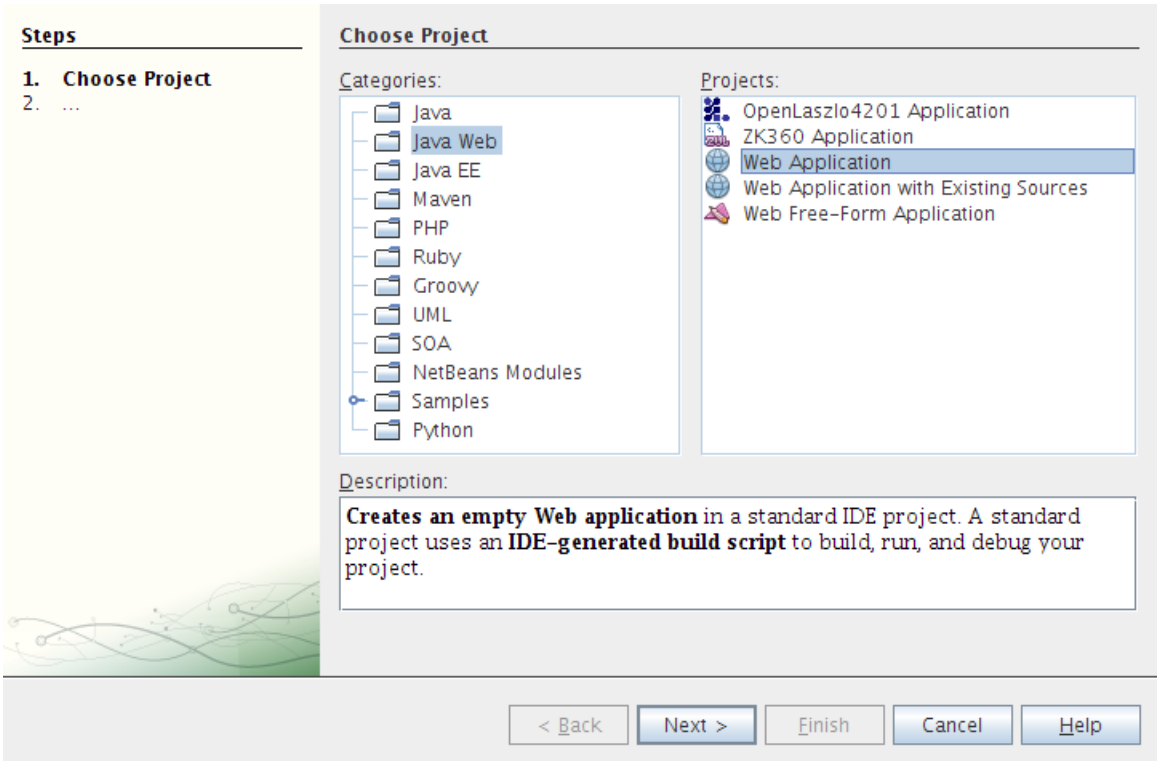
NetBeans提供了友好的安装界面，你只要根据安装向导一步步进行。安装完毕，你可以准备创建一个 web 项目。

> **提示**
> NetBeans 同样提供了 zip 包，直接解压便可以使用。NetBeans 是纯 Swing 开发的程序，如果你在使用多系统的话，就不用下载针对平台的安装程序，节省了不少空间。

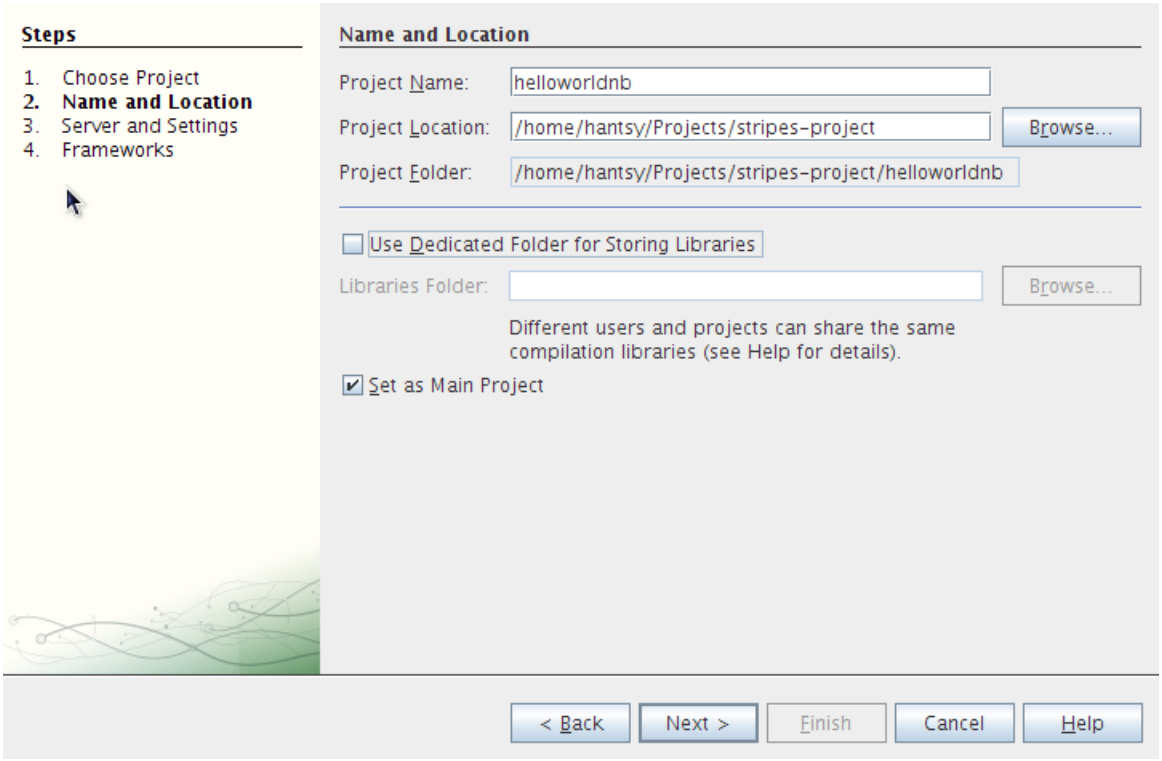
用 NetBeans 创建一个 Stripes 的步骤。

- 创建一个 web 项目。
- 添加 Stripes 所依赖的jar文件。
- 在 `web.xml` 中注册相应的 StripesFilter 和 StripesServlet。

启动 NetBeans ，从 IDE 主菜单中点击File/New Project 打开新建项目窗口。

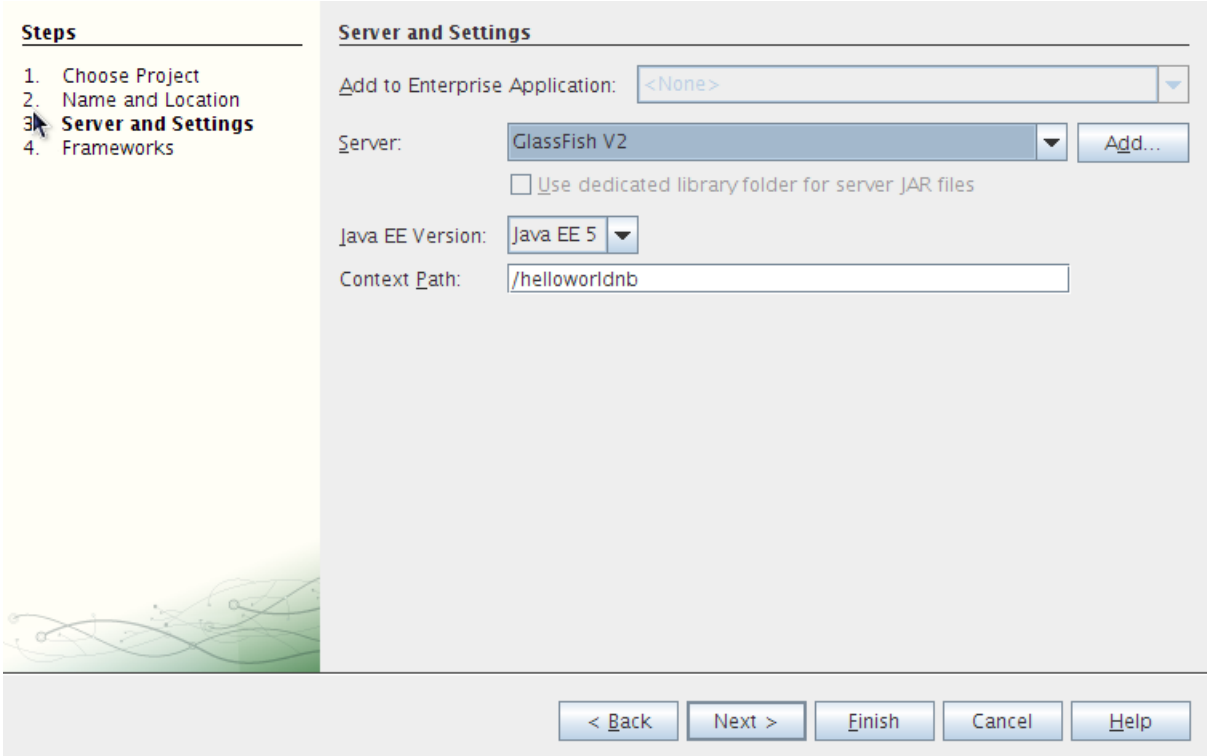


从对话框中左边 Categories 中选择 “Java Web”，右边 Projects 中选择 “Web Application”，点击 “Next” 进入下一步。



输入项目名称和位置。如果你想与其它项目共享 lib ，可以点击 “Use Dedicated Folder for Storing Libraries” 前面的 checkbox ，指定 Libraries Folder 目录。对于 Stripes 项目，你可以将 Stripes 所需要的jar放到同一个文件夹中，供所有的Stripes项目共享使用。但这里我不打算这么做，我选择更符合 NetBeans 的方式，使用 NetBeans 的 Libraries Manager 创建一个 Stripes 包。

点击“Next”进入下一步，设置服务器属性。



在此对话框中，你可以设置运行环境，Java EE 版本，和 Context path，点击“Finish” 完成 web 项目的创建。

现在可以从Projects窗口中，右键点击新建的 helloworldnb 项目节点，点击“运行”来运行项目。

下面开始对刚刚新建的 helloworldnb 添加 Stripes 支持。

首先要新建一个 NetBeans 管理的 Stripes 包。你可以从 Stripes 官方网站下载最新的 Stripes 1.5.1，解压到本地硬盘。从主菜单选择 Tool/Libraries ，打开 Libraries 窗口。点击 “New Library...”，输入名称“Stripes”。点击“Add Jar/Folders”，添加 Stripes jar文件。将 `stripes/lib/deploy` 中的 `commons-logging.jar` 和 `cos.jar` ，还有`dist`中的 `stripes.jar`添加到 Stripes 包定义中。

在Projects 窗口中点击 helloworldnb 项目根点，展开项目。在“Libraries”节点上点击右键，选择“Add Library”，从弹出窗口的列表中找到刚刚新建的 Stripes 包，点击“OK”确定。

打开 `Configurations Files/web.xml`，在web描述文件注册相应的 Stripes 的Filter和 Servlet。

```
<filter>
  <filter-name>StripesFilter</filter-name>
  <filter-class>net.sourceforge.stripes.controller.StripesFilter</filter-class>
  <init-param>
   <param-name>ActionResolver.Packages</param-name>
   <param-value>tutorial.action</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>StripesFilter</filter-name>
  <url-pattern>*.jsp</url-pattern>
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
```

```
<servlet>
  <servlet-name>DispatcherServlet</servlet-name>
  <servlet-class>net.sourceforge.stripes.controller.DispatcherServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>

<filter-mapping>
  <filter-name>StripesFilter</filter-name>
  <servlet-name>DispatcherServlet</servlet-name>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>FORWARD</dispatcher>
</filter-mapping>

<servlet-mapping>
  <servlet-name>DispatcherServlet</servlet-name>
  <url-pattern>*.action</url-pattern>
</servlet-mapping>
```

> **提示**
> 现在有一个 Stripes for NetBeans 项目为 NetBeans 提供了一个 Stripes 插件，可以帮你完成这些操作。

2.4. 开始创建Stripes项目

现在 Stripes 开发环境已经准备就绪，不管你是喜欢用 Maven 还是习惯使用 NetBeans IDE 来构建项目。

新建一个 `ActionBean` 类，实现 `net.sourceforge.stripes.action.ActionBean` 。

```
package tutorial.action;

import net.sourceforge.stripes.action.ActionBean;
import net.sourceforge.stripes.action.ActionBeanContext;
import net.sourceforge.stripes.action.ForwardResolution;
import net.sourceforge.stripes.action.Resolution;

/**
 *
 * @author hantsy
 */
public class HelloActionBean implements ActionBean{
    ActionBeanContext context;
    public void setContext(ActionBeanContext context) {
        this.context=context;
    }

    public ActionBeanContext getContext() {
        return this.context;
    }

    String message;

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public Resolution sayHello(){
```

```
        return new ForwardResolution("/greeting.jsp");
    }
}
```

将默认的 index.jsp 页面内容修改成如下。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
   <%@taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello Stripes!</h1>
        <stripes:form beanclass="tutorial.action.HelloActionBean" method="post">
            <div><stripes:text name="message"/></div>
            <div><stripes:submit name="sayHello" value="Say Hello"/></div>
        </stripes:form>
    </body>
</html>
```

创建一个名为greeting.jsp 的结果页面。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello World!</h1>
        <div>Message: ${actionBean.message}</div>
    </body>
</html>
```

现在可以开始运行项目。在首页输入一条信息，点击 SayHello 按钮，在结果页面你就可以看到你刚才输入的信息。

2.5. Stripes的运行原理

你现在已经创建第一个 helloworld 实例，并且能够在服务器运行起来，但是，helloworld 是如何在服务器上运行的呢？

你可能已经注意到，在 `web.xml` 我们注册一个 `StripesFilter` 和 `DispatcherServlet` 。在我们熟悉的 web 框架中，如 Struts 1 和 Struts 2，很少同时使用 Filter 和 Servlet 的。如在 Struts 1 中使用一个 ActionServlet 作为控制器，处理页面请求。在 Struts 2 中，使用了一个 Filter 处理所有

请的请求。在 Stripes StripesFilter 主要是为请求处理作一些准备工作，它负责读取配置，处理 Locale ，并根据页面的 form 取得相应的 ServletWrapper 包装类。

你可以也注意到 StripesFilter 同时会过滤所有的 JSP 文件和 DispatcherServlet。 如果请求的是一个 jsp 页面，如我们的首页 index.jsp ，在 StripesFilter 处理完后会直接跳转到相应的JSP页面。如果请求是一个 `ActionBean` 的某个 event ，后续处理工作会交给 DispatcherServlet 来进行。

当 helloworld 运行起来之后，如果查看首页 `index.jsp` 的源代码，你会发现，在 `stripes:form` 标签中设置的 `beanclass="tutorial.action.HelloActionBean"`，转换了 `action="/helloworld/Hello.action"`，Stripes 根据自己规则为你生成了 URL。

在 helloworld 程序中，index.jsp 提交时，DispatcherServlet 会根据 URL 来创建一个 `ActionBean` ，将 HttpServletRequest中的参数绑定 `ActionBean` 的属性上。然后执行相应的 event 对应的方法。而这 event 的名称正是 index.jsp 中页面 form 中提交按钮的 name 属性值。

> **注意**
> 这里，我们跳过了一些细节没有提及，如数据类型转换，输入验证。以后会慢慢深入了解。

第 3 章 创建 ActionBean

使用 *ActionBean* 处理客户端请求

ActionBean 是 Stripes 的核心，本章介绍如何有效的使用 ActionBean，让 web 开发更加简单。

3.1. ActionBean 接口

在 web.xml 中注册 StripesFilter 时，需要指定一个参数ActionResolver.Packages，其值是 ActionBean 所在的包名，你可以使用指定多个包，中间用逗号隔开。

```
<init-param>
  <param-name>ActionResolver.Packages</param-name>
  <param-value>tutorial.action</param-value>
</init-param>
```

Stripes 会扫描参数值(这里是tutorial.action)中包中的 ActionBean 类。

可能你已经注意，你的 HelloActionBean 实现了 ActionBean 接口，这是必须的。

```
public class HelloActionBean implements ActionBean{
 ...
}
```

让我们再来看一下 ActionBean 接口定义。

```
public interface ActionBean {
      public ActionBeanContext getContext();
      public void setContext(ActionBeanContext context);
}
```

看起来很简单，仅仅提供了 getter 和 setter 两个方法。在运行环境时，Stripes 会通过 setter 方法注入一个 ActionBeanContext 实例，你可以通过 getter 方法来取得一个 ActionBeanContext 对象。通过 ActionBeanContext 你可以访问 HttpServletRequest，HttpServletResponse 和 ServletContext 对象。这里就可以看出，为了简化开发，一般情况下，Stripes 将它们隐藏起来，同时也提供了简单的方法，在需要的时候访问它们。

在实际开发中，可以用一个公用类来访问来实现 ActionBean 接口，其它具体的 ActionBean 这个基类继承。

```
public abstract class BaseActionBean implements ActionBean {
    private ActionBeanContext context;

    public ActionBeanContext getContext() {
        return context;
    }
    public void setContext(ActionBeanContext context) {
        this.context = context;
    }
```

```
}
```

3.2. 处理请求事件

Stripes 是一个基于 action 框架，但它也包含了一些基于事件的框架的概念。在 helloworld 实例中，页面中一个 FORM 是由一个 `ActionBean` 负责进行处理，提交时会触发一个事件，submit 按钮的名称(sayHello)映射到 `ActionBean` 的某个方法，这个方法称为 event handler。

下面是一个 Event Handler 的定义。

```
public Resolution sayHello(){
        return new ForwardResolution("/greeting.jsp");
}
```

它是一个无参数的方法，返回一个 `Resolution` 。

默认情况下，submit 按钮的 name 属性值就是相应的 `ActionBean` 的方法名。你可以在方法使用 Annotation `HanlesEvent` 来指定 `ActionBean` 的方法来处理那些事件。

```
@HandlesEvent("sayHello")
public Resolution greeting(){
        return new ForwardResolution("/greeting.jsp");
}
```

在上面的代码中，用 greeting 方法来处理 sayHello 事件请求。

另外一个可能会用到 Annotation 是 `DefaultHandler`，当请求中没有提供相应的事件名称时，会执行带有 `DefaultHandler` Annotation 的方法。如果 `ActionBean` 只一个事件处理方法，那么这个方法就是默认的。

3.3. URL 映射

可能你已经注意到，在 helloworld 的 `index.jsp`，在 `stripes:form` 使用了一个 `beanclass="tutorial.action.HelloActionBean"` 属性，在运行时，它自动为你生成了一个非常友好的 URL 。

Stripes 提供一套有效的机制，根据 `ActionBean` 类名和 包名生成 URL 地址。它会将`ActionBean` 的包名转换成路径添加到 URL中，再添加 `ActionBean` 类名，最后加一个后缀 `.action`。你可以会产生疑问，觉得 hellworld中生成的 URL 不符合这一规则。在URL处理时，为了使用URL 看起来更加简洁，Stripes 作了以下处理。

- 将包名转换成路径，但是忽略包中有'web'， 'www'， 'stripes' 和 'action' 子包之前的部分，只截取其后的部分。
- 在 URL 中添加类名，如果类名是以 `Action` ， `Bean` 或者是 `ActionBean` 结尾，去掉这些后缀。
- 最后添加一个 `.action` 后缀，与 `web.xml` 中定义一致。

现在你不会觉得奇怪了，`tutorial.action.HelloActionBean`的蝶变过程如下。

- tutorial.action.HelloActionBean
- /tutorial/Hello
- /tutorial/Hello.cation

这里我们忽略了前面的Servlet Context Path("/helloworld")。

如果你不愿意使用它生成的URL ，可以自定义 URL，在 `ActionBean` 类上使用 Annotation `UrlBinding`，在参数中指定你的自定义的 URL。

```
@UrlBinding("/SayHello")
public class HelloActionBean implements ActionBean{
 ...
}
```

重新运行这个程序，你可以发现生成的文件，已经替换成你的 URL。

第 4 章 输入控制

如何处理表单数据提交

本章介绍表单提交，数据类型转换，数据有效性验证等。

4.1. 处理表单

当你提交一个表单时，Stripes 会进行以下处理。

- 首先会检查要求必填 (required) 的字段是否已经填充。
- 如果必填字段没有出现错误，则执行下面步骤。
  1. 执行最小/最大长度检测和模式匹配验证。
  2. 如果上一步没有错误出现，将字段值转换成相应的类型，并绑定到`ActionBean`的属性上。
  3. 如果类型转换过程没有出现错误，执行最小值/最大值检测(针对数值型)。
- 执行自定义的检测方法。

如果在验证和数据转换过程中出现错误，默认情况下 Stripes 会返回原页面，所有数据都会重新填充到表单中，并可以在页面显示错误信息。

4.2. 用户注册程序

为了节省时间你可以复制一份helloworld项目。在NetBeans中，直接在项目结点，右键点击“Copy”，输入新项目名称 registration 。

在这个项目中，将会创建一个resgiter.jsp 页面，一个打印结果的页面 success.jsp和一个注册处理的`ActionBean` 。

创建 `register.jsp` 页面。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Registeration Page</title>
    </head>
    <body>
        <h1>User Registeration</h1>
        <div>
            <stripes:form beanclass="tutorial.action.RegisterActionBean">
               <div>Username: </div><div><stripes:text name="username"/></div>
               <div>Password: </div><div><stripes:password name="password"/></div>
               <div>Confirm Password: </div><div><stripes:password name="confirmPassword"/></div>
               <div>Email: </div><div><stripes:text name="email"/></div>
               <div><stripes:submit name="register" value="Register Now"/> </div>
            </stripes:form>
        </div>
```

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

创建一个 ActionBean ,名为 RegisterActionBean 。

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package tutorial.action;

import net.sourceforge.stripes.action.ForwardResolution;
import net.sourceforge.stripes.action.Resolution;

/**
 *
 * @author hantsy
 */
public class RegisterationActionBean extends BaseActionBean {

    private String username;
    private String password;
    private String email;
    private String confirmPassword;

    public String getConfirmPassword() {
        return confirmPassword;
    }

    public void setConfirmPassword(String confirmPassword) {
        this.confirmPassword = confirmPassword;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public Resolution register() {
        return new ForwardResolution("/success.jsp");
    }
}
```

创建一个结果页面 `success.jsp` 。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Registeration Result Page</title>
    </head>
    <body>
        <h1>Registered Successfully!</h1>
        <div>Your username is ${actionBean.username}. </div>
    </body>
</html>
```

在服务器上运行这个程序，体验一下流程。但是这个程序对输入信息没有作任何限制。下面我们会一步步的完善这个程序。

4.3. 添加 `required` 约束

在 RegisterActionBean 要求输入字段上加 `@Validate(required=true)`。

```
 @Validate(required=true)
    private String username;

    @Validate(required=true)
    private String password;

    @Validate(required=true)
    private String email;

    @Validate(required=true)
    private String confirmPassword;
```

如果验证出错，Stripes 会默认返回原页面，即 `register.jsp` 。

在 `register.jsp` 页面添加以下代码片断。

```
  <h1>User Registeration</h1>
  <div><stripes:errors/></div>
```

从 Stripes 包中复制一份 `StripesResources.properties` 到项目的源代码目录下。

运行程序，看看验证是否生效。浏览地址 http://localhost:8080/registration/register.jsp ，不输入任何字段，直接提交表单。如果你看到下面信息，那么验证已经生效。

```
Please fix the following errors:

   1. Confirm Password is a required field
   2. Email is a required field
   3. Password is a required field
```

```
   4. Username is a required field
```

4.4. 其它输入控制

对于输入的信息，还可以添加字符长度范围，及字符匹配条件。如 username，password 要求输入6－20位字符，并且必须是数字或者字母。

将 `RegisterActionBean` 中 `username`，`password`上的Annotation 作出一点修改。

```
  @Validate(required = true, minlength = 6, maxlength = 20)
    private String username;
    @Validate(required = true, minlength = 6, maxlength = 20)
    private String password;
```

重新运行程序，在 username 和 password 输入的字符如果不够6位，提交后你会看到如下错误信息。

```
Please fix the following errors:

   1. Password must be at least 6 characters long
   2. Username must be at least 6 characters long
```

这时你没有限制输入字符类型，可以借助正则表达式来完成。

```
  @Validate(required = true,  mask = "[0-9a-zA-Z]{6,20}")
    private String username;
```

这里限制 username 只能是字母或数字，并且长度是6到20。

重新运行程序，当 username 输入不符合要求。会看到类似下面的错误信息。

```
  Please fix the following errors:

 1. test is not a valid Username
```

4.5. 数据类型转换

在输入数据正式绑定 `ActionBean` 之前，需要将它们转换成相应的类型。

Stripes 内置了多个 TypeConverter。对于基本的数据类型，如byte，int，float，double，long，boolean 及其包装类 Byte，Integer，Float，Double，Long，Boolean 及 BigDecimal，BigInteger 都能自动转换成相应的类型。Date， Enum 等也有相应的 `TypeConverter`，`DateTypeConverter` 和 `EnumeratedTypeConverter`
。

Email 地址合法性检测常常被认为应该属于 validation 范围，Stripes 利用 TypeConverter 来检测其合法性，显得有些牵强。Stripes 提供了一个 `EmailTypeConverter`，实际它就是利用 `JavaMail` 的 API 检测地址的合法性。

Stripes 还提供另外两个其它框架少有 `TypeConverter`，`PercentageTypeConverter` 可以将百分比转换成实数。`OneToManyTypeConverter` 将字符串按分隔符转换成一个List。

修改 `RegiterActionBean`， 在 email字段上的Validate Annotation 中使用 `EmailTypeConverter` 。另外追回两个字段，一个 `java.util.Date` 类型的 `birthDate` ，另一个 `boolean` 类型的 `subscriptionEnabled` 。

```
@Validate(required = true, converter=EmailTypeConverter.class)
   private String email;

   @Validate(converter=DateTypeConverter.class)
   Date birthDate;

   @Validate(converter=BooleanTypeConverter.class)
   boolean subscriptionEnabled;
```

页面 register.jsp的修改。

```
<div>Birth Date: </div><div><stripes:text name="birthDate"/></div>
<div>Would like receive our product infomation by email? </div>
<div><stripes:checkbox name="subscriptionEnabled" value="on"/>If you would like, check on please. It is
```

在 `StripesResources.properties`中定义 Date 的格式。

```
stripes.dateTypeConverter.formatStrings=yyyy MM dd
```

4.6. 自定义验证方法

在表单输入数据转换完成后，就绑定到 `ActionBean` 的对应的字段上。

在这个程序中，还有其它需要验证的地方。如，两次密码一致性，注册用户名的唯一性。

```
   @Validate(required = true, expression="this eq password")
   private String confirmPassword;
```

一些验证需要与后台数据进行对比，比如检测用户名的唯一性。Stripes 提供了自定义的方法进行验证。

```
 @ValidationMethod(on="register")
   public void userExsited(ValidationErrors errors) {
       if("testuser".equals(username)){
               errors.add("username",
               new SimpleError("This username is taken , please select a different one."));
       }
   }
```

重新运行这个程序，检测验证是否生效。

4.7. 使用嵌套验证

如果 ActionBean 的属性是一个普通的 Java 类，如何进行验证？Stripes 提供了 ValidateNestedProperties Annotation 解决这个问题。

我们添加一个address字段，它是一个 Address类。

```
 @ValidateNestedProperties({
        @Validate(field="zipcode", required=true),
        @Validate(field="addressLine1", required=true),
        @Validate(field="addressLine2", required=true)
    })
    private Address address;

    public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }
```

在注册页面 register.jsp中，添加以下代码。

```
<fieldset>
 <legend>Address</legend>
 <div>ZipCode:</div>
 <div><stripes:text name="address.zipcode"  size="10"/></div>
 <div>Address Line1: </div>
 <div><stripes:text name="address.addressLine1"/></div>
 <div>Address Line2: </div>
 <div><stripes:text name="address.addressLine2"/></div>
</fieldset>
```

新建一个Address 类，它包含几个最基本的属性，zipcode ，addressLine1，addressLine2，并包含相应的 getter 和 setter 。

```
package tutorial.action;

/**
 *
 * @author hantsy
 */
public class Address {
    private String zipcode;
    private String addressLine1;
    private String addressLine2;


    public Address() {
    }

    public String getAddressLine1() {
        return addressLine1;
    }
```

```
    public void setAddressLine1(String addressLine1) {
        this.addressLine1 = addressLine1;
    }

    public String getAddressLine2() {
        return addressLine2;
    }

    public void setAddressLine2(String addressLine2) {
        this.addressLine2 = addressLine2;
    }



    public String getZipcode() {
        return zipcode;
    }

    public void setZipcode(String zipcode) {
        this.zipcode = zipcode;
    }

}
```

重新运行这个程序，验证信息。

4.8. 显示错误信息

在前面的例子中，我们都是显示全部错误信息。<stripes:errors/>　提供几个可选的属性`action`, `beanclass`, `field`, `globalErrorsOnly`。添加一个 action 或是 beanclass 可以指定显示哪个 form 的错误信息。设置field 参数可以精确指明哪一个输入字段的错误。globalErrorsOnly 可以设为 `true` 或 `false`，表明是否显示全局错误。

```
  ...
  <h1>User Registeration</h1>
        <div><stripes:errors globalErrorsOnly="true"/></div>
        <div>
            <stripes:form beanclass="tutorial.action.RegisterActionBean">
                <div>Username: </div><div> <stripes:text name="username"/><stripes:errors field="userna
        ...
```

这里我们在表单上面显示全局错误，在 `username` 字段上显示其详细信息。

在 `userExsited` 方法中添加一个 `globalError`。

```
@ValidationMethod(on="register")
    public void userExsited(ValidationErrors errors) {
        if("testuser".equals(username)){
                errors.add("username",
                new SimpleError("This username is taken , please select a different one."));
        }
        if(!errors.isEmpty()){
            errors.addGlobalError(
            new SimpleError("Error occurs while saving data. Please fix it firstly."));
        }
```

```
    }
```

如果你想完全自己定义错误信息，控制转向的页面。Stripes 提供了`ValidationErrorHandler`接口，它提供一个 `handleValidationErrors` 方法，你可以自己处理错误。

```
  public class RegisterActionBean extends BaseActionBean implements ValidationErrorHandler {
  ...
  public Resolution handleValidationErrors(ValidationErrors errors) throws Exception {
        return new ForwardResolution("/error.jsp");

    }
    ...
  }
```

这里如果出现错误，转向error.jsp。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Error!</h1>
        <div><stripes:errors/></div>
    </body>
</html>
```

重新运行这个程序，验证失败时，不再回到原来页面，而跳转到了error.jsp页面显示错误信息。

第 5 章 Resolution接口

在前面的例子你可以看到，ActionBean 中一个事件对应的方法返回一个 Resolution对象。

Resolution 接口定义。

```
public interface Resolution {

    void execute(HttpServletRequest request, HttpServletResponse response)
        throws Exception;
}
```

Resolution 提供了 HttpServletRequest 和 HttpServletResponse 的访问能力。

Stripes 提供了几种 Resolution 实现，ForwardResolution ，RedirectResolution，StreamingResolution，ErrorResolution。

在前面例子中已经用到了ForwardResolution，它最终调用RequestDispatcher的forward方法显示目标页面。

ForwardResolution 提供了几种构造方法，用于不同目的。

```
public ForwardResolution(String path) {

    }

    public ForwardResolution(Class<? extends ActionBean> beanType) {

    }


    public ForwardResolution(Class<? extends ActionBean> beanType, String event) {

    }
```

第一种，直接指定 URL 地址，这种方法简单明了。后两种不直接使用 URL ，它可以从一个 ActionBean 转向另外一个 ActionBean ，Stripes 会自动转向目标 ActionBean 和触发 ActionBean 事件所对应的 URL 地址。

RedirectResolution与ForwardResolution不同的是它调用 HttpServletResponse 的 sendRedirect方法显示目标页面。

StreamingResolution 不会将客户端转向另一个页面，它向客户端发送数据流。这个 Resolution 常常用于动态显示图片，显示图表，XML数据。它提供一个可选的 filename 属性，可以用于文件下载。如果设置了该属性，那么在输出时 Stripes 会在输出流中会在文件头写入 Content-Disposition信息，它指定下载文件的名称，在浏览器中会自动弹出下载文件窗口。StreamingResolution 会在后介绍使用。

ErrorResolution可以向客户端发送 HTTP 错误状态码和自定义的错误信息。

另外，Stripes 还提供了一种 JavaScriptResolution ，但奇怪的是这个 Resolution 没有从 StreamingResolution 继承。

第 6 章 防止重复提交

这里介绍一下 `RedirectResolution` 使用时应该考虑的几个问题。

想像一下，在一些电子购物网站中，如果在最后支付时，如果因为网络等因素，可能需要一段等待时间。如果你是个急性子，多点击了两下提交按扭。结果可能就是你为同一单物品，支付了两次，或者网站的处理是送给两份相同的物品。

这是你不愿意看到的，所幸的是，一些浏览器作了优化处理。比如 Firefox，在提交表单数据的过程中多次提交会视为同一次提交。但是，你仍然回避不了另外一个问题。提交完成之后，点击了浏览器刷新按扭，又会重新提交表单。

Struts 1 提供了一种简单的方法，来解决这个问题。在表单初始化时，设置一个隐藏的token值，提交时会比较token值。Stripes 没有提供这一方法。

6.1. 使用 RedirectResolution

使用 `RedirectResolution` 替代 `ForwardResolution`， 是最简单的方法。

将 `RegisterActionBean` 中的 `register`方法修改成如下。

```
public Resolution register() {
    return new RedirectResolution("/success.jsp");
}
```

重新运行程序，你会发现表单提交之后。浏览器地址栏中地址变成 `http://localhost:8080/redirect/success.jsp` ，而不是之前的 `http://localhost:8080/redirect/Register.action` 。你可能已经发现，另外还有一个问题就是，结果页面无法显示注册信息。

很多其它 web 框架提供了一个 Flash Scope来解决这个问题。Flash类似 Session，Request，它的生命周期比普通 Request 长，比 Session 短。Flash 通常也是基于 Session来实现，它跨越当前请求和下一个请求，保证了下一请求中还能读取当前请求中的属性，在下一请求结束后， Flash 就过期。Stripes 提供一个类似的实现，但是 Stripes 的实现的依赖 Session 过期。

你可以把 `ActionBean` 放入 Flash scope中。

```
public Resolution register() {
    return new RedirectResolution("/success.jsp").flash(this);
}
```

重新运行程序，进行测试。

6.1.1. FlashScope

Stripes 提供了一个 FlashScope 类，处理 Flash 状态。

你可以将用下面的方法将 `ActionBean` 添加到 Flash Scope 中。

```
FlashScope.getCurrent(getContext().getRequest(), true).put(this);
```

将其它对象放入 Flash Scope 也很简单。

```
public Resolution register() {
 addMessage("Registered successfully!");
 addMessage("Congratulations!");
 return new RedirectResolution("/success.jsp").flash(this);
}

public void addMessage(String message) {
 FlashScope scope = FlashScope.getCurrent(getContext().getRequest(), true);
 List<String> messages = (List<String>) scope.get("messages");
 if (messages == null) {
  messages = new ArrayList<String>();
  scope.put("messages", messages);
 }
 messages.add(message);
}
```

在 JSP 页面显示信息。

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
...
        <c:forEach var="msg" items="${messages}">
            <div class="msg">${msg}</div>
        </c:forEach>
        ...
```

重新运行这个程序，体验一下。

6.2. 使用验证码

Captcha是Java Completely Automated Public Test to tell Computers and Humans Apart的缩写。Captcha为了每一次请求生成唯一的验证码，除了有效的防止数据重复提交外，还能够一些跨站的攻击。

JCAPTCHA 提供了 Java 实现。请从 JCAPTCHA 网站 下载最新的稳定版本 1.0。

解压后，将 `jcaptcha-all.jar` 复制到你的项目的lib中。另外还需要一份 Commons Collections，它是 Apache Commons 项目的一部分。解压后，将 `commons-collections.jar` 文件复制到项目的 lib 目录中。

> **警告**
> 注意Commons Collections 的 3.x 版本与 2.x 版本不兼容，请下载最新的 3.x 版本。

我们通过一个 `ActionBean` 显示验证码图片。

```
public class CaptchaActionBean extends BaseActionBean {

    private final static Log log = LogFactory.getLog(CaptchaActionBean.class);

    @DontValidate
    @DefaultHandler
```

```
    @HttpCache(allow=false)
    public Resolution display() {

        byte[] captchaChallengeAsJpeg = null;
        ByteArrayOutputStream jpegOutputStream = new ByteArrayOutputStream();
        try {
            String captchaId = getContext().getRequest().getSession().getId();
            if (log.isDebugEnabled()) {
                log.debug("=================" + captchaId + "==================");
            }
            BufferedImage challenge = CaptchaManager.get().getImageChallengeForID(captchaId, getContext
            JPEGImageEncoder jpegEncoder = JPEGCodec.createJPEGEncoder(jpegOutputStream);
            jpegEncoder.encode(challenge);
        } catch (IllegalArgumentException e) {
            e.printStackTrace();
        } catch (CaptchaServiceException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        captchaChallengeAsJpeg = jpegOutputStream.toByteArray();
        return new StreamingResolution("image/jpeg",
                new ByteArrayInputStream(captchaChallengeAsJpeg));
    }
}
```

CaptchaManager 提供ImageCaptchaService 服务接口。

```
public class CaptchaManager {

    private static ImageCaptchaService imageCaptchaService = new DefaultManageableImageCaptchaService()

    public static ImageCaptchaService get() {
        return imageCaptchaService;
    }
}
```

应用到 JSP 页面，用一个 HTML 标签显示验证码图片。

```
<div>Verify Code</div>
<div>
     <stripes:text name="captchaResponse" size="10"/>
     <img src="${pageContext.servletContext.contextPath}/Captcha.action"
       alt="jCaptcha image"/>
     <stripes:errors field="register.captcha.error"/>
</div>
```

在 RegisterActionBean 中验证是否一致，仍然使用 jCaptcha 提供的方法。

```
@ValidationMethod(on = "register")
    public void verifyCode(ValidationErrors errors) {
        String captchaId = getContext().getRequest().getSession().getId();
        boolean isResponseCorrect = CaptchaManager.get().validateResponseForID(captchaId, captchaRespon
        if (!isResponseCorrect) {
            errors.add("register.captcha.error",
                    new SimpleError("Captcha code missmatch!"));
        }
```

```
}
```

运行这个程序进行测试。

第 7 章 页面显示

多页面分页显示，表单分步处理

本章讨论 web 最常见的页面显示问题，如分页显示，表单分步提交等。

7.1. 多记录分页显示

一些框架提供了专有分页显示方法，如 Apache Wicket，Apache Tapestry5。Stripes 没有提供相应的组件，来处理分页显示问题。很多开源的第三方框架提供了解决方案，比较著名的有 DisplayTag，JMesa 。我在过去的项目用得最多的就是 Display Tag。

7.1.1. 使用 Display Tag 处理分页

首先，你需要从 Display Tag 下载最新的 Display Tag，Display Tag 依赖下面的文件。

- commons-logging
- commons-lang
- commons-collections
- commons-beanutils
- log4j
- itext(可选，用于导出 PDF 和 RTF)

你可以从 Apache Commons 网站下载相应的包，并将相应的 jar 文件加入到项目的依赖库中。另外，Excel 导出需要添加displaytag-export-poi jar文件，它依赖pio jar，它是Apache Jakarta 下的一个子项目。这里不演示导出 PDF 和 Excel 文件。

创建一个 Users 类，提供分页显示中的数据源。这里 Users是一个伪数据库访问类，后面会替换成真实的数据库环境。

```
package tutorial.dao;

import tutorial.model.User;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 *
 * @author hantsy
 */
public class Users {

    static List<User> users = new ArrayList<User>();


    static {
        users.add(new User(1L, "Harry", "Potter"));
        users.add(new User(2L, "Bill", "Gates"));
        users.add(new User(3L, "Steven", "Jobs"));
        users.add(new User(4L, "Bob", "Lee"));
```

```
        users.add(new User(5L, "test", "test"));
        users.add(new User(6L, "test2", "test2"));
        users.add(new User(7L, "test3", "test3"));
        users.add(new User(8L, "test4", "test4"));
        users.add(new User(9L, "test5", "test5"));
        users.add(new User(10L, "test6", "test6"));
        users.add(new User(11L, "test7", "test7"));
        users.add(new User(12L, "test8", "test8"));
    }

    public static List<User> getUserList() {
        return users;
    }

    public static void deleteUser(Long id) {
        for (Iterator iterator =users.iterator(); iterator.hasNext();) {
            if (((User) iterator.next()).getId().longValue() == id) {
                iterator.remove();
            }
        }
    }

    public static List<User> subList(int start, int len) {
        if (start + len > users.size()) {
            return users.subList(start, users.size());
        }
        return users.subList(start, start + len);
    }
}
```

创建一个 `ActionBean` 显示用户列表。

```
package tutorial.action;

import java.util.List;
import net.sourceforge.stripes.action.DefaultHandler;
import net.sourceforge.stripes.action.ForwardResolution;
import net.sourceforge.stripes.action.Resolution;
import tutorial.dao.Users;
import tutorial.model.User;

/**
 *
 * @author hantsy
 */
public class ListUserActionBean extends BaseActionBean {

    private List<User> users;

    public List<User> getUsers() {
        return users;
    }

    public void setUsers(List<User> users) {
        this.users = users;
    }


    @DefaultHandler
    public Resolution listUsers(){
        users=Users.getUserList();
        return new ForwardResolution("/userList.jsp");
    }
```

```
}
```

创建 JSP 显示页面。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib uri="http://displaytag.sf.net" prefix="display" %>
<%@taglib uri="http://stripes.sourceforge.net/stripes.tld" prefix="stripes" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>User List Page</title>
    </head>
    <body>
        <h1> User List Page!</h1>
        <display:table id="row" defaultsort="1" pagesize="5" sort="list" name="actionBean.users" reques
            <display:column property="id" title="ID" />
            <display:column property="firstname" sortable="true" sortName="firstname" title="First Name
            <display:column property="lastname" sortable="true" sortName="lastname" title="Last Name" /
            <display:column title="Action">
                <stripes:link beanclass="tutorial.action.EditUserActionBean" event="edit">
            Edit
            <stripes:param name="id" value="${row.id}"/>
        </stripes:link> 
        <stripes:link beanclass="tutorial.action.DeleteUserActionBean" event="delete">
            Delete
            <stripes:param name="id" value="${row.id}"/>
        </stripes:link>
            </display:column>
        </display:table>
    </body>
</html>
```

创建一个简单的 `DeleteUserActionBean`，用于删除用户。

```
package tutorial.action;

import net.sourceforge.stripes.action.RedirectResolution;
import net.sourceforge.stripes.action.Resolution;
import tutorial.dao.Users;

/**
 *
 * @author hantsy
 */
public class DeleteUserActionBean extends BaseActionBean {

    private Long id;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }
```

```
    public Resolution delete() {
        Users.deleteUser(id);
        return new RedirectResolution(ListUserActionBean.class);
    }
}
```

创建一个简单的 `EditUserActionBean`。

```
package tutorial.action;

import net.sourceforge.stripes.action.ForwardResolution;
import net.sourceforge.stripes.action.Resolution;

/**
 *
 * @author hantsy
 */
public class EditUserActionBean extends BaseActionBean{

    public Resolution edit(){
        return  new ForwardResolution("/editUser.jsp");
    }
}
```

创建编辑页面 `editUser.jsp`。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Edit User!</h1>
    </body>
</html>
```

现在你已经可以体验分页程序。但是，我们每次都是查询了全部用户，当数据量很大时，存在性能问题。Display Tag 1.1 起也提供了动态查询，每次只需要取出当前页显示的记录即可。下面对这个程序加以改造。

Display Tag 提供了两种方法，一种是实现 `org.displaytag.pagination.PaginatedList`。另外一种方法，在Display Tag 标签中设置必要的属性参数。这种使用前者实现，修改 JSP 页面。

```
<display:table id="row" defaultsort="1" name="actionBean.users" pagesize="5" sort="external" size="acti
            <display:column property="id" title="ID" />
            <display:column property="firstname" sortable="true" sortName="firstname" title="First Name
            <display:column property="lastname" sortable="true" sortName="lastname" title="Last Name" /
            <display:column title="Action">
                <stripes:link beanclass="tutorial.action.EditUserActionBean" event="edit">
            Edit
            <stripes:param name="id" value="${row.id}"/>
        </stripes:link> 
```

```
        <stripes:link beanclass="tutorial.action.DeleteUserActionBean" event="delete">
            Delete
            <stripes:param name="id" value="${row.id}"/>
        </stripes:link>
            </display:column>
        </display:table>
```

修改ListUserActionBean。

```
public class ListUserActionBean extends BaseActionBean {

    private ResultList users;
    private String sort = "id";
    private String dir = "asc";
    private int page = 1;

    public String getDir() {
        return dir;
    }

    public void setDir(String dir) {
        this.dir = dir;
    }

    public int getPage() {
        return page;
    }

    public void setPage(int page) {
        this.page = page;
    }

    public String getSort() {
        return sort;
    }

    public void setSort(String sort) {
        this.sort = sort;
    }

    public ResultList getUsers() {
        return users;
    }

    public void setUsers(ResultList users) {
        this.users = users;
    }

    @DefaultHandler
    public Resolution listUsers() {
        users = new ResultList();
        return new ForwardResolution("/userList.jsp");
    }

    class FirstnameComparator implements Comparator<User> {

        public int compare(User o1, User o2) {
            return o1.getFirstname().compareTo(o2.getFirstname());
        }
    }

    class LastnameComparator implements Comparator<User> {

        public int compare(User o1, User o2) {
```

```
            return o1.getLastname().compareTo(o2.getLastname());
        }
    }

    class ResultList implements PaginatedList {

        public List getList() {
            List<User> list = Users.subList((getPageNumber() - 1) * getObjectsPerPage(), getObjectsPerP

            if (getSortCriterion().equals("firstname")) {
                Collections.sort(list, new FirstnameComparator());
            } else if(getSortCriterion().equals("lastname")){
                Collections.sort(list, new LastnameComparator());
            }
            if (getSortDirection().equals(SortOrderEnum.DESCENDING)) {
                Collections.reverse(list);
            }

            return list;
        }

        public int getPageNumber() {
            return page;
        }

        public int getObjectsPerPage() {
            return 5;
        }

        public int getFullListSize() {
            return Users.getUserList().size();
        }

        public String getSortCriterion() {
            return sort;
        }

        public SortOrderEnum getSortDirection() {
            if (dir.equals("desc")) {
                return SortOrderEnum.DESCENDING;
            }
            return SortOrderEnum.ASCENDING;
        }

        public String getSearchId() {
            return null;
        }
    }
}
```

这段程序中模拟了数据库查询。

7.1.2. 使用 JMesa 进行分页处理

JMesa 提供了更多的特性，如结果集过滤等。它也提供了类似的 Display Tag 类似的 JSP 页面 taglib。

从 JMesa 官方网站 下载，解压将 jmesa.jar 复制到项目的 lib 中。 另外 JMesa 依赖一些第三方的库。

- commons-lang
- commons-collections

- commons-beanutils
- slf4j

前面三个都可以从 Apache Commons下载。slf4j可以从 slf4j官方网站 下载。

修改`ListUserActionBean`。

```
public class ListUserActionBean extends BaseActionBean {

    private List<User> users;
    private String html;
    private String id = "users_table";

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getHtml() {
        return html;
    }

    public void setHtml(String html) {
        this.html = html;
    }

    public List<User> getUsers() {
        return users;
    }

    public void setUsers(List<User> users) {
        this.users = users;
    }

    @DefaultHandler
    public Resolution listUsers() {

        TableFacade tableFacade = TableFacadeFactory.createTableFacade(id, getContext().getRequest());
        tableFacade.setColumnProperties("id", "firstname", "lastname", "action");

        tableFacade.setMaxRows(5);
        tableFacade.setMaxRowsIncrements(5, 10);
        tableFacade.autoFilterAndSort(true);
        tableFacade.setStateAttr("state");

        Limit limit = tableFacade.getLimit();
        if (!limit.isComplete()) {
            int totalRows = Users.getUserList().size();
            tableFacade.setTotalRows(totalRows);
        }
        int rowEnd = limit.getRowSelect().getRowEnd();
        int rowStart = limit.getRowSelect().getRowStart();
        users = Users.subList(rowStart, rowEnd - rowStart);

        final SortSet sortSet = limit.getSortSet();
        if (sortSet != null) {
            Sort firstnameSort = sortSet.getSort("firstname");
            if (firstnameSort != null && firstnameSort.getOrder() != Order.NONE) {
                Collections.sort(users, new FirstnameComparator());
                if (firstnameSort.getOrder() == Order.DESC) {
```

```
                Collections.reverse(users);
            }
        }

        Sort lastnameSort = sortSet.getSort("lastname");
        if (lastnameSort != null && lastnameSort.getOrder() != Order.NONE) {
            Collections.sort(users, new LastnameComparator());
            if (lastnameSort.getOrder() == Order.DESC) {
                Collections.reverse(users);
            }
        }
    }



    // users = Users.subList(rowStart, rowEnd - rowStart);
    tableFacade.setItems(users);

    HtmlTable table = (HtmlTable) tableFacade.getTable();
    //table.setCaption("Presidents");
    table.getTableRenderer().setWidth("600px");

    HtmlRow row = table.getRow();

    HtmlColumn userid = row.getColumn("id");
    userid.setTitle("User ID");
    userid.setSortable(false);

    HtmlColumn firstName = row.getColumn("firstname");
    firstName.setTitle("First Name");


    HtmlColumn lastName = row.getColumn("lastname");
    lastName.setTitle("Last Name");


    HtmlColumn action = row.getColumn("action");
    action.setTitle("Action");
    action.setFilterable(false);
    action.setSortable(false);

    action.getCellRenderer().setCellEditor(new CellEditor() {

        public Object getValue(Object item, String property, int rowcount) {
            Object value = ItemUtils.getItemValue(item, "id");
            HtmlBuilder html = new HtmlBuilder();
            html.a().href().quote().append(getContext().getServletContext().getContextPath()).apper
            html.append("Edit");
            html.aEnd();
            html.append(" ");
            html.a().href().quote().append(getContext().getServletContext().getContextPath()).apper
            html.append("Delete");
            html.aEnd();
            return html.toString();
        }
    });

    html = tableFacade.render();
    return new ForwardResolution("/userList.jsp");
}

class FirstnameComparator implements Comparator<User> {

    public int compare(User o1, User o2) {
        return o1.getFirstname().compareTo(o2.getFirstname());
    }
```

```
    }

    class LastnameComparator implements Comparator<User> {

        public int compare(User o1, User o2) {
            return o1.getLastname().compareTo(o2.getLastname());
        }
    }
}
```

修改 userList.jsp 页面，将 body 中间内容替换成如下。

```
   ...
<stripes:form beanclass="tutorial.action.ListUserActionBean" method="post">
        ${actionBean.html}
        </stripes:form>
        <script type="text/javascript">
            function onInvokeAction(id) {
                createHiddenInputFieldsForLimitAndSubmit(id);
            }
        </script>
        ...
```

JMesa 依赖 JQuery，对于熟悉的 JQuery 的用户来说，这无疑是个好消息。

从 JMesa 的例子程序复制 `css`，`js`，`images` 目录到项目的 `web` 目录中。

现在你可以运行程序了。

> **注意**
> 这里我们没有使用 JMesa 的taglib，你可以尝试使用 taglib 来替代这种编程的方式。

7.2. 分步提交

Struts 1 内置了多步提交表单的方案，在 ActionForm 添加一个 page 属性，指定当前是第几步，将 ActionForm 放入 Session 中，或者在下一步表单中使用隐藏属性存放其上一步的表单数据，保证多步表单数据不丢失。Stripes 也提供了一种有效的方式，来处理多步提交。

回到注册程序，在填写注册信息之前，要求用户先阅读相关协议。

创建一个阅读协议 JSP 页面。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Read User License!</h1>
        <stripes:errors/>
        <div style="height:300pt;width:500pt">
```

```
            License,License,License,License,License,License,
            License,License,License,License,License,License,
            License,License,License,License,License,License,
            License,License,License,License,License,License,
            License,License,License,License,License,License,
        </div>
        <stripes:form beanclass="tutorial.action.RegisterActionBean">
            <stripes:checkbox name="acceptLicense"/> I have read the license and gree with the content.
            <stripes:submit name="prepareRegister" value="I Agree"/>
        </stripes:form>
    </body>
</html>
```

在RegisterActionBean中添加一个属性 acceptLicense ， 类型为 boolean ， 并添加相应的 event 方法。

```
@Wizard(startEvents = "readLicense")
public class RegisterActionBean extends BaseActionBean {

    @Validate(required = true, mask = "[0-9a-zA-Z]{6,20}")
    private String username;
    @Validate(required = true, minlength = 6, maxlength = 20, mask = "[0-9a-zA-Z]+")
    private String password;
    @Validate(required = true, converter = EmailTypeConverter.class)
    private String email;
    @Validate(required = true, expression = "this eq password")
    private String confirmPassword;
    @Validate(converter = DateTypeConverter.class)
    Date birthDate;
    @Validate(converter = BooleanTypeConverter.class)
    boolean subscriptionEnabled;
    @ValidateNestedProperties({
        @Validate(field = "zipcode", required = true),
        @Validate(field = "addressLine1", required = true),
        @Validate(field = "addressLine2", required = true)
    })
    private Address address;


    private boolean acceptLicense = false;

    public boolean isAcceptLicense() {
        return acceptLicense;
    }

    public void setAcceptLicense(boolean acceptLicense) {
        this.acceptLicense = acceptLicense;
    }

    public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }

    public Date getBirthDate() {
        return birthDate;
    }

    public void setBirthDate(Date birthDate) {
        this.birthDate = birthDate;
```

```
    }

    public boolean isSubscriptionEnabled() {
        return subscriptionEnabled;
    }

    public void setSubscriptionEnabled(boolean subscriptionEnabled) {
        this.subscriptionEnabled = subscriptionEnabled;
    }

    public String getConfirmPassword() {
        return confirmPassword;
    }

    public void setConfirmPassword(String confirmPassword) {
        this.confirmPassword = confirmPassword;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    @DefaultHandler
    public Resolution register() {
        return new ForwardResolution("/success.jsp");
    }

    @ValidationMethod(on = "register")
    public void userExsited(ValidationErrors errors) {
        if ("testuser".equals(username)) {
            errors.add("username", new SimpleError("This username is taken , please select a different
        }
        if (!errors.isEmpty()) {
            errors.addGlobalError(new SimpleError("Error ocurs on save. Please fix it firstly."));
        }
    }

    public Resolution readLicense() {
        return new ForwardResolution("/readlicense.jsp");
    }

    @DontValidate
    public Resolution back() {
        return new ForwardResolution("/readlicense.jsp");
    }
```

```
    @ValidationMethod(on = "prepareRegister")
    public void mustAcceptLicense(ValidationErrors errors) {
        if (this.acceptLicense == false) {
            errors.add("acceptLicense", new SimpleError("You must agree with the content of the license
        }
        if (!errors.isEmpty()) {
            errors.addGlobalError(new SimpleError("Error ocurs on save. Please fix it firstly."));
        }
    }

    public Resolution prepareRegister() {
        return new ForwardResolution("/register.jsp");
    }
}
```

Stripes 提供了 `Wizard` Annotation 保证多个页面的表单处理起来像在个页面一样。

在 `register.jsp` 页面添加一个回退到第一步按钮。

```
<div>
     <stripes:submit name="back" value="Last Step"/>
     <stripes:submit name="register" value="Register Now"/>
</div>
```

现在运行这个程序来体验一下。

第 8 章 文件上传

文件上传

本章讨论 web 中常见的文件上传问题。

电子相册，论坛注册添加一个个性化图标，等等，这些都是很常见的 web 应用，都需要文件上传功能把本地的文件上传到远程的服务器。

与普通的 Form 不同的是，文件上传的表单 form 标签必须添加一个 `enctype="multipart/form-data"` 属性，当使用标签 stripes:form 时它会自动添加这一属性。当表单提交后，表单数据以字节流的方式传递到远程服务器。如果自己分析上传表单内容，是件麻烦事。

cos 和 commons-fileupload 是两种主流的上传工具，内置了表单分析方法。 Stripes 对他们进行了包装，不需要了解两种工具的上传操作的细节。提供了统一的接口，从一种实现切换到另一种实现，不需要修改任何代码。

基于程序的向前的兼容性考虑，Stripes 自带了 cos ，你可以通过简单的配置决定使用哪一种后端实现。如果你想使用 commons-fileupload 后端来处理文件上传，在 Stripes Filter 上添加一个初始化参数。

```
<init-param>
    <param-name>MultipartWrapper.Class</param-name>
    <param-value>net.sourceforge.stripes.controller.multipart.CommonsMultipartWrapper</param-value>
</init-param>
```

你可以从Apache Commons 上下载最新的 Commons FileUpload 。同时它还依赖其它 Commons包，你至少要添加一个 Commons IO。

你可以通过另外一个参数 `FileUpload.MaximumPostSize` 来控制上传文件的大小，但必须注意的是这一参数控制的是整个上传表单的数据大小，而不是文件的大小。一般情况下，其它输入字段的体积相对较小，如果你允许上传体积为 1mb 以上的文件，这些体积几乎可以忽略不计。

> **警告**
> `FileUpload.MaximumPostSize` 限制的整个表单上传数据的大小。

8.1. 单个文件上传

创建JSP 文件。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
   <%@taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Upload File Page</title>
    </head>
    <body>
        <h1>Upload File</h1>
        <div><stripes:errors/></div>
```

```
        <stripes:form beanclass="tutorial.action.UploadActionBean" method="post">
            <div><stripes:file name="uploadFile"/></div>
            <div><stripes:submit name="upload" value="Upload Now"/></div>
        </stripes:form>
        <div>
            <stripes:link beanclass="tutorial.action.MultiUploadActionBean">
                Upload more than one files at the same time.
            </stripes:link>
        </div>
    </body>
</html>
```

创建 `ActionBean` 处理文件上传。

```
public class UploadActionBean extends BaseActionBean {

    private final static Log log = LogFactory.getLog(UploadActionBean.class);
    private FileBean uploadFile;
    private long uploadFileSize;

    public long getUploadFileSize() {
        return uploadFileSize;
    }

    public void setUploadFileSize(long uploadFileSize) {
        this.uploadFileSize = uploadFileSize;
    }

    public FileBean getUploadFile() {
        return uploadFile;
    }

    public void setUploadFile(FileBean uploadFile) {
        this.uploadFile = uploadFile;
    }

    public Resolution upload() {
        log.debug("Upload File :" + uploadFile.getFileName());
        log.debug("File size:" + uploadFile.getSize());
        log.debug("File Content type:" + uploadFile.getContentType());
        this.uploadFileSize = uploadFile.getSize();
        String rootPath = getContext().getServletContext().getRealPath("/");
        try {
            uploadFile.save(new File(rootPath + "/public/" + uploadFile.getFileName()));
        } catch (IOException ex) {
            log.debug("Upload file exception, root cause @" + ex);
        }
        return new ForwardResolution("/success.jsp");
    }
}
```

上传结果页面。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
```

```
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Upload Result Page</title>
    </head>
    <body>
        <h1>Uploaded Successfully!</h1>
        <div>File Name is: ${actionBean.uploadFile.fileName}. </div>
        <div>File Size is: ${actionBean.uploadFileSize}</div>
    </body>
</html>
```

运行程序，上传一个文件进行测试。

8.2. 多文件上传

创建上传文件页面。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Upload File Page</title>
    </head>
    <body>
        <h1>Upload File!</h1>
        <div><stripes:errors/></div>
        <c:set var="list" value="<%= new Object[5] %>" scope="page"/>
        <stripes:form beanclass="tutorial.action.MultiUploadActionBean" method="post">
            <c:forEach items="${list}" varStatus="loop">
                <div>File ${loop.index+1}: <stripes:file name="uploadFiles[${loop.index}]"/></div>
            </c:forEach>
            <div><stripes:submit name="upload" value="Upload Now"/></div>
        </stripes:form>
    </body>
</html>
```

创建 `ActionBean`，处理文件上传，一个List 包装上传的文件。

```
public class MultiUploadActionBean extends BaseActionBean {

    private final static Log log = LogFactory.getLog(MultiUploadActionBean.class);
    private List<FileBean> uploadFiles =new ArrayList<FileBean>();

    public List<FileBean> getUploadFiles() {
        return uploadFiles;
    }

    public void setUploadFiles(List<FileBean> uploadFiles) {
        this.uploadFiles = uploadFiles;
    }

    @DefaultHandler
    public Resolution preUpload() {
        return new ForwardResolution("/upload2.jsp");
```

```
    }

    public Resolution upload() {

        for (FileBean uploadFile : uploadFiles) {
            if (uploadFile != null) {
                log.debug("Upload File :" + uploadFile.getFileName());
                log.debug("File size:" + uploadFile.getSize());
                String rootPath = getContext().getServletContext().getRealPath("/");
                try {
                    uploadFile.save(new File(rootPath + "/public/" + uploadFile.getFileName()));
                } catch (IOException ex) {
                    log.debug("Upload file exception, root cause @" + ex);
                }
            }
        }

        return new ForwardResolution("/success2.jsp");
    }
}
```

创建结果页面。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Upload Result Page</title>
    </head>
    <body>
        <h1>Uploaded Successfully!</h1>
        <p>File Name is:</p>
        <c:forEach items="${actionBean.uploadFiles}" var="fileVar" varStatus="loop">
            <c:if test="${fileVar !=null}">
                <div>${fileVar.fileName}</div>
            </c:if>
        </c:forEach>
    </body>
</html>
```

第 9 章 文件下载

文件下载

在前面的章节中已经认识了 Resolution 接口，这一章将使用StreamingResolution 实现文件下载。

创建一个 ActionBean。

```
public class DownloadActionBean extends BaseActionBean {

    @Validate(required=true)
    private String filename;

    public String getFilename() {
        return filename;
    }

    public void setFilename(String filename) {
        this.filename = filename;
    }

    public Resolution download() {
        return new StreamingResolution("application/octet-stream", buildDowloadSource()).setFilename(fi
    }

    public InputStream buildDowloadSource() {
        String rootPath = getContext().getServletContext().getRealPath("/");
        String filePath = rootPath + "/public/Fedora.png";
        try {
            return new FileInputStream(filePath);
        } catch (FileNotFoundException ex) {
            getContext().getValidationErrors().addGlobalError(new SimpleError("Can not found file!"));
        }
        return null;
    }
}
```

当 StreamingResolution 设置了filename 属性时，会自动弹出下载文件确认框。

创建下载页面，提供下载链接。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
   <%@taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Download File Page</title>
    </head>
    <body>
        <h1>Download File!</h1>
        <div><stripes:errors/></div>
        <div>Fedora BackGround:</div>
        <stripes:link beanclass="tutorial.action.DownloadActionBean">
            <img src="${pageContext.servletContext.contextPath}/public/Fedora.png" style="border:0; wid
            <stripes:param name="filename" value="fedora-bg.png"/>
        </stripes:link>
    </body>
```

```
</html>
```

这里将一张图片放入项目程序的根目录下的 `/public`下。运行程序进行测试。

第 10 章 页面布局

如何让页面最大限度的进行复用

一个web 项目常常包含很多页面，各页面常常只是内容不同。如果每个页面都是独立的，显然维护起来非常麻烦。JSP 提供了 `<jsp:include>` 可以将在 JSP 文件中包括其它 JSP 文件片断。但是每个特定的 JSP 页面都需要用 `<jsp:include>` 来插入相应的模板，这也是件麻烦的事。

Struts 1 内置了 Tiles 支持，Tiles 能够有效的组织页面。现在 Tiles 已经 Apache 上一个一级项目，称为Tiles 2 ，相应的 Struts 1 中的 Tiles 就是 Tiles 1。

另外一种页面布局工具是 Sitemesh ，原理上与 Tiles 有很大差别，Sitemesh 对开发人员和设计人员更加友好，特定页面依然可以使用完整的 HTML 标签，即可以包含 `head`，`body`等标签，运行时才与模板文件进行重新组合。

Stripes 内置了一套简单的方案，与 Tiles 有些类似，但要简单很多，它只含三个相应的标签。

- <stripes:layout-definition> 定义了一套可复用的页面模板。
- <stripes:layout-component> 定义了一个模板页面的组件
- <stripes:layout-render> 按模板生成页面

下面我们通过示例来说明使用。

10.1. 示例

这里重新改造最初的 helloworld 程序。首先创建模板文件，新建/layouts 目录，创建一个 JSP 文件，名为 `default.jsp`。这里假设把一个页面分为三部分，header，content，和 footer，实际应用中应该复杂得多。一般经常变化的就是内容，即content。

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
<stripes:layout-definition>
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
            <title>Stripes Layout Examples </title>
        </head>
        <body>
            <stripes:layout-component name="header">
                <jsp:include page="/layouts/header.jsp"/>
            </stripes:layout-component>

            <stripes:layout-component name="content"/>

            <stripes:layout-component name="header">
                <jsp:include page="/layouts/footer.jsp"/>
            </stripes:layout-component>
        </body>
    </html>
</stripes:layout-definition>
```

我们将header，footer 部分独立成一个单独文件，便于管理。这个layout模板定义了三个组件，即header, content, footer。

`/layouts/header.jsp`，`/layouts/footer.jsp`是普通的 JSP 文件片断。

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

    <h1>Say Hello to Stripes! </h1>
```

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<hr noshade="true"/>
<div>Powered by <a href="http://www.stripesframework.org">Stripes</a>.</div>
```

layout模板页面已经定义好了，现在看看如何应用。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
<stripes:layout-render name="/layouts/default.jsp">
    <stripes:layout-component name="content">
        <stripes:form beanclass="tutorial.action.HelloActionBean" method="post">
            <div><stripes:text name="message"/></div>
            <div><stripes:submit name="sayHello" value="Say Hello"/></div>
        </stripes:form>
    </stripes:layout-component>
</stripes:layout-render>
```

运行程序进行测试。

10.2. 向layout模板文件传递参数

现在所的页面标题都是一样的，所有的窗口标题也是不变的。如何让标题动态的显示呢？

修改 layout 文件。

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
<stripes:layout-definition>

    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
            <title>Stripes Layout Examples:${pageTitle} </title>
        </head>
        <body>
            <stripes:layout-component name="header">
                <jsp:include page="/layouts/header.jsp"/>
            </stripes:layout-component>
            <div>${pageTitle}</div>
```

```
            <stripes:layout-component name="content"/>

            <stripes:layout-component name="header">
                <jsp:include page="/layouts/footer.jsp"/>
            </stripes:layout-component>
        </body>
    </html>
</stripes:layout-definition>
```

这里打算从外部传一个 pageTitle 进来，用 EL 表示。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
<stripes:layout-render name="/layouts/default.jsp" pageTitle="Hello Stripes">
    <stripes:layout-component name="content">
        <stripes:form beanclass="tutorial.action.HelloActionBean" method="post">
            <div><stripes:text name="message"/></div>
            <div><stripes:submit name="sayHello" value="Say Hello"/></div>
        </stripes:form>
    </stripes:layout-component>
</stripes:layout-render>
```

pageTitle 作为 `<stripes:layout-render>` 的一个属性。

另外参数也可以定义为 layout-component， 其属性 `name`为要传递的参数名。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<stripes:layout-render name="/layouts/default.jsp">
    <jsp:useBean id="user" scope="page" class="tutorial.action.UserSession"/>
    <stripes:layout-component name="pageTitle">
        <c:if test="${user.logined}">
            Welcome back, ${user.username}
        </c:if>
        <c:if test="${!user.logined}">
            Welcome, Guest!
        </c:if>
    </stripes:layout-component>
    <stripes:layout-component name="content">
        <stripes:form beanclass="tutorial.action.HelloActionBean" method="post">
            <div><stripes:text name="message"/></div>
            <div><stripes:submit name="sayHello" value="Say Hello"/></div>
        </stripes:form>
    </stripes:layout-component>
</stripes:layout-render>
```

创建一个 `UserSession`类，它提供用户检测功能，但只是一个伪类，这里仅仅只是为了演示。

10.3. 嵌套使用

这三个标签的使用非常灵活。

```
<stripes:layout-component name="header">
                <stripes:layout-render name="/layouts/footer.jsp"/>
```

```
            </stripes:layout-component>
```

footer.jsp 现在是一个 layout-definition。

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
<stripes:layout-definition>
<hr noshade="true"/>
<div>Powered by <a href="http://www.stripesframework.org">Stripes</a>.</div>
</stripes:layout-definition>
```

运行测试项目。

第 11 章 国际化和本地化

在全球化的进程，各种语种的人进行交流越来越频繁，web 程序的国际化也是必不可少的，对于每个人来讲，最好的方式就是提供自己最熟悉的语言环境。

国际化常常被称为 i18n ，因为英文单词 internationalization 的开始字母 i 和结尾字母 n 之间有18 个字符。国际化的过程就是将自己的语言环境转换成其它语言环境的过程。本地化所做的工作则恰恰相反，它是将其它语言环境转换成本地语言环境。

web 程序中的国际化和本地化牵涉到一些最基本的工作，不仅仅是字面上的翻译，还应该考虑各种语言中日期格式，时区，货币格式的转换等。

11.1. 取得当前Locale

Stripes 中使用 `LocalePicker` 类来处理请求中该使用哪个 Locale，`LocalePicker` 由`StripesFilter`来执行。Stripes 提供了一个初始化参数 `LocalePicker.Locales`，它可以指定一系列的Locale，用逗号隔开，如en_US，zh_CN等，其中包括了语言和国家地区信息，还有可能包含变体信息。

```
<init-param>
    <param-name>LocalePicker.Locales</param-name>
    <param-value>en_US,zh</param-value>
</init-param>
```

如果没有指定，Stripes 会取得系统默认的Locale，即通过Locale.getDefault()取得。

`LocalePicker.Locales`取得Locale信息后，Stripes 会使用`HttpServletRequestWrapper`来调用 `getLocale()`和`getLocales()` 返回唯一的选择的Locale。

Stripes 在每次请求时都会重新确定 Locale，这也保证了你更换 Locale 时能够及时的生效。

11.2. 选择字符编码

除了选择语言之外，必须正确选择字符编码，才能正确的显示页面。Stripes添加`StripesFilter`初始化参数 `LocalePicker.Locales`来设置编码。例如：

```
<init-param>
    <param-name>LocalePicker.Locales</param-name>
    <param-value>en_US:UTF-8,zh_CN:GBK</param-value>
</init-param>
```

Stripes可以为每种Locale指定一种编码，在Locale后面以冒号隔开，添加编码名称。运行时，Stripes会将`HttpServletRequest`中的所有的字符都按指定的编码进行转换，所有字符数据返回客户端也是使用相同的编码。如果没有指定，就会使用 `Servlet` 容器的默认的方式进行处理。在 Tomcat 中，字符编码默认使用 ISO8859 。

11.3. 查找资源信息

默认情况下，Stripes 提供唯一的 StripesResources.properties 文件，它是以key-value 形式存在的文本文件，用于错误信息和表单字段的信息显示。在多语言环境，每种语言都应该有一个相应的版本。如，StripesResources_zh_CN.properties，StripesResources_zh_TW.properties。当针对某一个 Locale 时，Stripes 根据一定的算法查找最适合的信息。如zh_CN，Stripes 首先会根据 key 值在StripesResources_zh_CN.properties中进行查找，如果找不到会查找 StripesResources_zh.properties(当然前提是 StripesResources_zh.properties要存在)，最后会查找默认文件 StripesResources.properties。

Stripes 提供了两个 StripesFilter 初始化参数 `LocalizationBundleFactory.ErrorMessageBundle` 和 `LocalizationBundleFactory.FieldNameBundle`，可以将资源文件分开来存放。

11.4. 示例

修改web.xml，在 StripesFilter 中添加一个初始化参数，指定Locale。

```
<init-param>
            <param-name>LocalePicker.Locales</param-name>
            <param-value>zh_CN:UTF-8,en_US</param-value>
        </init-param>
```

复制一份StripesResources.properties文件，另存为StripesResources_zh_CN.properties。以register程序为，添加表单相应的键值对。

```
username=Username
password=Password
confirmPassword=Confirm Password
email=Email
birthDate=Birth Date
address.zipcode=Zip Code
address.addressLine1=Address Line 1
address.addressLine2=Address Line 1
username.isTaken={0} is taken by another user.
errors.must.fix=Errors occured, you must fix them and continue.
```

StripesResources_zh_CN.properties文件中添加相应的内容。

```
#stripes.dateTypeConverter.preProcessPattern=-
username=用户名
password=密码
confirmPassword=密码确认
email=电子邮件
birthDate=出生日期
address.zipcode=邮编
address.addressLine1=地址
address.addressLine2=城市
username.isTaken={0} 已经其他人被占用
errors.must.fix=当前页面出现错误，在进行下一步之前必须进行修复
```

> **警告**
> 你需要用 native2ascii 将一些非Unicode 字符转换成 Unicode 的ascii码。一些IDE，如NetBeans 提供了良好的可视编辑能力，你根本就不关心这个问题，它自动帮你完成转换。

修改register.jsp页面文件。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="stripes" uri="http://stripes.sourceforge.net/stripes.tld" %>
<%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Registeration Page</title>
    </head>
    <body>
        <h1>User Registeration</h1>
        <div><stripes:errors globalErrorsOnly="true"/></div>
        <div>
            <stripes:form beanclass="tutorial.action.RegisterActionBean">
                <div><stripes:label name="username"/>: </div><div> <stripes:text name="username"/><stri
               <div><stripes:label name="password"/>: </div><div><stripes:password name="password"/><st
               <div><stripes:label name="confirmPassword"/>: </div><div><stripes:password name="confirm
               <div><stripes:label name="email"/>: </div><div><stripes:text name="email"/><stripes:erro
               <div><stripes:label name="birthDate"/>: </div><div><stripes:text name="birthDate"/><stri
               <fieldset>
                   <legend>Address</legend>
                   <div><stripes:label name="address.zipcode"/>:</div>
                   <div><stripes:text name="address.zipcode"  size="10"/><stripes:errors field="address
                   <div><stripes:label name="address.addressLine1"/>: </div>
                   <div><stripes:text name="address.addressLine1"/><stripes:errors field="address.addre
                   <div><stripes:label name="address.addressLine2"/>: </div>
                   <div><stripes:text name="address.addressLine2"/><stripes:errors field="address.addre
              </fieldset>

               <div>Would like receive our product infomation by email? </div>
               <div><stripes:checkbox name="subscriptionEnabled" value="on"/>If you would like, check o
               <div><stripes:submit name="register"/> </div>
            </stripes:form>
        </div>
    </body>
</html>
```

你可以看到，label和input都可以通过name 的值来配置，但是当有很多页面时容易混淆。Stripes 支持以actionPath.fieldName方式进行查找。

```
/Register.action.username=Username
/Register.action.password=Password
/Register.action.confirmPassword=Confirm Password
```

对于自定义的错误信息，Stripes 提供了LocalizableError 和 ScopedLocalizableError 进行处理。前者需要提供key值和可选的参数，后者会绑定到某个scope上，如用户名被其他人占用，把它绑username上显示。

```
@ValidationMethod(on = "register")
    public void userExsited(ValidationErrors errors) {
```

```
        if ("testuser".equals(username)) {
            errors.add("username", new ScopedLocalizableError("username", "isTaken", "testuser"));
        }

    }

    public Resolution handleValidationErrors(ValidationErrors errors) throws Exception {
        if (!errors.isEmpty()) {
            errors.addGlobalError(new LocalizableError("errors.must.fix"));
        }
        return getContext().getSourcePageResolution();
    }
```

运行程序进行测试。

11.5. 与JSTL共处

JSTL 也提供了相应的国际化标签库，你可以让它使用 Stripes 的资源文件。

```
<context-param>
        <param-name>javax.servlet.jsp.jstl.fmt.localizationContext</param-name>
        <param-value>StripesResources</param-value>
</context-param>
```

第 12 章 Ajax 技术

Ajax 无疑是当今最引人注目的 web 技术。与传统的请求方式不同的是，Ajax 请求常常只是提交部分页面数据到服务器上，页面常常只是局部更新。Ajax 显然提高了用户检验，但同时带来的问题也明显的。由于请求频率上升了一个数量级，服务器负担明显加重。另外客户端浏览器的导航功能瘫痪，你无法使用"前进"和"倒退"功能。

本单会讨论 Ajax 一些常用 Ajax 技术结合 Stripes 框架的应用。关于Ajax 的细节使用，你可以参考这方面的专著。

12.1. 示例：即时检测账号的合法性

在前面的 register 程序中，可以提前检测用户名是否合法。如果用户名已经存在，即时进行提示。

这里打算使用`Prototype`的 Ajax 库。

在web 目录下新建一个 `js` 文件夹，下载最新的稳定版本 1.6.0.3，放入此文件夹中。

在 JSP 页面引入 `prototype.js` 文件。

```
        <script type="text/javascript" src="${pageContext.servletContext.contextPath}/js/prototype-1.6.
        </script>
```

在username 的input 标签中添加属性 id，另外设置错误显示区域。

```
<div>Username: </div><div> <stripes:text name="username" id="username" onchange="checkUserExist();"/><s
```

使用 Ajax.Request 方法处理 Ajax 请求。

```
        <script language="Javascript" type="text/javascript">
            function checkUserExist(){
                var username=$('username');
                $('username-error').innerHTML="";
                console.log("Username @"+username.value);
                console.log("username len#"+username.value.length);
                if(username.value.length>0){
                    new Ajax.Request('${pageContext.servletContext.contextPath}/Register.action?checkUs
                        method:'get',
                        onSuccess: function(transport){

                            var response = transport.responseText || "no response text";
                            console.log("Success! \n\n" + response);
                            if(response=='true'){
                                $('username-error').innerHTML="<font color='red'>用户名已经存在</font>";
                            }else{
                                $('username-error').innerHTML="<font color='green'>用户名可用</font>";
                            }
                        }
                    });
                }
            }
```

```
        </script>
```

在对应的 ActionBean 中添加对应的处理方法。

```
    @DontValidate
    @HttpCache(allow = false)
    public Resolution checkUser() {
        System.out.println("@@@checkUser method invoked...@@@");
        System.out.println("@@@ username value is:" + username);
        return new StreamingResolution("text/plain") {

            @Override
            protected void stream(HttpServletResponse response) throws Exception {
                if ("testuser".equals(username)) {
                    response.getWriter().write("true");
                } else {
                    response.getWriter().write("false");
                }
            }
        };
    }
```

运行程序进行测试。

12.2. 示例：重新获取验证码

一个 captcha 组件，可能初次提次的验证码不够清楚，无法辩认，你可能想刷新页面重新获得验证码。如果是一个表单，并且你已经输入信息，刷新整个页面的话就有可能丢失表单信息。很多 captcha 组件也提供 captcha 图片局部刷新的方式来获得新的验证码。

修改验证码页面片断。

```
<span id="verifycode-content">
                        <img src="${pageContext.servletContext.contextPath}/Captcha.action" alt="jCapto
                    </span>
                    <span><a href="javascript:refresh();">Refresh</a></span>
```

添加相应的js代码。

```
function refresh(){
                new Ajax.Updater('verifycode-content',
                '${pageContext.servletContext.contextPath}/Register.action?refresh',
                { method: 'get' }
            );
            }
```

ActionBean 类中的处理方法。

```
    @DontValidate
    @DontBind
    @HttpCache(allow = false)
    public Resolution refresh() {
```

```
        return new ForwardResolution("/captcha.jsp");
    }
```

这里使用一个JSP 页面片断来显示内容。

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<img src="${pageContext.servletContext.contextPath}/Captcha.action?<%=java.lang.Math.random()%>" alt="j
```

这里在链接结尾添加一个随机数，以便区别已经有图片名称。

第 13 章 单元测试

过去工作中，我常常听一些共事的人讲这样的话，“我的程序不可能有问题”，但结果恰恰总是事与愿违，这些人的自大往往导致了程序无法运行。事实证明，人无完人，人的思维不可能是尽善尽美的。对于我来讲，与不写的测试的人共事，是件相当头痛的事。测试能够有效提高软件的质量，将bug数量降低到最低。对于程序员，首先要做的是单元测试。

*Pragmatic Programmer, The: From Journeyman to Master*一书中 Tip 27为“Don't Assume It—Prove It”，道出了测试的真谛。

13.1. 使用 TestNG 进行测试

以 helloworld 为例，测试 HelloActionBean。

```
public class HelloActionBeanTest {

    @Test
    public void sayHello(){
        HelloActionBean hello=new HelloActionBean();
        hello.setContext(new  ActionBeanContext());
        hello.setMessage("Stripes!");
        hello.sayHello();
        Assert.assertEquals("Hello,Stripes!", hello.getMessage());

    }

}
```

因为 `ActionBean` 是一个POJO 类，测试内部逻辑并不困难，因为它不依赖容器类，如`HttpServletRequest`，Session，Cookie等。如果要测试这种类相关的逻辑，单纯依靠 TestNG 可能比较难实现。但是，Stripes 提供一套测试 API，能够在隔离环境中测试整个页面流程。

> **提示**
> TestNG 官方网站提供了 testng for eclipse。NetBeans 的nightly build 也支持 TestNG ，参见 http://wiki.netbeans.org/TestNG。

13.2. 使用 Stripes 测试 API

在`net.sourceforge.stripes.mock`包中提供多个Mock类。其中包括MockHttpServletRequest，MockSession等。有了这些类，你可以脱离容器的情况下，模拟 servlet 容器环境。

```
public class HelloIntegratedTest {

    MockServletContext context;

    @BeforeTest
    public void setupNonTrivialObjects() {
        context = new MockServletContext("testing");

        // Add the Stripes Filter
        Map<String, String> filterParams = new HashMap<String, String>();
        filterParams.put("ActionResolver.Packages", "tutorial.action");
        context.addFilter(StripesFilter.class, "StripesFilter", filterParams);
```

```
        // Add the Stripes Dispatcher
        context.setServlet(DispatcherServlet.class, "StripesDispatcher", null);
    }

    @Test
    public void sayHelloTest() throws Exception {
        // Setup the servlet engine
        MockRoundtrip trip = new MockRoundtrip(context, HelloActionBean.class);
        trip.setParameter("message", "Stripes!");
        trip.execute("sayHello");

        HelloActionBean bean = trip.getActionBean(HelloActionBean.class);
        Assert.assertEquals("Hello,Stripes!", bean.getMessage());
        Assert.assertEquals(trip.getDestination(), "/greeting.jsp");
    }
}
```