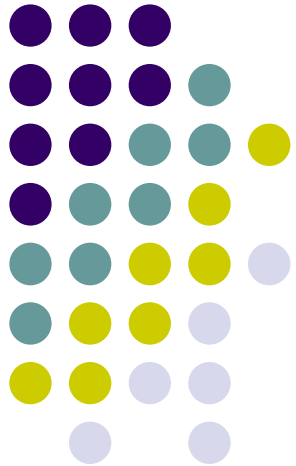
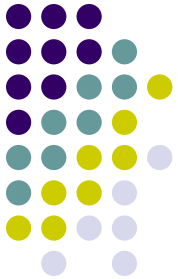
第十一章 Java EE开源框架介绍





主要内容

- 框架技术
- JavaEE开源框架
- 表示层框架Struts
- 业务逻辑层框架Spring
- 数据持久层框架Hibernate
- 基于SSH组合框架的Web应用

框架技术

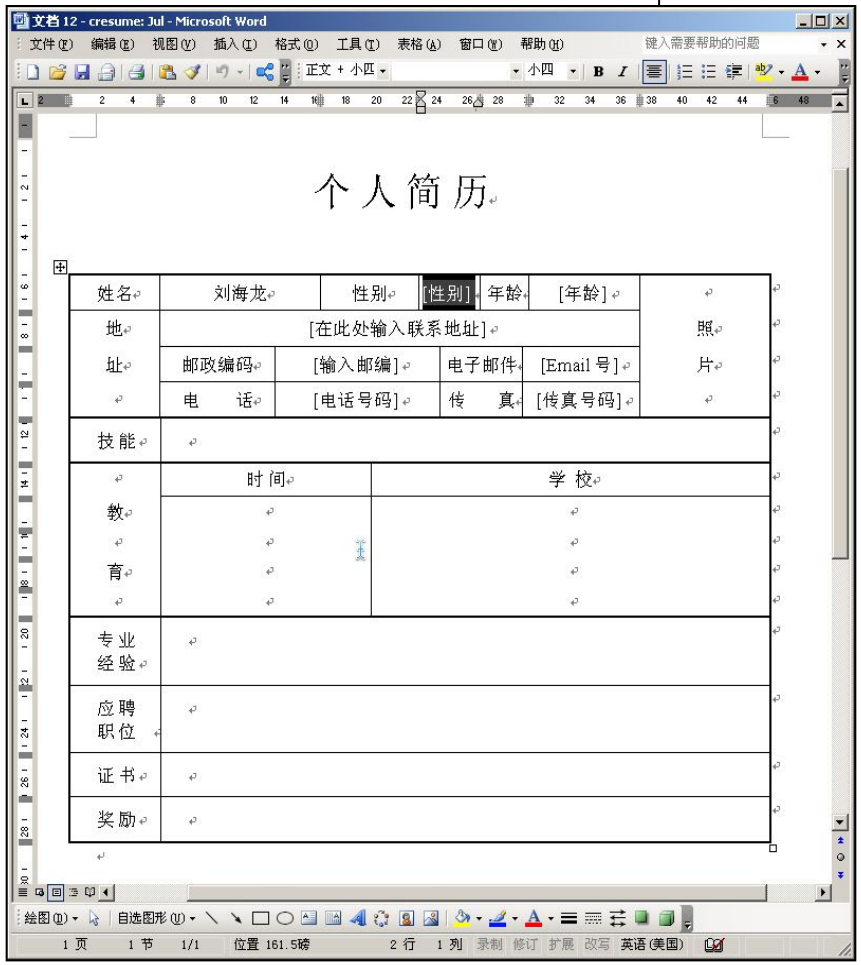
- 如何更快更好地写简历？
  - ⇗ 使用Word简历模板
- 思考：
  - ⇗ 使用模板有什么好处呢？

不用考虑布局、排版等，提高效率

可专心在简历内容上

结构统一，便于人事阅读

新手也可以作出专业的简历



框架技术

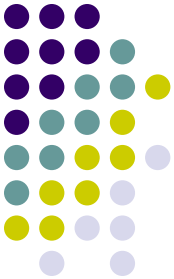
- 如何更快更好地盖房子？

如何更快更好地做软件呢？

框架技术

- “框架技术”帮我们更快更好地构建程序：
  - 是一个应用程序的半成品
  - 提供可重用的公共结构
  - 按一定规则组织的一组组件

- 优势：
  - 不用再考虑公共问题
  - 专心在业务实现上
  - 直接组装汽车，不用自己造轮子
  - 站在巨人的肩膀上，享用前人经验和智慧



技术、组件、框架和系统

概念	说明	示例1	示例2
技术	解决某一类问题的方法	锻造技术	JSP技术、JDBC技术 XML技术、 JavaScript技术
组件	应用程序里可重用的“零件”	空心钢管、铆钉	分页组件、控制器组件、视图组件
框架	一系列组件，按照一定的结构组合在一起作为系统开发的平台	自行车车架	Struts、Spring、Hibernate、WebWork、JSF、EJB DWR框架
系统	实现完整功能的应用程序	自行车	物流管理系统、销售系统

框架技术

- 软件大师Ralph Johnson的框架(Framework)定义
  - 框架是整个系统或系统的一部分的可重用设计，由一组抽象的类及其实例间的相互作用方式组成。框架一般具有即插即用的可重用性、成熟的稳定性以及良好的团队协作性。
  - 使用框架的好处是

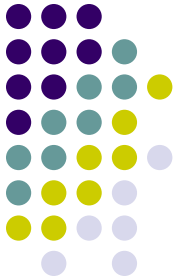
    （1）在好的框架下，开发者只需要写一些必须的代码；他们不需要直接接触底层的API。 这一点很重要。

    （2）经过良好设计的框架可以为程序提供清晰的结构并且提高程序的内聚性。好清晰的结构使得其他人可以更容易加入项目。

    （3）一个容易使用的框架可以通过一些例子和文档为用户提供最佳实践。

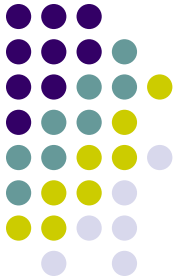
    （4）采用成功的框架的代码比自己的代码容易测试

JavaEE开源框架



- JavaEE复杂的多层结构决定了大型的J2EE项目需要运用框架和设计模式来控制软件质量。
- JavaEE标准平台本身就是一个基础的框架结构
- 为了进一步标准化JavaEE的开发代码，目前市场上出现了一些商业的、开源的基于JavaEE的应用框架，其中主流的框架技术有：基于MVC模式的Struts框架和基于IoC模式的 Spring框架以及对象/关系映射框架Hibernate等。

表示层框架Struts

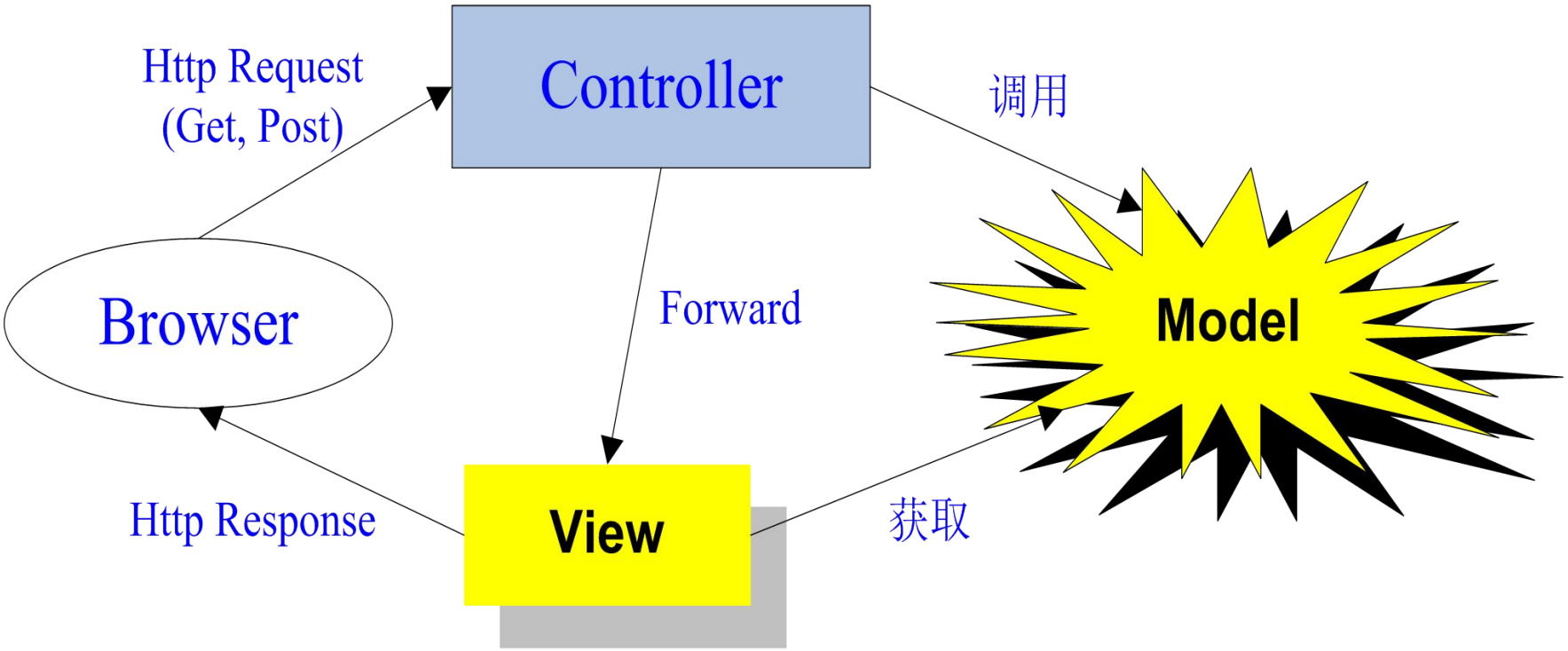


- Struts是一个在JSP Model2基础上实现的MVC框架，主要分为模型(Model)、视图(Viewer)和控制器(Controller)三部分，其主要的设计理念是通过控制器将表现逻辑和业务逻辑解耦，以提高系统的可维护性、可扩展性和可重用性。

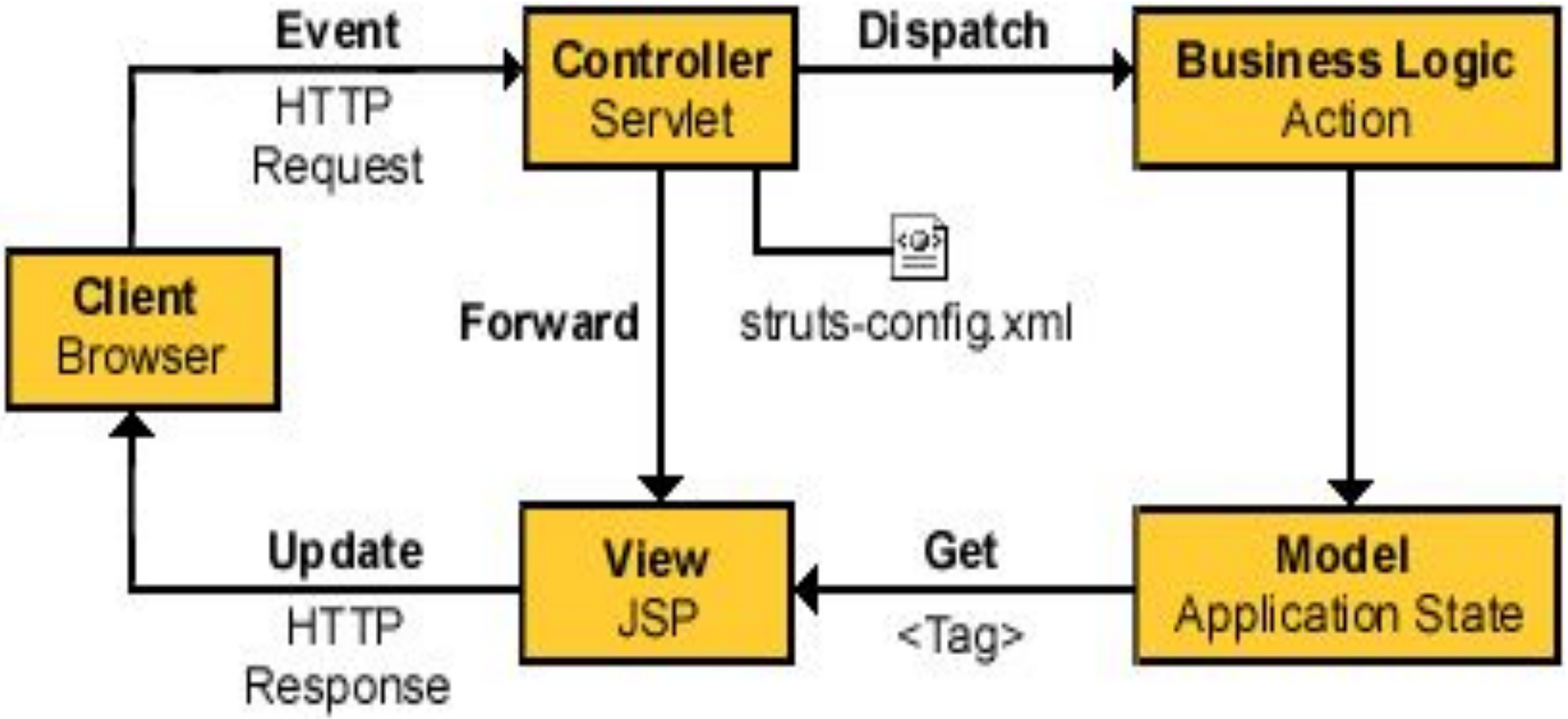
Model1与Model2设计模式简介

- 以JSP为中心的开发模型，称为Model1（JSP+JAVABEAN）
  - 业务逻辑与表示逻辑混和，不利维护与重用
  - HTML中嵌入了大量的JAVA代码
  - 验证、流程控制、更新程序的状态全部在JSP中完成
- 基于MVC模式的框架
  - MVC将问题进行分解
  - 模型包含应用程序的核心功能。模型封装了应用程序的业务逻辑。它对视图或控制器一无所知。
  - 视图提供模型的表示。它是应用程序的 外观。视图可以访问模型的读方法，但不能访问写方法。此外，它对控制器一无所知。
  - 控制器对用户的输入作出反应。它创建并设置模型。

Model2


Http Request
(Get, Post)
Controller
调用
Browser
Forward
Model
Http Response
View
获取

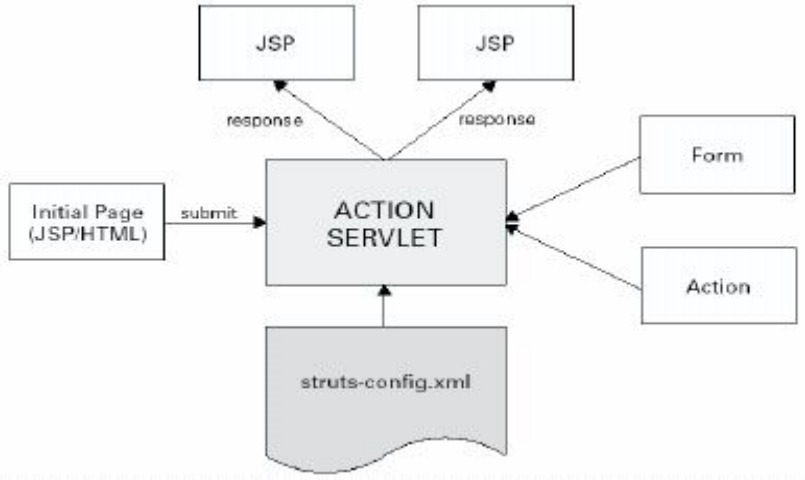
Struts框架概览


Event
HTTP
Request
Controller
Servlet
Dispatch
Business Logic
Action
Client
Browser
Forward
struts-config.xml
Update
HTTP
Response
View
JSP
Get
<Tag>
Model
Application State

Struts框架概览

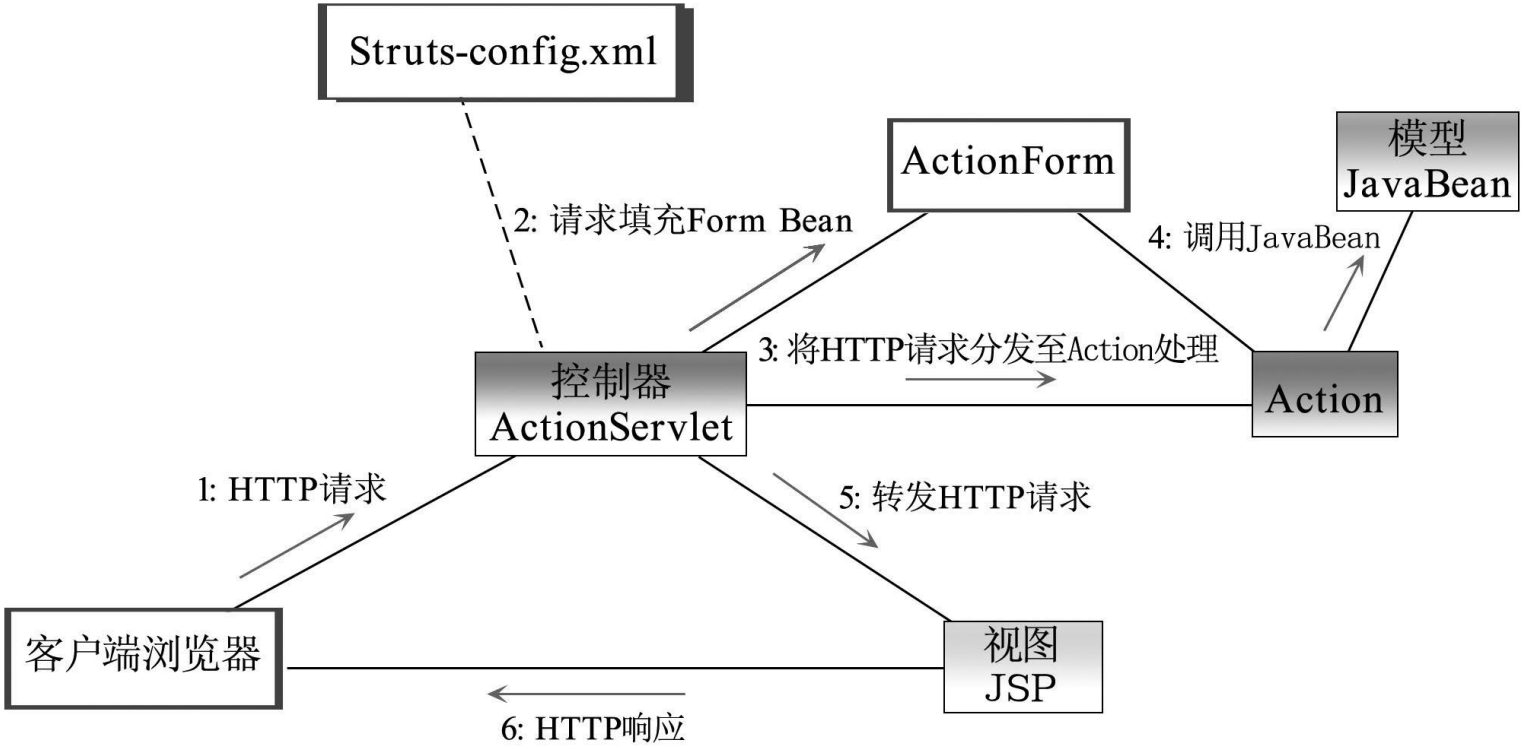
- 浏览器
  - web容器将对来自HTTP的每个请求创建一个request对象，并用一个response对象作出响应
- 控制器
  - 控制器接收来自浏览器的请求，在struts中，是由一个servlet来充当控制器的角色,struts-config.xml文件配置控制器
- 模型
  - 在struts中，由Action类充当业务逻辑的包装器，ActionForm是程序的状态
- 视图
  - JSP文件

Struts框架组件



- ActionServlet类控制导航流
- ActionServlet根据URI来决定哪个Action类被用于处理请求,Action可以校验输入,并访问业务层以便从数据库检索信息
- Action需要知道页面提交了哪些内容,所以由ActionServlet根据请求URI来决定将请求参数绑定到哪个ActionForm中,并传入Action
- Action在完成业务逻辑后,返回一个ActionForward对象,ActionServlet根据ActionForward对象中的路径来调用页面完成响应
- Struts将这些信息绑定在一个ActionMapping对象中,一个ActionMapping对应一个请求URI,当请求路径到达的时候,ActionServlet就会查询ActionMapping对象，ActionMapping对象将告诉ActionServlet哪个Action类会被调用、哪个ActionForm类被用于传递页面数据以及哪些ActionForward将被用于转向
- 有关Action、ActionForm、ActionForward等信息，Struts通过一个配置文件：struts-config.xml文件来定义。

Struts工作流程


Struts-config.xml
ActionForm
模型
JavaBean
2: 请求填充Form Bean
4: 调用JavaBean
3: 将HTTP请求分发至Action处理
控制器
ActionServlet
Action
1: HTTP请求
5: 转发HTTP请求
客户端浏览器
视图
JSP
6: HTTP响应

一个简单的Struts项目

- 第一个项目，实现用户登录操作
  - 用户将看到一个登录页面，要求用户输入用户名以及密码
  - 如果用户名以及密码都是admin，提示登录成功
  - 否则提示登录失败

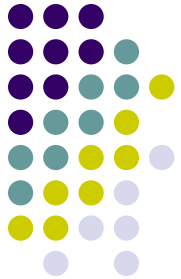
1、用MyEclipse创建一个J2EE Web应用项目struts_test1

2、导入Struts1.3的库包

3、配置ActionServlet:

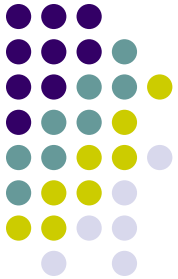
修改web项目的web.xml文件，添加如下Servlet映射配置

（转下一页）



web.xml 的配置

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>

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



我们将需要创建如下文件

- 一个ActionForm – LoginActionForm.java
- 一个Action – LoginAction.java
- struts-config.xml文件
- 三个页面
  - 登录页面 – login.jsp
  - 登录成功提示页面 – login_success.jsp
  - 登录失败提示页面 – login_error.jsp

- 就这些！没别的了！！

创建LoginActionForm.java

- ActionForm是一个JavaBean，需继承org.apache.struts.action.ActionForm类，它捕获通过HTTP请求传送的参数
- ActionForm针对每个HTML表单中的字段具有一个对应的属性
- ActionServlet匹配请求中的参数和ActionForm中的属性，并调用ActionForm中的setter方法，将参数传入ActionForm
- 我们的login.jsp有username和password两个表单字段（下面将会看到），所以，我们需要定义ActionForm中相应的setter方法：setUsername和setPassword方法
- ActionForm中的getter/setter方法，可以通过Eclipse集成环境，自动生成
- ActionForm中的内部属性全部定义为私有的（private），并通过公共(public)的getter/setter方法来访问

```
package com.strutstest;

import org.apache.struts.action.ActionForm;

public class LoginActionForm extends ActionForm {
	private String username;
	private String password;
	/**
	 * @return Returns the password.
	 */
	public String getPassword() {
		return password;
	}
	/**
	 * @param password The password to set.
	 */
	public void setPassword(String password) {
		this.password = password;
	}
	/**
	 * @return Returns the username.
	 */
	public String getUsername() {
		return username;
	}
	/**
	 * @param username The username to set.
	 */
	public void setUsername(String username) {
		this.username = username;
	}
}
```

创建LoginAction.java

- Action是一个Java类，需继承org.apache.struts.action.Action类
- ActionServlet将会组装ActionForm，并将它传递给Action
- Action 通常负责：
  - 输入校验
  - 调用业务逻辑类执行业务逻辑操作
  - 决定返回哪个ActionForward
- 我们的LoginAction做了如下事情，这些是一个Action通常都会做的最典型的事情：
  - 将输入的ActionForm强制转换为LoginActionForm
  - 从LoginActionForm对象中获取用户名以及密码的数据信息
  - 执行用户名及密码的逻辑判断操作（在通常的情况下，要将这些业务逻辑交给专门的类去处理，这里这样做是为了演示的需要）
  - 根据业务逻辑执行的结果，决定返回哪个ActionForward，我们在这里使用success这个标识来表示登录成功页面，用error标识来表示登录失败页面

```
package com.strutstest;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public class LoginAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
                                 HttpServletRequest request, HttpServletResponse response) throws Exception {
        //将ActionForm强制转换为LoginActionForm
        LoginActionForm loginForm = (LoginActionForm)form;

        //从LoginActionForm中提取从页面表单传递过来的参数
        String username = loginForm.getUsername();
        String password = loginForm.getPassword();

        //根据这些参数，执行业务逻辑操作
            if("admin".equals(username) && "admin".equals(password)){
                // 如果用户名和密码均为admin，则转向登录成功页面
                return mapping.findForward("success");
        }else{
                //否则转向登录失败页面
                return mapping.findForward("error");
        }
    }
}
```

创建Struts配置文件struts-config.xml

- 在WebContent/WEB-INF目录下创建struts-config.xml文件
- 并添加如下内容（空白的struts-config.xml）,紧接着，我们将往这个空白的配置文件中添加其它配置信息

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC
        "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
        "http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">
<struts-config>
</struts-config>
```

- struts-config.xml文件，是由ActionServlet读取的配置文件，它定义了所有关于Action、ActionForm、ActionForward等的详细信息

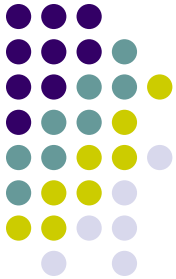
添加ActionForm配置，配置LoginActionForm

- 我们在struts-config.xml文件中，在<struts-config>标签的内部，添加如下配置：

```
<form-beans>
                    <form-bean name="loginForm"
  type="com.strutstest.LoginActionForm"/>
</form-beans>
```

- <form-beans>标签内部可以包含多个<form-bean>标签
- <form-bean>标签必须指定name和type属性
  - name属性是给此ActionForm一个标识名称
  - type属性指定了此ActionForm是哪个类，必须是全路径的类名

添加Action配置，配置LoginAction



- 我们在struts-config.xml文件中，紧接着<form-beans>标签的下面，添加对LoginAction的配置
- <action>标签可以配置的重要属性包括：
  - path-从页面上通过一个什么样的URL路径来访问Action（不包含.do）
  - type – 访问这个URL的时候，调用哪个Action类，这是Action的全路径类名
  - name – 这个属性用来标识哪个ActionForm将被创建，并将提交的表单组件给它
  - scope – FormBean的作用域范围，可以取值为session和request，一般取值都是request

```
<action-mappings>
<action
path="/login"
type="com.strutstest.LoginAction"
name="loginForm"
scope="request"
>
      <forward name="success"
          path="/login_success.jsp"/>
       <forward name="error"
          path="/login_error.jsp"/>
</action>
</action-mappings>
```

创建login.jsp

- 在WebContent目录下创建login.jsp文件，如右边所示
- 添加一个表单，action为login.do，这个login.do的意思，将会告诉struts的ActionServlet，它将需要调用哪个Action来处理这个表单的请求
- 添加输入域username，这个username的表单字段，必须跟LoginActionForm中的属性一致
- 添加密码输入域password

```
<%@ page language="java" contentType="text/html;
      charset=GB18030"
   pageEncoding="GB18030"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
      Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
      charset=GB18030">
<title>请登录</title>
</head>
<body>
<form action="login.do" method="post">
请输入用户名：<input type="text" name="username"> <br/>
请输入密码：<input type="password" name="password">
      <br/>
<input type="submit" name="submit1" value="登录">
</form>
</body>
</html>
```

创建login_success.jsp和login_error.jsp

- login_success.jsp

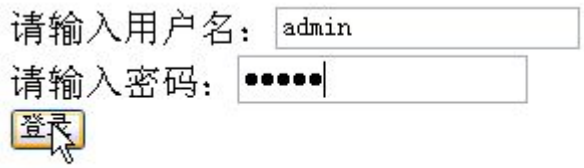
```
<%@ page language="java"
    contentType="text/html; charset=GB18030"
  pageEncoding="GB18030"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN">
<html>
<head>
      <meta http-equiv="Content-Type"
    content="text/html; charset=GB18030">
<title>登录成功</title>
</head>
<body>
欢迎您，您已经成功登录！您创建的第一个Struts应用程
    序已成功运行！！！
</body>
</html>
```

- login_error.jsp

```
<%@ page language="java"
    contentType="text/html; charset=GB18030"
  pageEncoding="GB18030"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
    4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type"
    content="text/html; charset=GB18030">
<title>登录失败</title>
</head>
<body>
您的登录失败了，可能原因是用户名或密码不正确，请
    返回重新输入
<a href="login.jsp">返回登录页面</a>
</body>
</html>
```

启动Tomcat并运行login.jsp

- **运行login.jsp之后，能看到如下所示的登录表单**

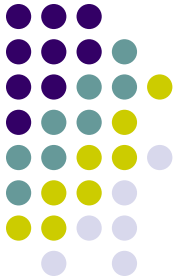


- **输入用户名admin和密码admin，将能看到登录成功的界面**

欢迎您，您已经成功登录！您创建的第一个Struts应用程序已成功运行！！！

- **输入其它用户名或密码，将能看到登录失败的界面**

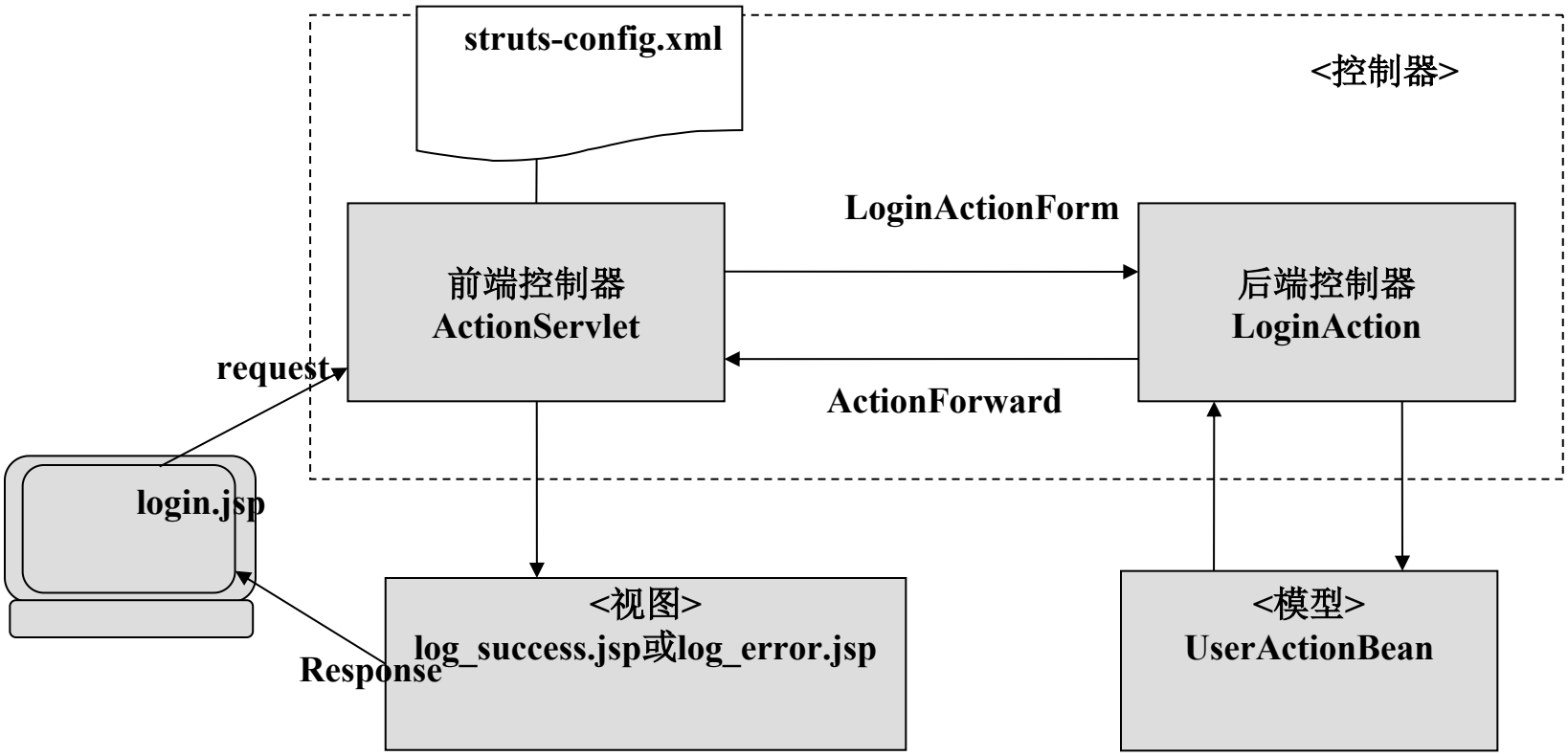
您的登录失败了，可能原因是用户名或密码不正确，请返回重新输入 返回登录页面



在这个简单的应用程序背后发生了什么？

- 当你从浏览器输入地址：http://localhost:8088/Struts-Test/login.jsp，Tomcat将按通常情况来处理这个JSP并返回浏览器
- 当你提交表单，实际上是提交到了这样一个URL地址：http://localhost:8088/Struts-Test/login.do，Tomcat将会根据web.xml的配置，将这个请求发送给相应的Servlet，在我们的应用中，Tomcat将会把这个请求发送给org.apache.struts.action.ActionServlet这个类（请参看web.xml的配置）
- 然后ActionServlet根据struts-config.xml的配置信息，调用LoginAction对象去处理这个请求，在此之前，它会将页面表单的请求数据封装到LoginActionForm对象中，并传递给LoginAction
- LoginAction返回一个ActionForward对象，包含了将要转向的路径信息
- ActionServlet根据这个ActionForward对象所包含的路径信息，调用相应的页面去执行响应
- 应用程序处理流程请参考下一页

LoginAction应用程序的处理流程



Struts与MVC

- 视图（View）
  - 在使用Struts框架的web应用程序中，JSP以及相关的技术（如Taglib）等共同组成视图层，这一层的主要职责是显示用户界面。Struts提供了很多机制让我们能更加轻松地创建视图
- 控制器（Controller）
  - Struts中，ActionServlet是控制器层组件
- 模型（Model）
  - 模型包括：系统的内部状态以及改变系统状态的动作
  - Struts中的Action和ActionForm是模型的一部分
  - Struts建议把”做什么”(Action)和”如何做”(业务逻辑)相分离

创建业务逻辑处理类（Model）

- 使用单例模式(Singleton)来创建业务逻辑处理类
- 创建UserManager业务逻辑处理类
- 创建validate方法
- 创建UserNotFoundException
- 创建PasswordErrorException

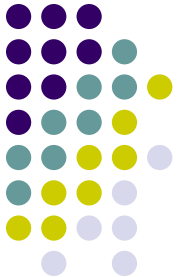
```
package com.bjsxt.strutstest;

public class UserManager {
	private static UserManager userManager;

	private UserManager(){
	}

	public static synchronized UserManager getInstance(){
		if(userManager == null){
			userManager = new
	UserManager();
		}
		return userManager;
	}

	public void validate(String username,String password)
		throws
	UserNotFoundException,PasswordErrorException
	{
		if(!"admin".equals(username)){
			throw new
	UserNotFoundException();
		}
		if(!"admin".equals(password)){
			throw new
	PasswordErrorException();
		}
	}
}
```

Action中如何调用业务逻辑处理类？

- 我们看下面的代码：

```
try {
        UserManager.getInstance().validate(username,password);
        return mapping.findForward("success");
} catch (UserNotFoundException e) {
        e.printStackTrace();
} catch (PasswordErrorException e) {
        e.printStackTrace();
}
return mapping.findForward("error");
```

- 通过添加业务逻辑处理类，我们将验证逻辑转移到了业务逻辑处理层



细节：所有的页面请求由容器接收

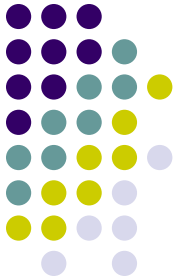
- Struts的核心组件是ActionServlet，像其它所有Servlet一样，它是生存在容器中的，比如Tomcat、WebLogic等，当容器启动的时候，它会读取web.xml文件（部署描述符），告诉容器它会装入哪些Servlet
- 一个标准的Servlet是通过servlet-mapping来设定，哪些请求，将会被提交到哪些servlet中
- Struts的servlet-mapping配置一般是：

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

- 这样配置的意思是：任何以.do结尾的URL请求，都会被发送到ActionServlet进行处理

Struts2

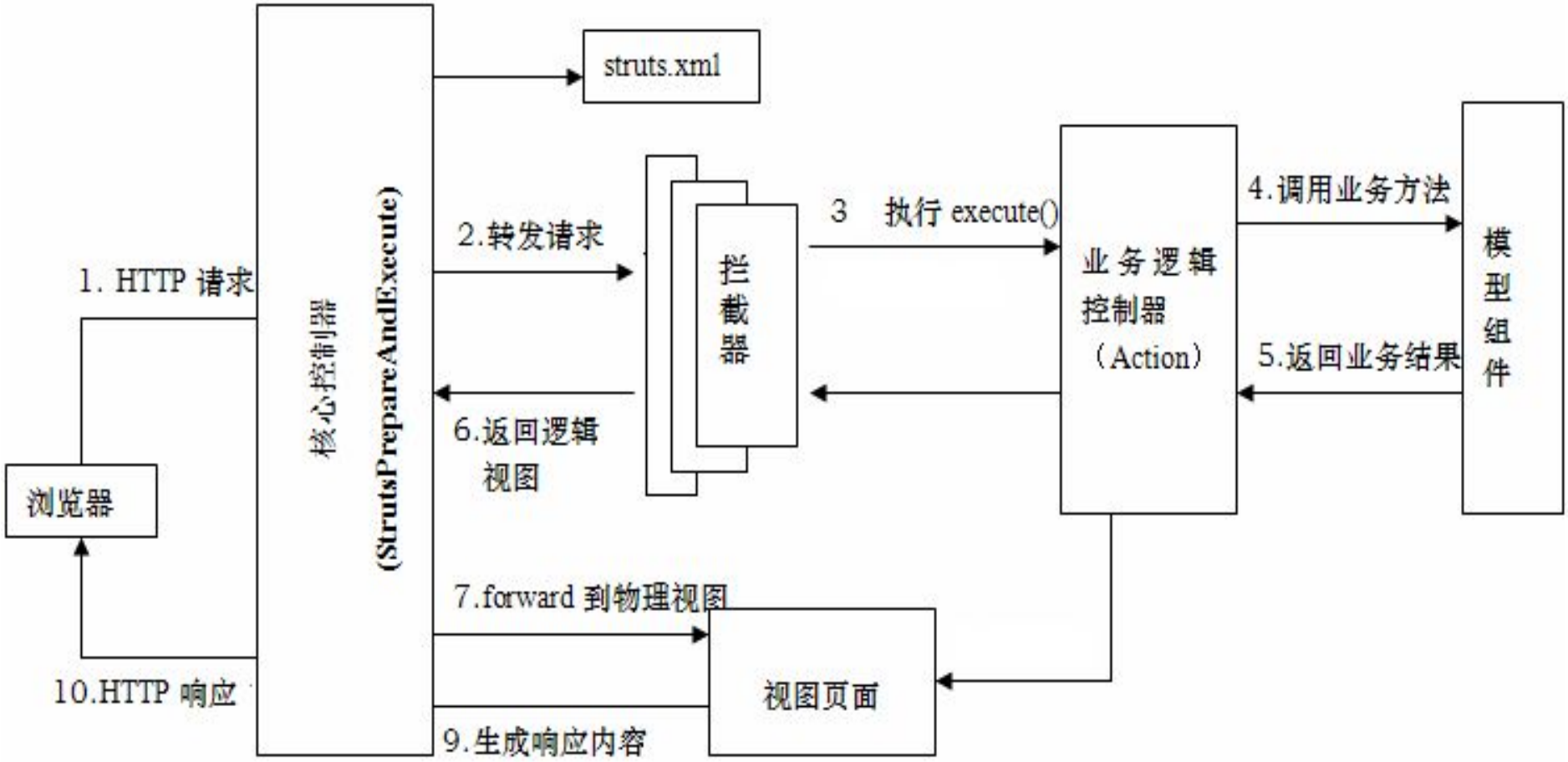
- Struts是Apache软件基金会赞助的一个开源项目，是一个基于JavaEE的MVC开源实现。是MVC设计模式中的一个经典产品，它为Servlet/JSP技术的应用提供了技术框架。
- Struts技术框架
  - Struts1
  - Struts2



Struts2

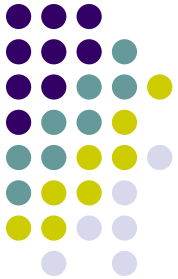
- Struts1于2001.7月正式发布
- 存在缺陷
  - 只支持JSP作为其表现层技术
  - 与Servlet API耦合严重，严重依赖于Web服务器，脱离服务器难于测试
  - 属于侵入式设计（Struts1的Action中包含了大量的Struts1 API），严重影响代码重用
- Struts2于2006年底正式发布，较好地解决了Struts1的缺陷，相对于Struts1而言，Struts2不是对Struts1的简单升级，它是一个全新的框架，继承了Struts1和WebWork的许多优点

Struts2



Struts2程序运行流程

Struts2



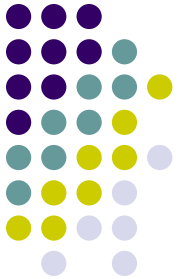
- Struts2项目的基本组成
  - 控制组件：核心控制器和业务控制(Action)
  - 模型组件：JavaBeans、EJB等
  - 视图组件：JSP、HTML页面等
  - 配置文件：web.xml、struts.xml等
- Struts2控制器分为核心控制器与业务控制器，核心控制器是org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter，业务控制器是Action。其中，核心控制器在Web应用中负责拦截所有的用户请求。

Struts2

Struts2的核心控制器（Controller）是一个过滤器，所有的用户请求都需要经过该过滤器。当StrutsPrepareAndExecuteFilter接受到一个用户请求后，会根据相关的配置信息查找服务于该请求的interceptors、action类并自动创建它们的实例及调用它的方法服务于请求，它还会根据interceptors或action的执行结果来调用视图层组件Result来生成响应。

StrutsPrepareAndExecuteFilter过滤器必须在应用的web.xml中部署，并配置为过滤所有请求。

Struts2



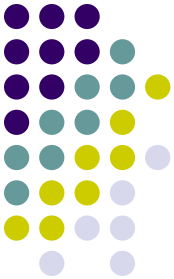
- **Struts2开发步骤：**
  - 创建web项目
  - 为项目添加Struts2类包支持
  - 设置核心控制器（配置web.xml)
  - 创建视图页面
  - 创建业务逻辑控制器（Action)
  - 创建struts.xml配置Action
  - 部署和运行struts2项目

Struts2

- **第一步：创建一个Web工程 ：在MyEclipse，通过菜单File->New->Web Project 创建工程struts_test3**
- **第二步：工程项目导入Struts2的核心支持包**
  - commons-fileupload-1.2.1.jar
  - commons-io-1.3.2.jar
  - commons-logging-1.0.4.jar
  - freemarker-2.3.15.jar
  - ognl-2.7.3.jar
  - struts2-core-2.1.8.1.jar
  - xwork-core-2.1.6.jar

简注：Struts2有大量的jar包，支持大量的功能，不同类型的应用可能需要不同的包支持。以上的5个包为Struts2的核心包，使用Struts2必须使用。

Struts2



第三步：配置struts2转发过滤器 ：编辑web.xml文件，添加以下内容

```
<filter>
        <filter-name>struts2</filter-name>
        <filter-class>     org.apache.struts2.dispatcher.FilterDispatcher
        </filter-class>
</filter>
<filter-mapping>
        <filter-name>struts2</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

- 简注：“/*”表示涉及本工程的所有浏览器端的请求都经过struts2过滤器处理。

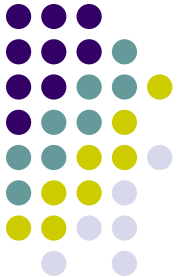
Struts2

- **第四步：创建输入页面login.jsp、结果页面welcome.jsp和error.jsp**

  login.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="GB2312"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head><title>登录界面</title></head>
  <body>
    <form action="LoginAction.action">
      用户名：<input name="username"><br>
      密 码：<input type="password" name="userpass"><br>
      <input type="submit" value="提 交">
      <input type="reset" value="取 消">
    </form>
  </body>
</html>
```

Struts2



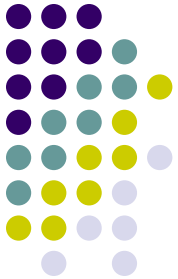
- welcome.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="GB2312"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>欢迎</title>
  </head>
  <body>
    <font color="red" size="10">登录成功！</font>
  </body>
</html>
```

- error.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="GB2312"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title></title>
  </head>
  <body>
    <font color="red" size="10">用户或密码错误！</font>
  </body>
</html>
```

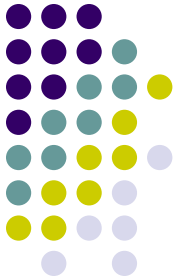
Struts2



- **第五步：创建Action文件LoginAction**

  LoginAction.java

```
package com.strutstest;
import com.opensymphony.xwork2.ActionSupport;
public class LoginAction extends ActionSupport{
 private String username;
 private String userpass;
  public String execute(){
  if("daniel".equals(username)&&"abcde".equals(userpass))
   return SUCCESS;
  else
   return ERROR;
 }
  public String getUsername() {
  return username;
 }
 public void setUsername(String username) {
  this.username = username;
 }
 public String getUserpass() {
  return userpass;
 }
 public void setUserpass(String userpass) {
  this.userpass = userpass;
 }
}
```



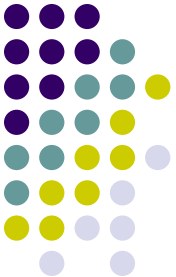
Struts2

- **第六步：创建并配置struts2.xml文件**
  - struts.xml

```
<!DOCTYPE struts PUBLIC
 "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
 "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
 <package name="struts2demo" extends="struts-default">
  <action name="loginAction" class="com.strutstest.LoginAction">
   <result name="success">/welcome.jsp</result>
   <result name="error">/error.jsp</result>
  </action>
 </package>
</struts>
```

  - 简注：默认配置情况下执行execute()方法，实际应用中经常更改配置。本书后面将深入讲解。注意本类中的username和userpass必须和网页文件的name属性名一致。
- **第七步：部署并运行项目**

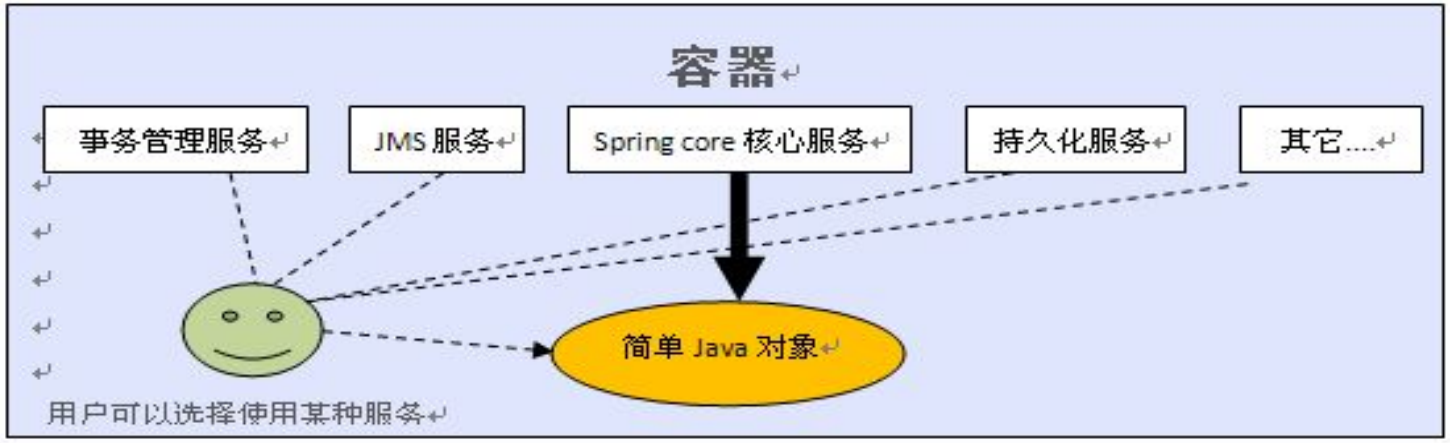
业务逻辑层框架Spring



- Spring是一个开源框架，它是为了解决企业应用开发的复杂性而创建的。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。
- Spring是一个轻量级（Lightweight）的控制反转(IoC, Inversion of Control)和面向切面(AOP, Aspect-oriented programming)的容器框架，它提供对持久层、事务的支持；提供MVC Web框架的实现，并对一些常用的企业服务API提供一致的模型封装。

Spring

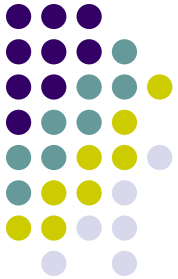
- 轻量级(Lightweight)
  - 划分一个应用是否属于轻量级还是重量级,主要看它使用了多少服务
  - 对于spring容器，它提供了很多服务，但这些服务并不是默认为应用打开的;目前EJB容器就因为它默认为应用提供了EJB规范中所有的功能，所以它属于重量级。



Spring

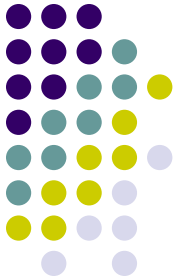
- 容器(Container)
  - **容器可以管理对象的生成、资源取得、销毁等生命周期，甚至建立对象对对象之间的依赖关系。**
- 非侵入性(NoIntrusive)
  - **有些框架一旦被使用，应用程序对框架就有了依赖性**
    - **使用大量框架API**
    - **继承API某些类型等**
  - **Spring的目标之一希望让应用程序几乎感受不到框架的存在，减少框架移植时的负担**

Spring



- 控制反转(IoC)与依赖注入(DI)
  - **IoC，就是由容器控制程序之间的关系，而非传统实现中由程序代码直接控制，这就是所谓的“控制反转”的概念所在：控制权由应用代码中转到了外部容器，控制权的转移，是所谓的反转。**
  - **DI（Dependency Injection），所谓依赖注入，即组件之间的依赖关系由容器在运行期决定，形象地来说，即由容器动态地将某种依赖关系注入到组件之中。**

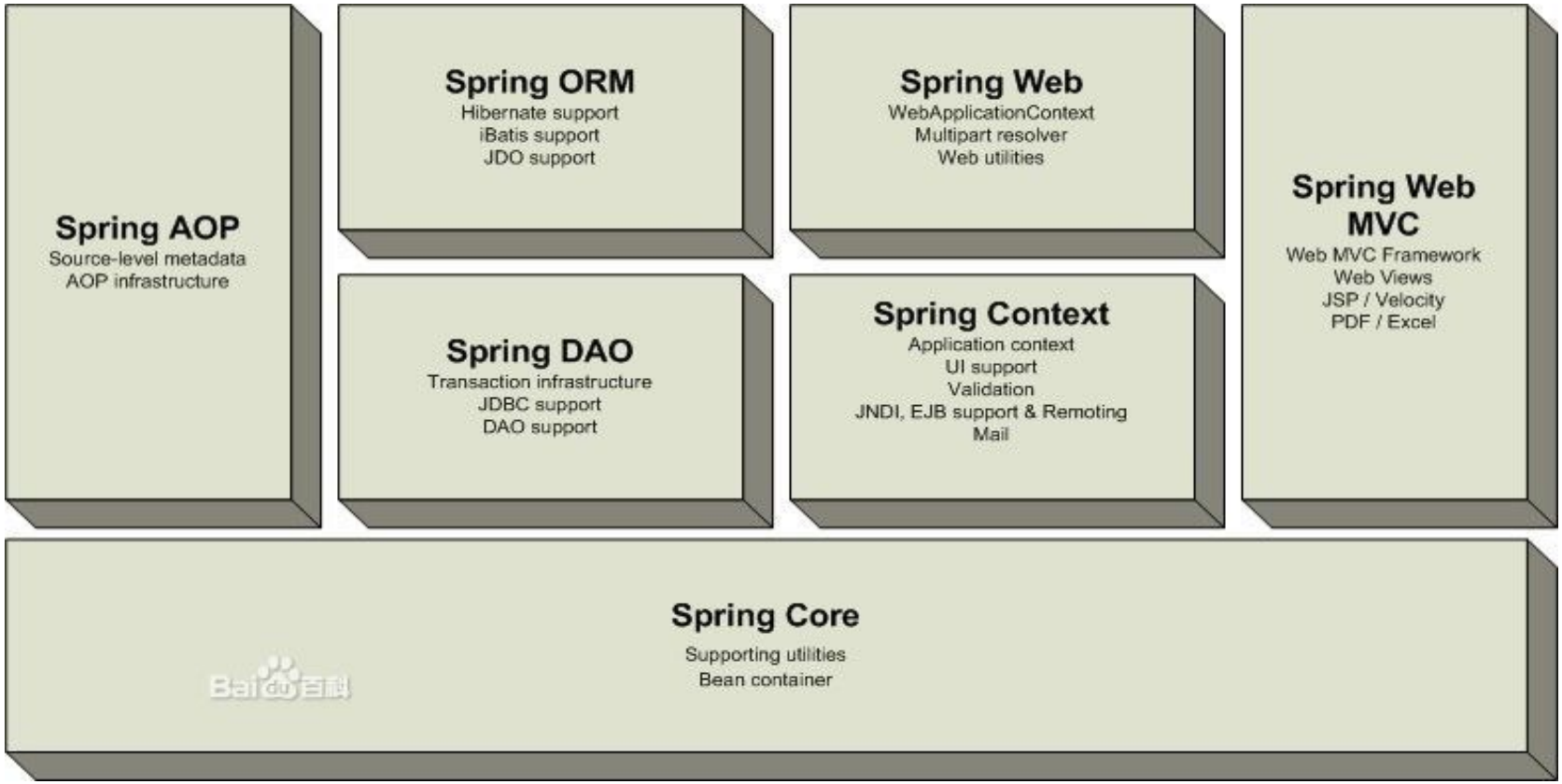
Spring

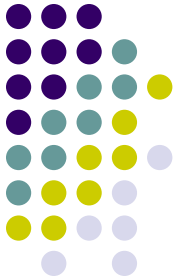


- **AOP（Aspect Oriented Programming）面向切面编程：**
  - 在一个服务的流程中插入与业务逻辑无关的系统服务逻辑(例如:logging、security),这样的逻辑称为cross-cutting concerns(横切关注点)，将cross-cutting concerns独立出来设计为一个对象，这样的特殊对象称之为Aspect(切面)， Aspect-oriented programming(面向切面编程)着重在Aspect的设计上以及与应用程序的织入(Weave)
  - **AOP则是针对业务处理过程中的切面进行提取，它所面对的是处理过程中的某个步骤或阶段，以获得逻辑过程中各部分之间低耦合性的隔离效果。**

Spring

- **Spring框架结构图**





- Spring框架由七个定义明确的模块组成，且每个模块或组件都可以单独存在，或者与其他一个或多个模块联合实现。Spring Core Container是一个用来管理业务组件的IoC容器，是Spring应用的核心；Spring DAO和Spring ORM不仅提供数据访问的抽象模块，还集成了对Hibernate、JDO和iBatis等流行的对象关系映射框架的支持模块，并且提供了缓冲连接池、事务处理等重要的服务功能，保证了系统的性能和数据的完整性；Sprnig Web模块提供了Web应用的一些抽象封装，可以将Struts、Webwork等Web框架与Spring整合成为适用于自己的解决方案。
- Spring框架可以成为企业级应用程序一站式的解决方案，同时它也是模块化的框架，允许开发人员自由地挑选适合自己应用的模块进行开发。Spring框架式是一个松耦合的框架，框架的部分耦合度被设计为最小，在各个层次上具体选用哪个框架取决于开发者的需要。

数据持久层框架Hibernate

- O/R mapping技术是为了解决关系型数据库和面向对象的程序设计之间不匹配的矛盾而产生的。Hibernate是目前最为流行的O/R mapping框架，它在关系型数据库和Java对象之间做了一个自动映射，使得程序员可以以非常简单的方式实现对数据库的操作。Hibernate工作原理如图3所示。

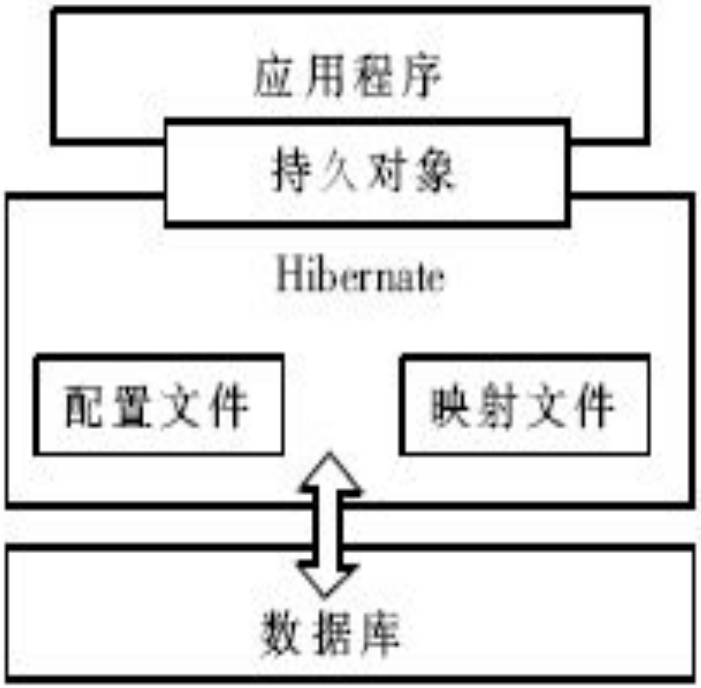
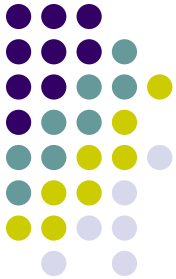


图 3 Hibernate 工作原理

- Hibernate通过对JDBC的封装，向程序员屏蔽了底层的数据库操作，使程序员专注于OO程序的开发，有助于提高开发效率。程序员访问数据库所需要做的就是为持久化对象编制xml映射文件。
- 底层数据库的改变只需要简单地更改初始化配置文件(hibernate.cfg.xml或者hibernate.properties)即可，不会对应用程序产生影响。
- Hibernate有自己的面向对象的查询语言HQL，HQL功能强大，支持目前大部分主流的数据库，如Oracle、DB2、MySQL、Microsoft SQL Server等，是目前应用最广泛的O/R映射工具。Hibernate为快速开发应用程序提供了底层的支持。

基于SSH组合框架的Web应用



- 集成SSH的新型JavaEE框架
  - 前面分析了基于J2EE的三种框架技术，下面通过集成以上三种框架技术来对传统的J2EE Web开发模型加以改进，以形成一种新的、轻量型的J2EE架构。
  - 集成SSH框架 的系统框架图 如图4所示，系统从职责上分为四层：表示层、业务逻辑层、数据持久层和域模块层。其中使用Struts作为系统的整体基础架构，负责MVC的分离，在 Struts框架的模型部分，利用Hibernate框架对持久层提供支持，业务层用Spring支持。
  - 具体做法是：用面向对象的分析方法根据需求提出一些模型，将这些模型实现为基本的Java对象，然后编写基本的DAO接口，并给出Hibernate的DAO实现，采用Hibernate架构实现的 DAO类来实现Java类与数据库之间的转换和访问，最后由Spring完成业务逻辑。

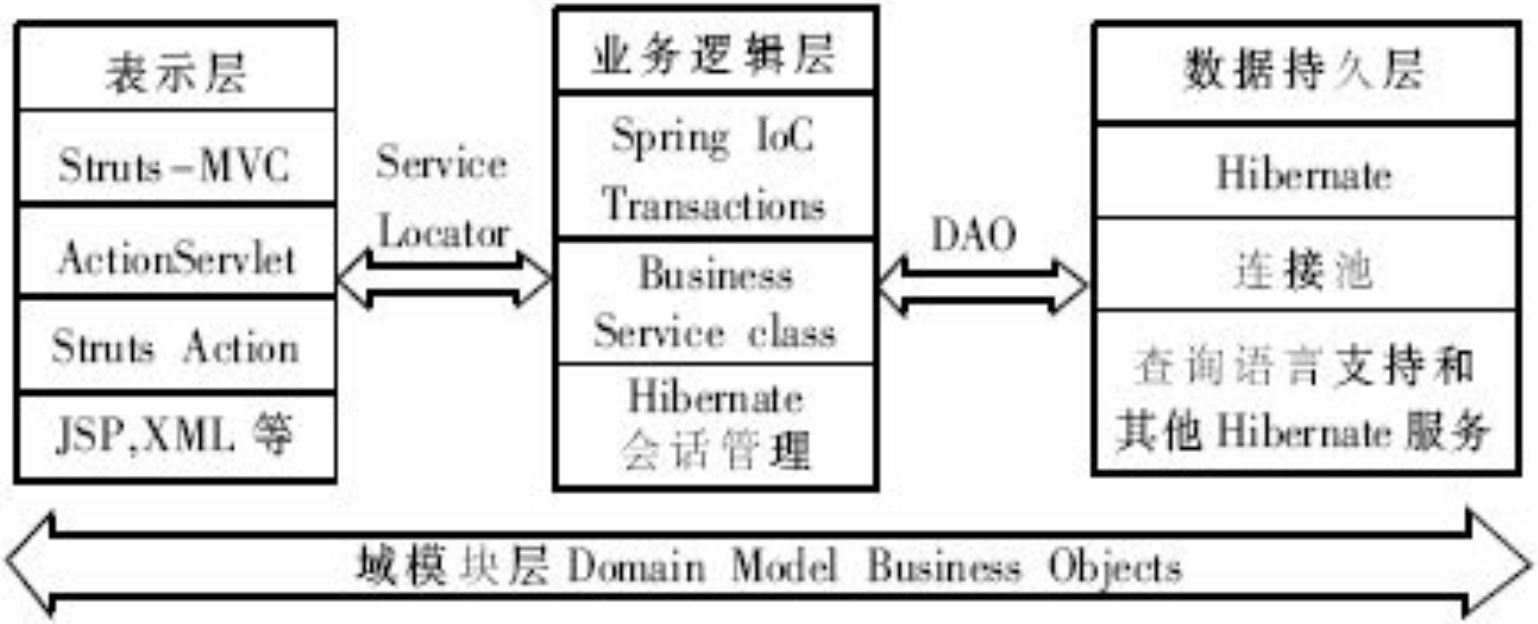


图 4 集成 SSH 框架的系统架构图

- 系统的基本业务流程是：在表示层中，首先通过JSP页面实现交互界面，负责传送请求(Request)和接收响应(Response)，然后Struts根据配置文件(struts-config.xml)将ActionServlet接收到的Request委派给相应的Action处理。在业务层中，管理服务组件的 Spring IoC容器负责向Action提供业务模型(Model)组件和该组件的协作对象数据处理(DAO)组件完成业务逻辑，并提供事务处理、缓冲池等容器组件以提升系统性能和保证数据的完整性。而在持久层中，则依赖于Hibernate的对象化映射和数据库交互，处理DAO组件请求的数据，并返回处理结果。
- 采用上述开发模型，不仅实现了视图、控制器与模型的彻底分离，而且还实现了业务逻辑层与持久层的分离。这样无论前端如何变化，模型层只需很少的改动，并且数据库的变化也不会对前端有所影响，大大提高了系统的可复用性。而且由于不同层之间耦合度小，有利于团队成员并行工作，大大提高了开发效率。

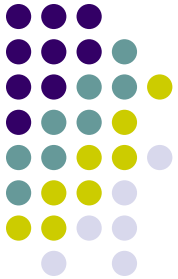
- **基于SSH框架 的Web应用系统的实现**
  - **下面将通过一个实际的系统来展示如何进行基于SSH框架 的Web应用开发。该系统是为某通信公司运营部开发的一个问答式系统，功能类似于百度知道和新浪爱问。由于系统的模块较多，下面就以一个用户管理模块为例来说明系统的开发实现过程，并将按照数据持久层、业务逻辑层、表示层的顺序说明系统构建过程。**

（1）数据持久层

- 数据持久层由Java对象持久化类和数据访问对象(DAO)组成。每个数据库表都对应着一个持久化对象，这样就给予了开发者使用OO思想设计和开发的便利，同时也屏蔽了具体的数据库和具体的数据表、字段，消除了对数据库操作的硬编码在重用性上的弊端。用户信息表的部分结构如表1所示。

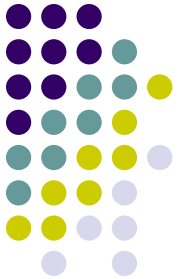
表1 用户信息表

表名：user			
字段名	数据类型	大小	空
user_id	numeric	10	否
user_name	varchar	40	否
… …			



- Hibernate通过映射(Mapping)文件将对象(Object)与关系型数据(Relational)相关联，因此需要编写和数据库表相对应的Java持久化类以及对应的映射文件。有了Java持久化类后就可以在此基础上实现数据访问类。在Spring框架中，数据访问类可以从辅助类 HibernateDaoSupport继承，这极大地方便了Hibernate框架在Spring中的使用，相应的部分代码如下。而具体的Hibernate数据源、session工厂、事务管理、缓冲连接池等功能都由业务层的Spring容器提供。

```
public class UserDao extends HibernateDaoSupport {
    public int add(User user) {
      return
    Integer.ParseInt(this.getHibernateTemplate().save(user).toString());
    }
    public List findAll() {
      return this.getHibernateTemplate().loadAll(User.class);
    }
}
```



（2）业务逻辑层

- 业务逻辑层由Spring框架支持，提供了处理业务逻辑的服务组件。开发者需要对业务对象建模，抽象出业务模型并封装在Model组件中。由于数据持久层实现了Java持久化类并且封装了数据访问对象(DAO)，因此可以在Model组件中方便地调用DAO组件来存取数据。Spring的IoC容器负责统一管理Model组件和DAO组件以及Spring所提供的事务处理、缓冲连接池等服务组件。
- 在用户管理模块中，通过业务建模创建了用户模型UserService类，封装了对用户的权限管理以及积分管理等功能。UserService类通过调用数据访问类UserDao实现对用户数据的操作。这些组件的关系将通过配置Spring框架的applicationContext.xml联系起来，配置文件的主要内容如下：

```
<!—创建业务模型组件 UserService,并调用 UserDao
组件作为协作对象-->
<bean  id="UserService"
      class="com.cellcom.ea2.business.
          UserService">
      <property  name="UserDao">
          <ref  local="UserDao"/>
      </property>
</bean>
<!—创建数据访问组件 UserDao, 并调用 Hibernate
的 session 工厂作为协作对象 -->
 <bean  id="UserDao"
      class="com.cellcom.ea2.dao.UserDao">
      <property  name="sessionFactory">
          <ref  local="sessionFactory"/>
      </property>
</bean>
```

（3）表示层

- 表示层结合JSP和Struts的TagLib库处理显示功能，利用ActionServlet将请求(*.do)映射到相应的Action，并由Action调用业务逻辑的服务组件，然后根据处理结果跳转到Forword对象指定的响应页面。
- 业务流程的部署由struts-config.xml完成。下面以一个显示所有用户信息的请求(ListUser.do)为例来说明配置文件的使用。

```
<!—该请求调用 ListUserAction 对象,并根据返回的
Forword 对象的状态来转到相应的页面 ——>
<action  path="/ListUser.do"
         type="com.cellcom.ea2.action.ListUserAction">
        <forward name="success" path="/listuser.jsp />
        <forward name="failure" path="/error.jsp />
</action>
```