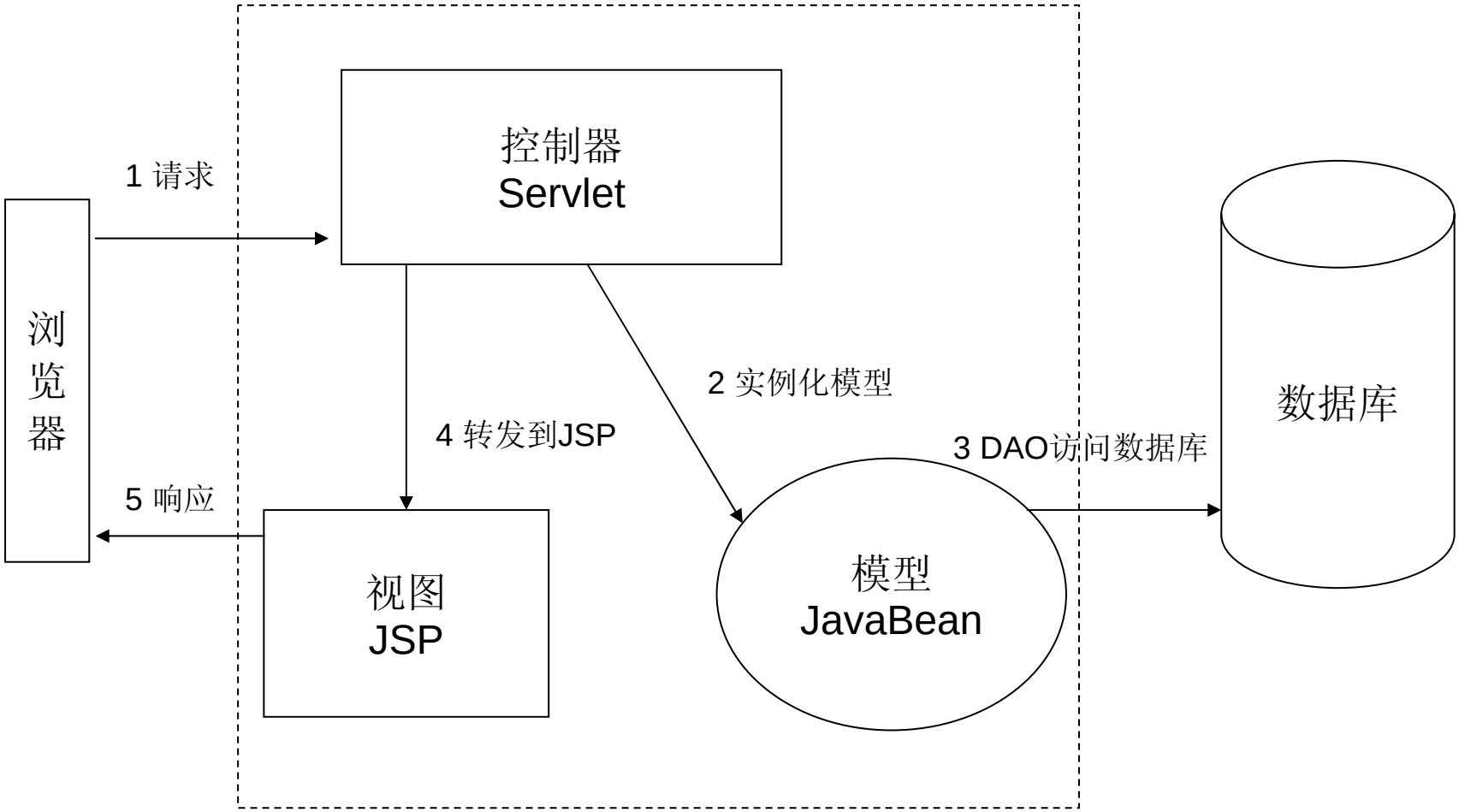
Struts概述

回顾MVC架构

- MVC（模型model 视图view 控制器controller）
- 模型组件：负责业务逻辑
- 视图组件：负责用户输入界面和显示数据界面
- 控制器组件：负责模型和视图之间的联系

JSP中MVC图示


控制器
Servlet
1 请求
浏览器
2 实例化模型
4 转发到JSP
3 DAO访问数据库
数据库
5 响应
视图
JSP
模型
JavaBean

框架framework概述

- 程序框架如同生活中的模板，我们可在已经搭好的基础环境下进行具体的自定义开发
- 框架包最简单的形式是一组类和接口，它们相互协作以解决特定类型的软件问题，特征如下：
  - 框架包包括大量类或组件，每一种类或组件都提供了某种概念的抽象
  - 框架包定义了这些抽象如何协作解决问题
  - 框架包组件可以重用
  - 框架包在更高层次上组织模型
- JAVA EE中的框架技术： Struts、Spring、Hibernate、WebWork等。

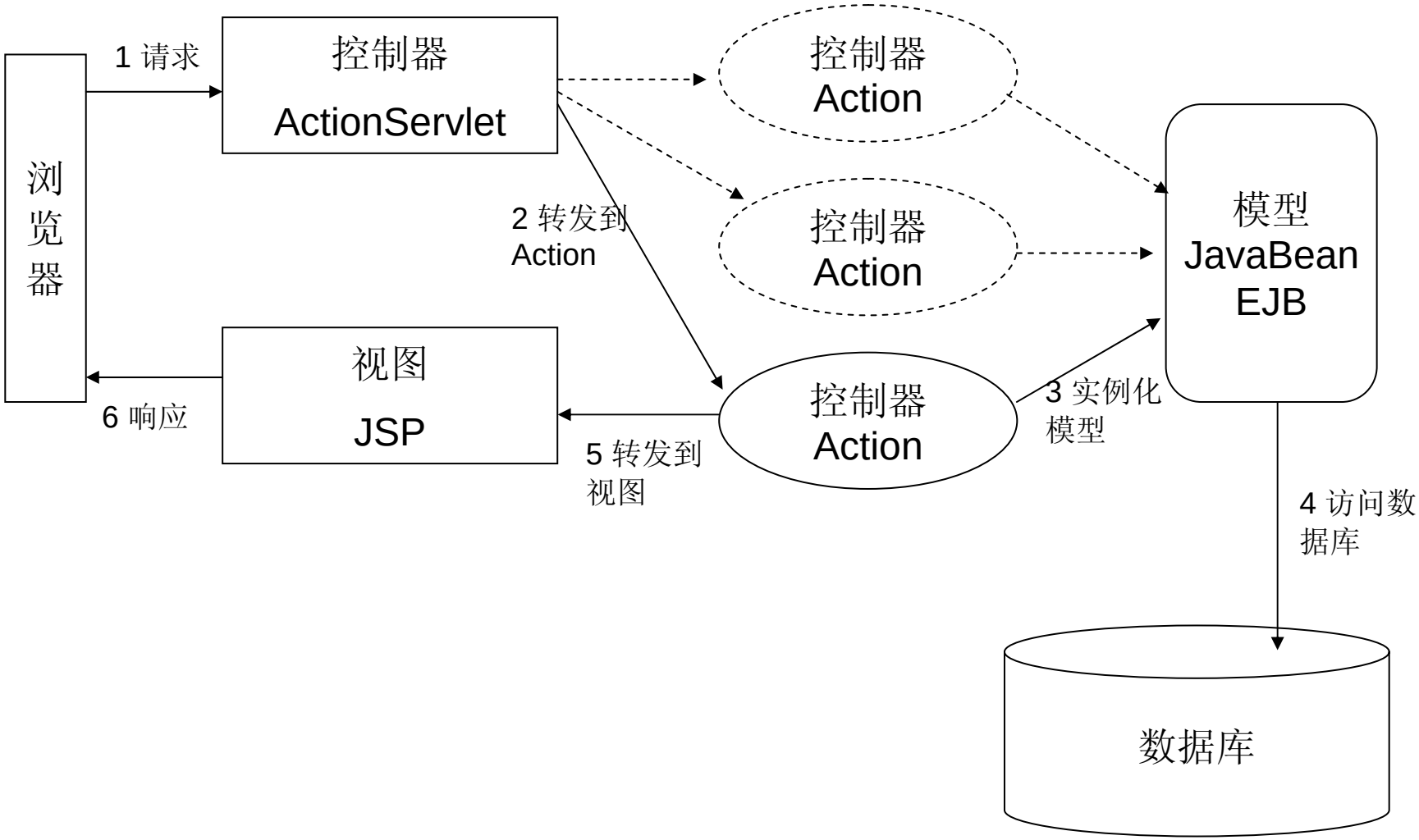
Struts框架概述

- Struts是一个基于Sun J2EE平台JAVA SERVLET/JSP技术的MVC应用框架
- 驻留在WEB层， 它把Servlet、JSP、自定义标签和信息资源(message resources)整合到一个统一的框架中
- 开发人员利用其进行开发时不用再自己编码实现全套MVC模式，极大的节省了时间。
- 现在，Struts是Apache软件基金会旗下Jakarta项目组的一部分，其官方网站是http://struts.apache.org

Struts框架与MVC

- 模型：
  - Struts框架没有提供任何模型组件，模型组件照常由javabean或者EJB组成。
- 视图：
  - 是一组JSP文件，可以在JSP中使用由Struts提供的客户化标签，通常把Struts框架中的ActionForm也划分到视图层
- 控制器：
  - 控制器由ActionServlet类和Action类（通常为Action子类）实现

Struts 基本MVC图



关于控制器controller

- 在Struts中ActionServlet,Action充当控制器
  - ActionServlet继承了javax.servlet.HttpServlet，在MVC模型中扮演中央控制器的角色。它的任务是接收http请求，然后根据struts-config.xml配置文件的配置信息，把请求转发给对应的Action对象
  - Action类负责调用模型的方法，如DAO等，之后转发到视图层

创建基于Struts的web应用程序

- 创建一个基本的web项目
- 在WEB-INF目录下添加下列文件
  - STRUTS框架配置文件
    - struts-config.xml
  - STRUTS框架标识库配置文件
    - struts-html.tld
    - struts-bean.tld
    - struts-logic.tld
    - struts-nested.tld
    - struts-tiles.tld
  - STRUTS框架plug-in配置文件
    - validation.xml
    - validator-rules.xml

创建基于Struts的web应用程序

- 在LIB目录下添加下列jar文件
  - antlr.jar
  - commons-beanutils.jar
  - commons-digester.jar
  - commons-fileupload.jar
  - commons-logging.jar
  - commons-validator.jar
  - jakarta-oro.jar
  - struts.jar

创建基于Struts的web应用程序

- 在web.xml文件中增加核心控制器ActionServlet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>0</load-on-startup>
</servlet>
<servlet-mapping>
   <servlet-name>action</servlet-name>
   <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

第一个HelloAction

- 功能实现：当客户端访问此Action时在控制台输出Hello,Action

HelloAction.java

```
package mystruts.action;

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 HelloAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
                    HttpServletRequest request, HttpServletResponse response)
                    throws Exception {

        System.out.println("Hello,Action");

        return null;
    }
}
```

配置Action映射

- 在struts-config.xml加入配置标签

```
<struts-config>
    ……
    <action-mappings>
        <action path="/helloAction“ type="mystruts.action.HelloAction"></action>
    </action-mappings>
    ……
</struts-config>
```

aciton标签属性
path 映射url路径，以/开头，表示当前web应用根目录
type 映射的Action类的完整类名

运行HelloAction

- 在浏览器地址栏输入
  - http://127.0.0.1:7777/mystruts/helloAction.do
- 在控制台输出
  - Hello,Action

HelloAction运行的流程

- 当用户请求mystruts/helloAction.do资源的时候，由于请求的地址是xxxx.do，所以会执行ActionServlet核心控制器（因为ActionServlet配置的时候就是<url-pattern>*.do</url-pattern>）
- 然后核心控制器读取内存中的Action配置信息，根据用户请求的helloAction.do找到对应的path=”/helloAction”的HelloAction类，如果这个类没创建过对象，就把这个对象创建出来
- 然后执行这个类的execute方法，打印输出Hello,Action，程序结束

- 注意 path=”/helloAction”不要写成path=”/helloAction.do”

org.apache.struts.action.Action类

- 当ActionServlet接收到请求后，会把请求转发到一个对应的Action对象，如果这个对象不存在会创建这个对象，然后调用Action对象的execute方法
- execute方法有4个参数
  - ActionMapping对象 存储了Action映射信息，对应<ation>标签
  - ActionForm对象 存储了ActionForm Bean信息，保存提交数据
  - HttpServletRequest请求对象
  - HttpServletResponse 响应对象
- execute方法返回ActionForward对象，这个对象封装了转发请求的信息。当返回一个ActionForward对象时，会自动按照此对象配置的路径信息转发到另外一个组件，如果返回null，则不进行转发
- 我们需要自定义一个类继承Action类并且覆盖execute方法

配置文件struts-config.xml

- 当web应用启动时会读取struts-config.xml配置文件，将配置信息存储到内存
- 标签元素一览
- <struts-config> 配置文件根元素，即最外层标签
  - <data-sources> 配置数据库连接池
  - <form-beans> 包含多个<form-bean> 子标签，配置ActionForm
  - <global-exceptions> 包含多个<exception>子标签，配置异常处理
  - <global-forwards> 包含多个<forward>子标签，配置转发路径
  - <action-mappings> 包含多个<action>子标签，配置请求路径与Action类之间的映射
  - <controller> 配置ActionServlet
  - <message-resources> 配置资源包
  - <plug-in> 配置Struts插件
- 注意标签的先后顺序，打乱顺序会导致错误

修改HelloAction

- 当用户请求HelloAction时，在控制台输出Hello,Action，并且转发到hello.jsp
- 需要创建/hello.jsp文件

配置一个转发路径<forward>标签

配置全局转发，针对于整个web应用的

```
<global-forwards>
    <forward name="hello" path="/hello.jsp"></forward>
</global-forwards>
```

或者配置局部转发，针对于当前Action的

```
<action path="/helloAction" type="mystruts.action.HelloAction">
    <forward name="hello" path="/hello.jsp"></forward>
</action>
```

- <forward>标签属性
  - name 自定义别名
  - path 别名对应的资源路径
- 如果同时配置两个name相同的全局和局部的<forward>，局部的优先级高于全局的
- 如果定义成局部的<forward>，那么别的Action就无法使用这个转发路径

修改HelloAction代码

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
                HttpServletRequest request, HttpServletResponse response)
                throws Exception {

        System.out.println("Hello,Action");

//      return new ActionForward(“/hello.jsp”);//直接转发目标资源

        return mapping.findForward(“hello”);//查找<forward>配置进行转发
}
```

运行修改后的Action