软件架构与中间件

涂志莹 tzy_hit@hit.edu.cn

苏统华 thsu@hit.edu.cn

哈尔滨工业大学

软件架构与中间件

Software Architecture and Middleware

第2章

软件架构的传统风格

课程内容


软件架构与中间件的内涵及要点
软件架构的传统风格
软件架构的评审、测试与优化
软件架构与中间件
计算层的软件架构技术
表示层的软件架构技术
数据层的软件架构技术

第2章 软件架构的传统风格

2.1 软件架构风格概述

2.2 主程序-子过程风格

2.3 面向对象风格

2.4 数据流风格

2.5 事件驱动风格

2.6 解释器风格

2.7 分层结构

2.8 模型-视图-控制器

2.9 本章作业

2.8 MVC风格

1、MVC风格概述

2、MVC实现框架

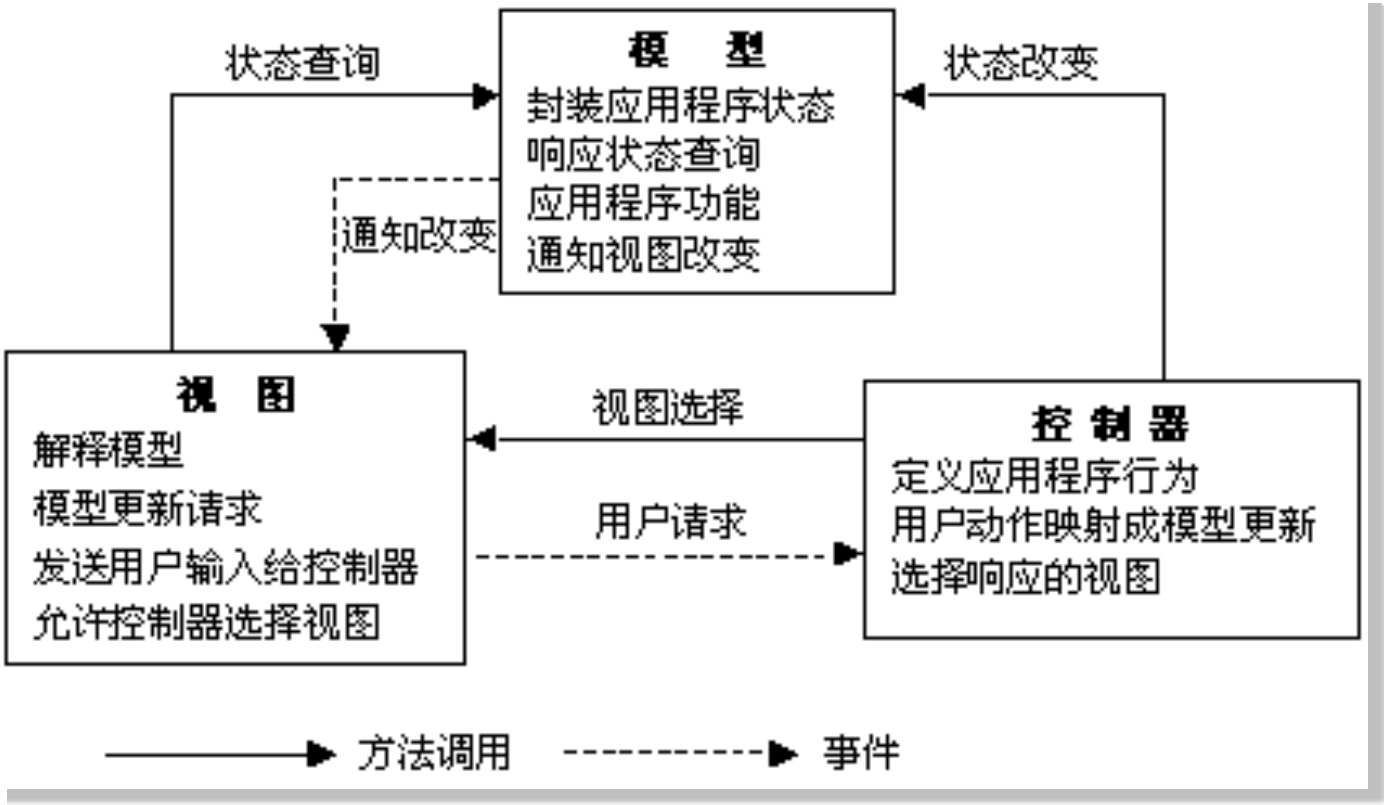
2.8.1 MVC风格概述

What is MVC?

- Model–View–Controller (MVC) splits interactions between users and applications into three roles: the Model (business logic), the View (user interface), and the Controller (user input). (关注点分离：模型、视图和控制器).
  - MVC was first described in 1979 by Trygve Reenskaug, then working on Smalltalk at Xerox PARC.
  - Since the late 1990s, MVC is commonly classified as an architectural pattern, i.e. a classic way of structuring software that is used in a software architecture in the modern sense of Shaw and Garlan.
  - The goal of MVC is to simplify the architecture by decoupling models and views, and to make source code more flexible and maintainable(从开发者的角度看，实现model与view的解耦).
  - Very popular, used extensively in Java and other languages.

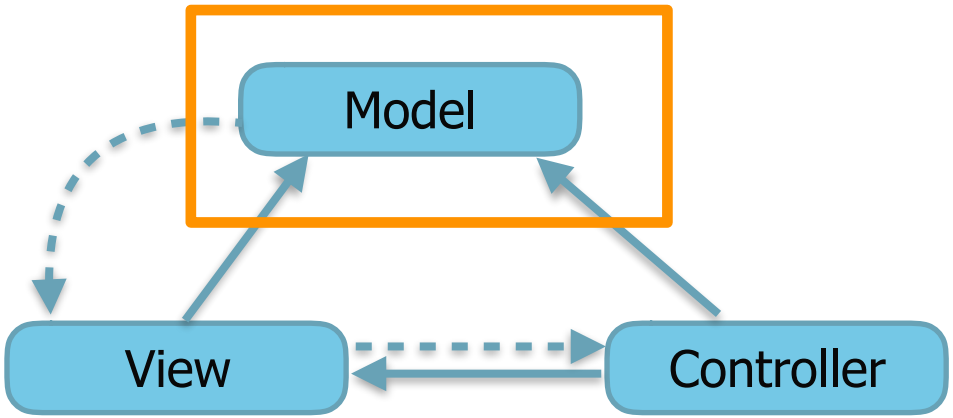
What is MVC?

- Components: Model, View , Controller
- Connectors: Explicit Invocation， Implicit Invocation or other mechanisms (e.g. HTTP protocol)



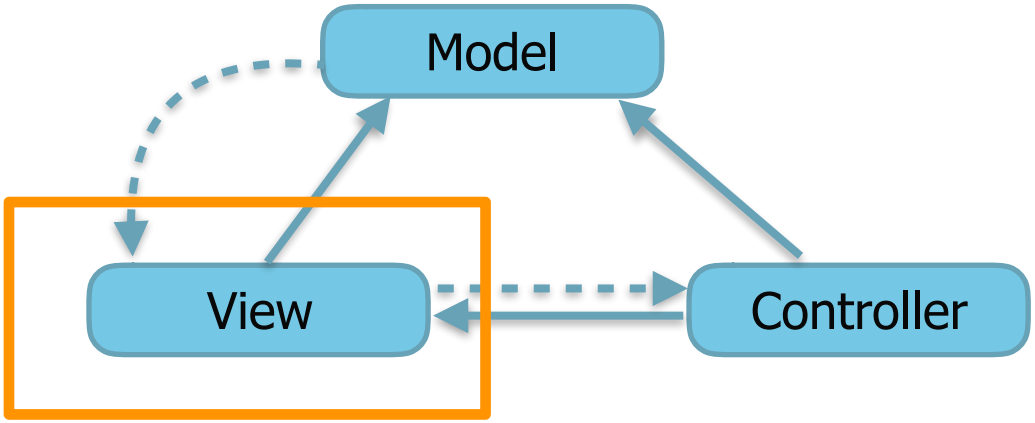
Model(模型)

- Model is responsible for
  - Providing the data from the database and saving the data into the data store(负责数据存取).
  - All the business logics are implemented in the Model(负责业务逻辑实现).
  - (Optional) Data entered by the user through View are checked in the model before saving into the database(可能负责数据验证).
- In event-driven systems, the model notifies observers (usually views) when the information changes so that they can react.

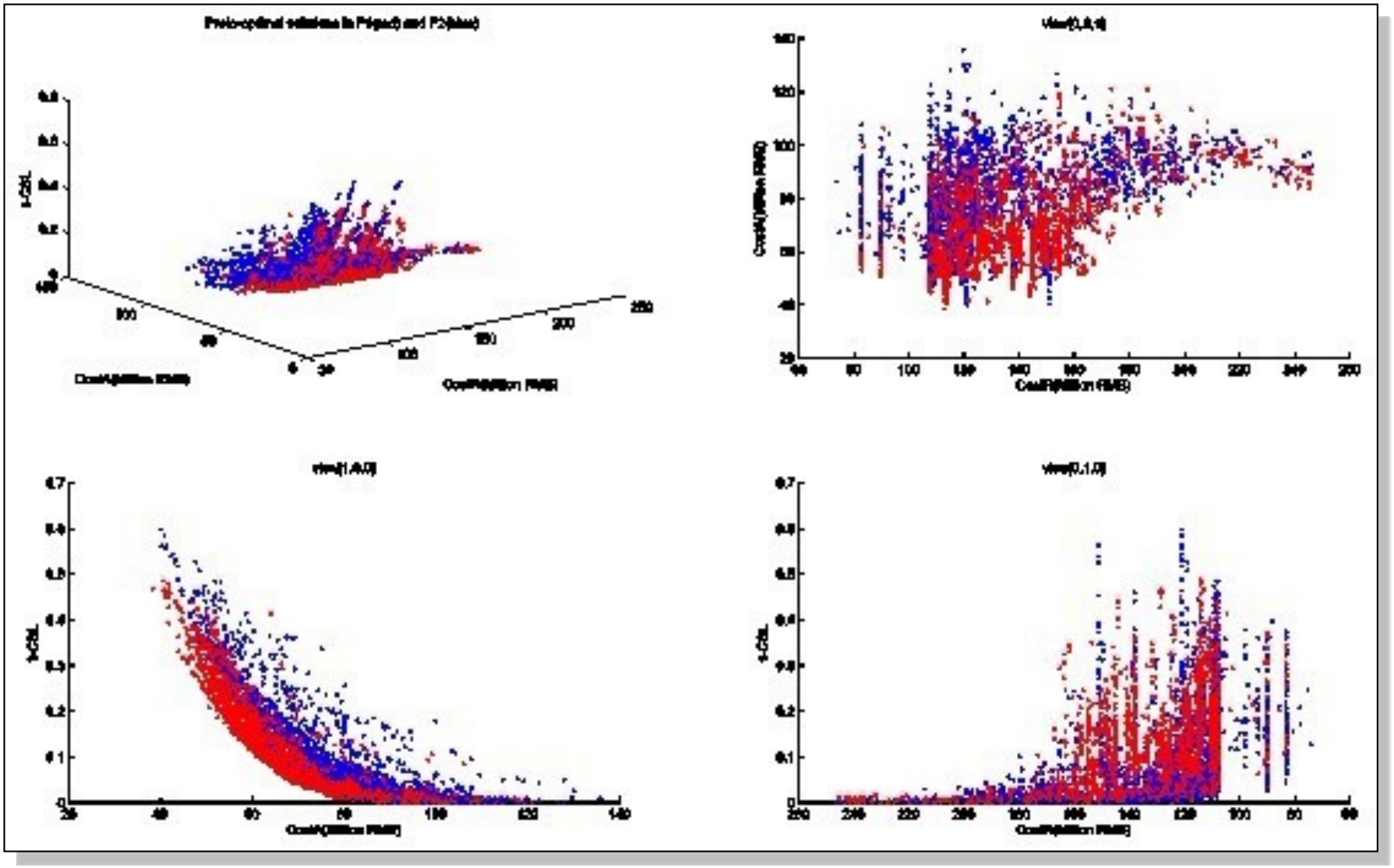


View(视图)

- View is responsible for:
  - Taking the input from the user (获取用户输入)
  - Dispatching the request to the controller(向controller发送处理请求)
  - Receiving response from the controller and displaying the result to the user.(接收来自Controller的反馈并将model的处理结果显示给用户)
- Multiple views can exist for a single model for different purposes.(一个model可能有多个View)

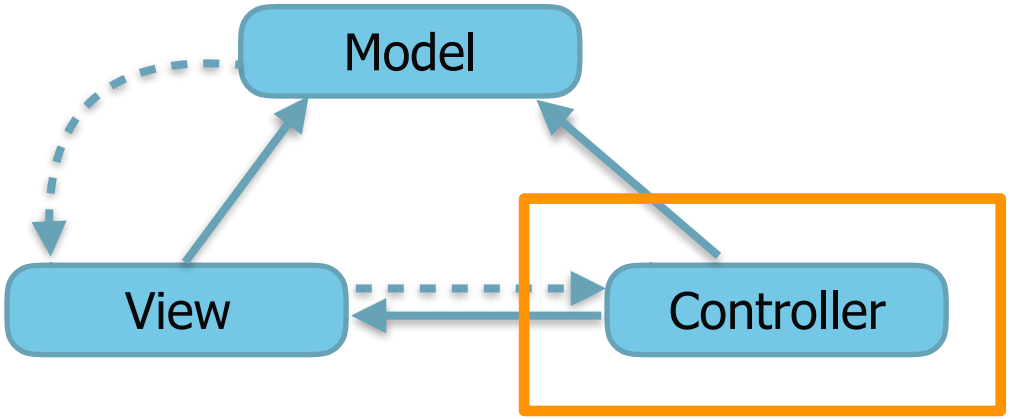


View(视图)



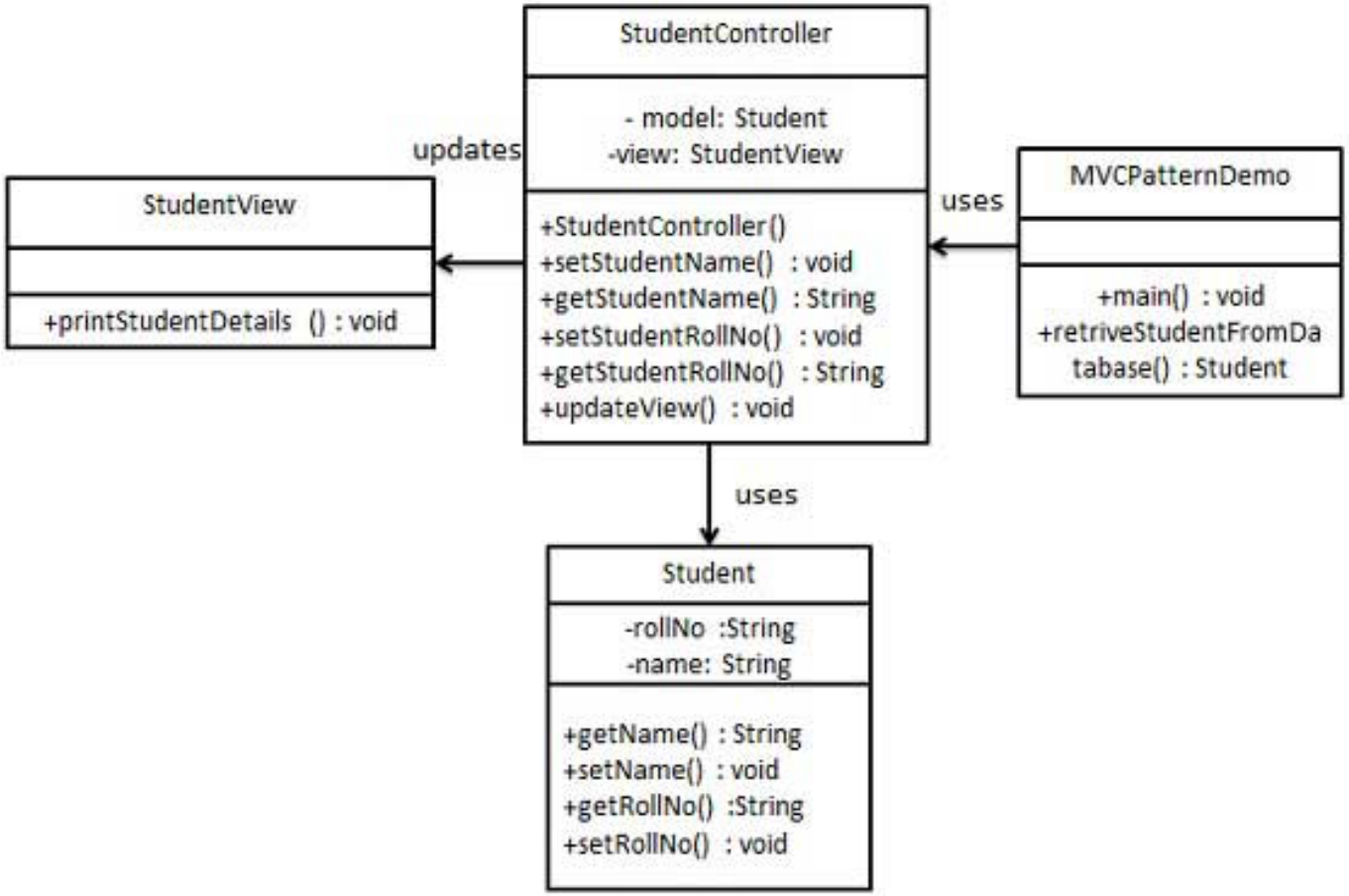
Controller

- Controller is responsible for:
  - Receiving the request from client.(接收来自客户的请求)
  - Executing the appropriate business logic from the Model (调用model执行业务逻辑)
  - Producing the output to the user using the View component. (调用View显示执行结果)



MVC使用示例

- 简化的学生管理系统



MVC使用示例

Student.java

```
public class Student {
   private String rollNo;
   private String name;
   public String getRollNo() {
      return rollNo;
   }
   public void setRollNo(String rollNo) {
      this.rollNo = rollNo;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
}
```

MVC使用示例

StudentView.java

```
public class StudentView {
   public void printStudentDetails(String
studentName, String studentRollNo){
      System.out.println("Student: ");
      System.out.println("Name: " + studentName);
      System.out.println("Roll No: " +
studentRollNo);
   }
}
```

MVC使用示例

StudentController.java

```
public class StudentController {
   private Student model;
   private StudentView view;

   public StudentController(Student model, StudentView view){
      this.model = model;
      this.view = view;
   }

   public void setStudentName(String name){
      model.setName(name);
   }

   public String getStudentName(){
      return model.getName();
   }

   public void setStudentRollNo(String rollNo){
      model.setRollNo(rollNo);
   }

   public String getStudentRollNo(){
      return model.getRollNo();
   }

   public void updateView(){
      view.printStudentDetails(model.getName(), model.getRollNo());
   }
}
```

MVC使用示例

MVCPatternDemo.java

```
public class MVCPatternDemo {
   public static void main(String[] args) {

      //从数据库获取学生记录
      Student model  = retriveStudentFromDatabase();

      //创建一个视图：把学生详细信息输出到控制台
      StudentView view = new StudentView();

      StudentController controller = new StudentController(model, view);

      controller.updateView();

      //更新模型数据
      controller.setStudentName("John");

      controller.updateView();
   }

   private static Student retriveStudentFromDatabase(){
      Student student = new Student();
      student.setName("Robert");
      student.setRollNo("10");
      return student;
   }
}
```

MVC使用示例

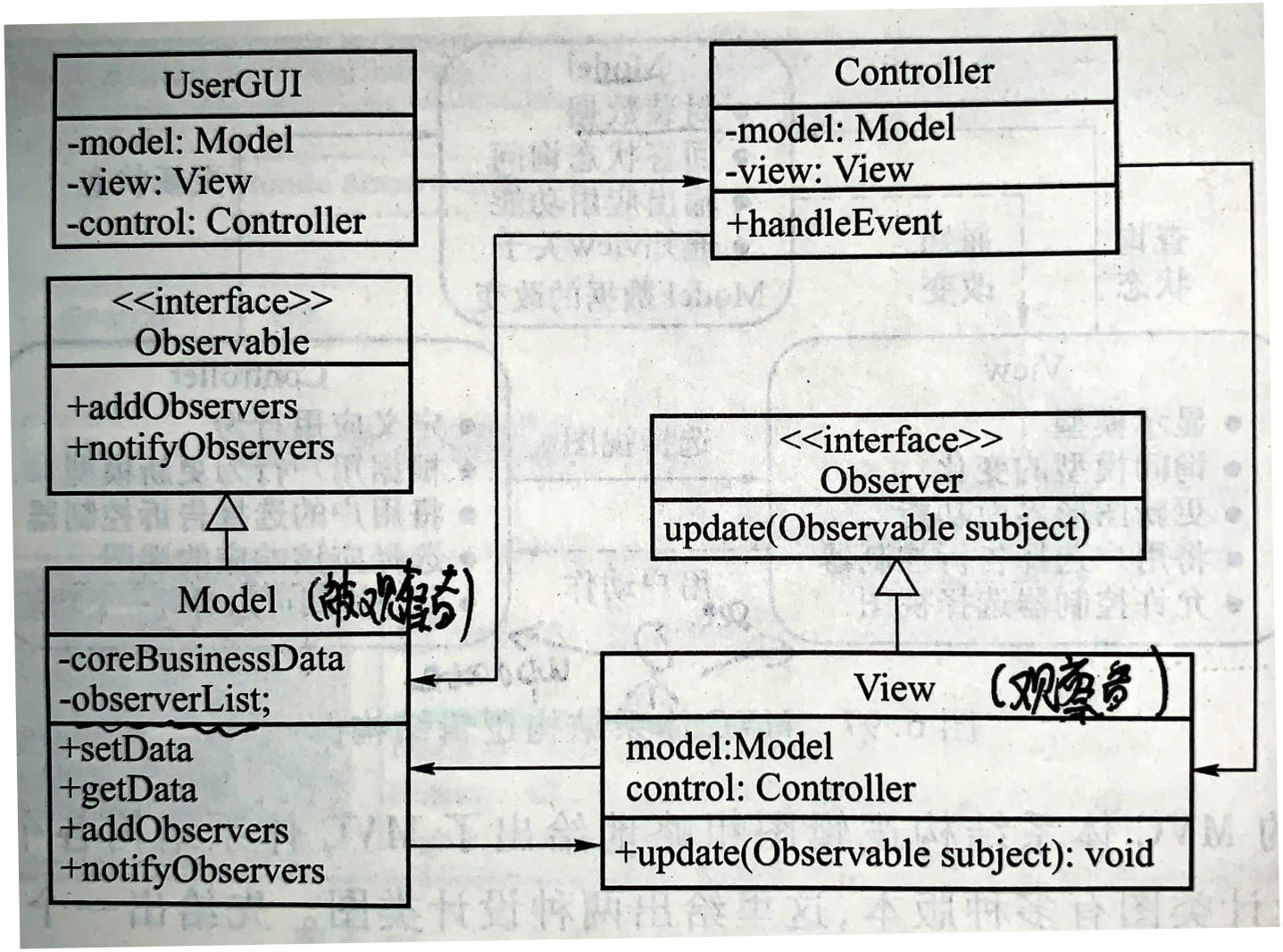
- 输出结果

执行程序，输出结果：

```
Student:
Name: Robert
Roll No: 10
Student:
Name: John
Roll No: 10
```

观测者模式的MVC

- MVC架构设计类图（使用观测者模式）



Advantages of MVC

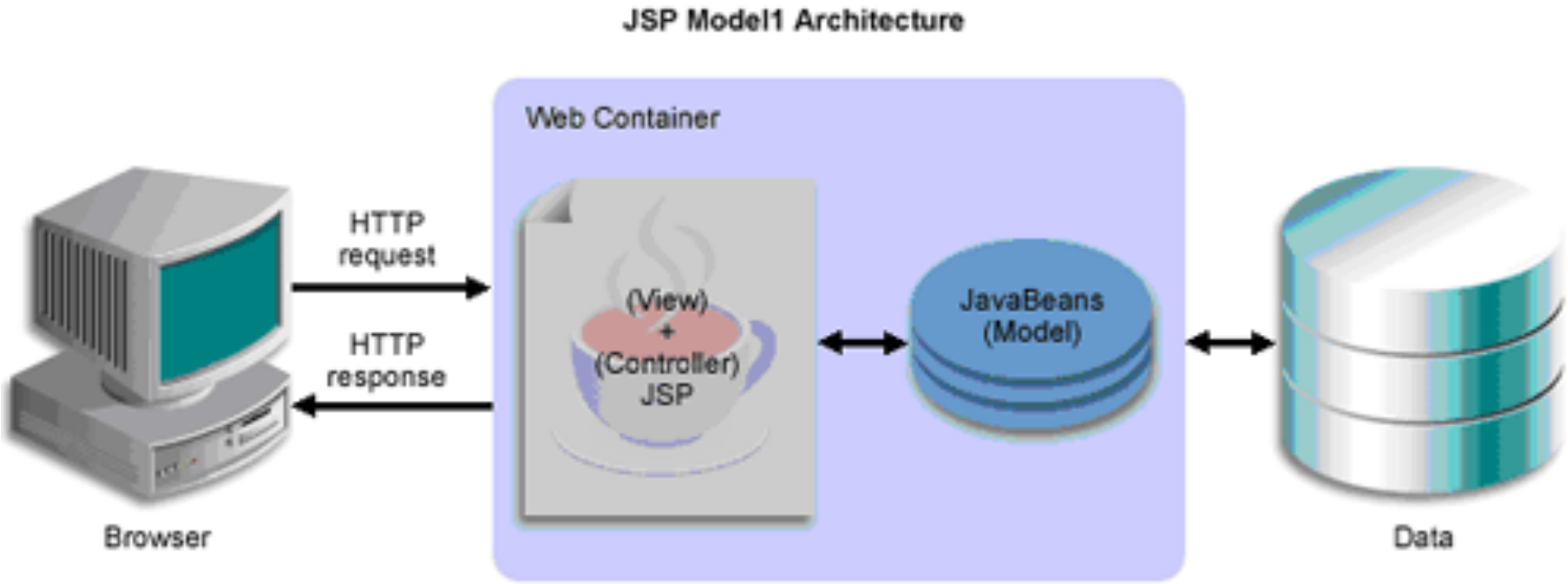
- Two principal separations:
  - Separating the presentation from the model展示与模型分离
  - Separating the controller from the view控制器与视图分离
- Advantages:
  - Make source code more flexible and maintainable.(代码易开发易维护)
  - Make it very easy to have multiple different displays of the same information(同一信息可以有不同的显示方式).
  - Make it very easy to test all the domain logic. Because Non-visual objects are usually easier to test than visual ones(业务逻辑更易测试).

2.8.2 MVC实现框架

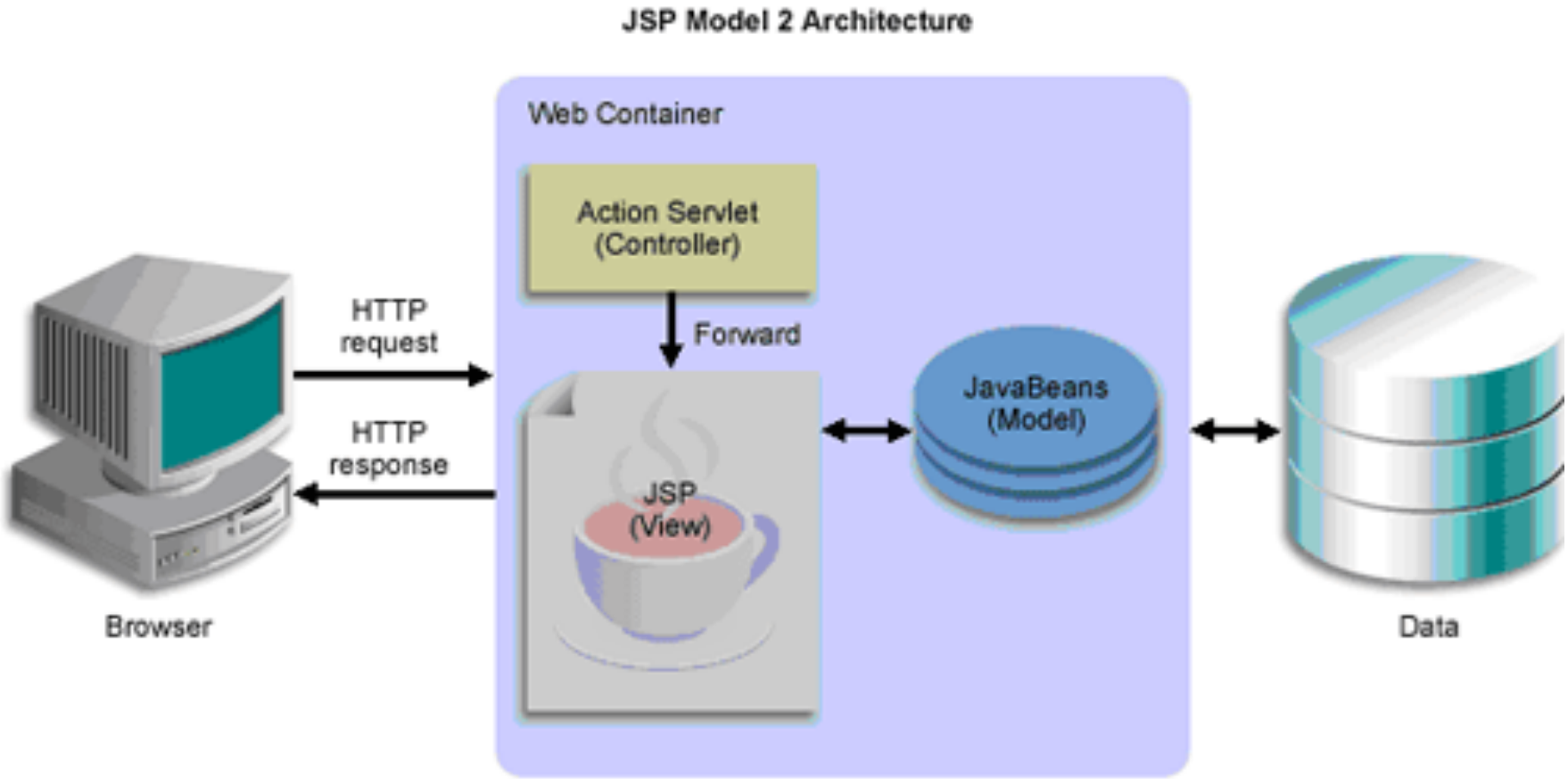
MVC Frameworks

- J2EE:
  - Struts MVC
  - Spring MVC
- PHP
  - CakePHP
  - Struts4php
- C#.NET
  - Girders
- Ruby on Rails

Java EE WEB Model 1:Non-MVC



Java EE WEB Model 2:MVC



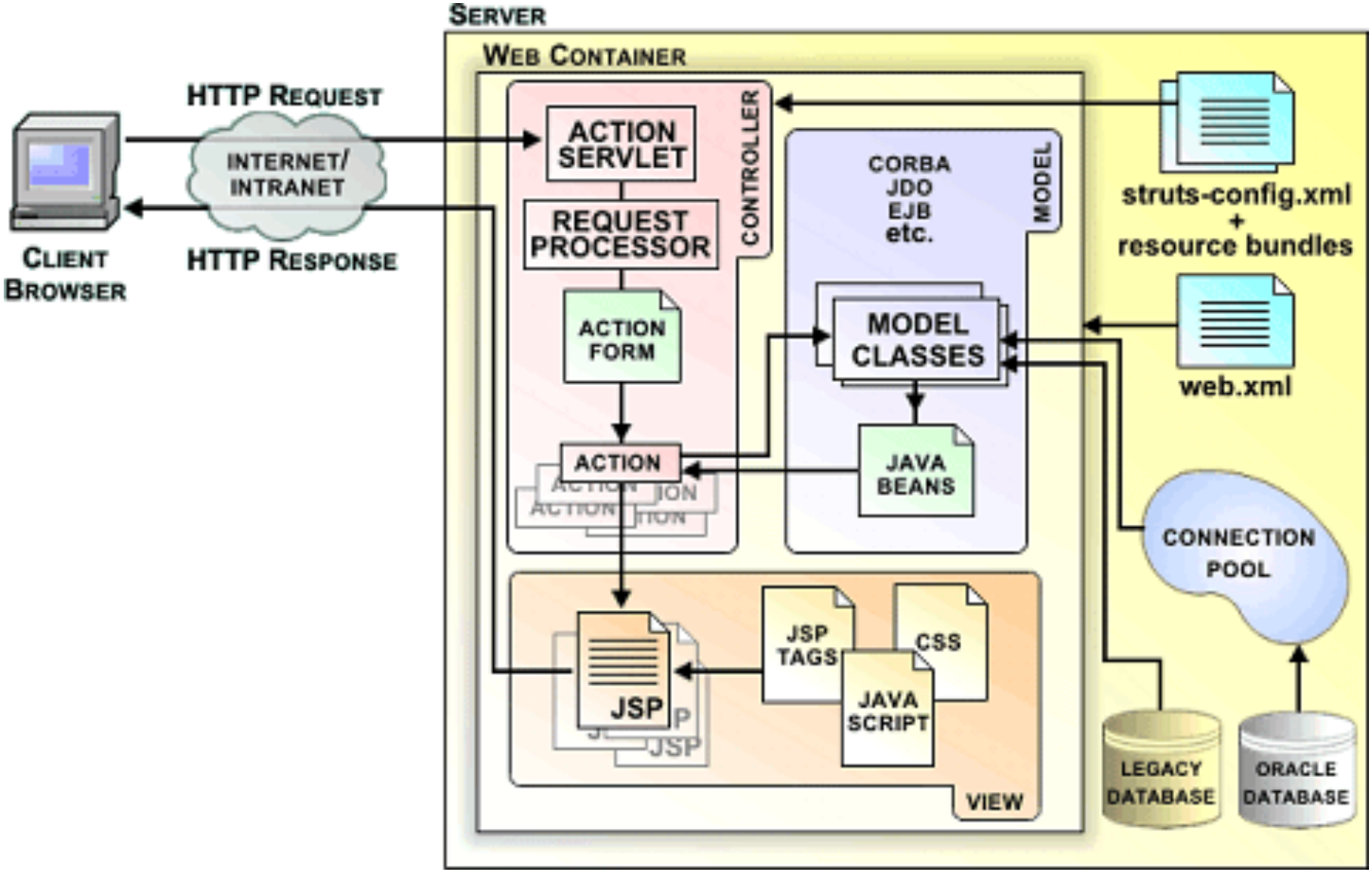
Java EE实现MVC技术基础

- Java Server Pages
  - Java Server Pages (JSP) is a technology that helps software developers serve dynamically generated web pages based on HTML, XML, or other document types. Released in 1999 by Sun Microsystems.
  - Architecturally, JSP may be viewed as a high-level abstraction of Java servlets. JSPs are translated into servlets at runtime; each JSP's servlet is cached and re-used until the original JSP is modified.
- Servlets
  - A Servlet is a java based server side web technology. A software developer may use a servlet to add dynamic content to a Web server. The generated content is commonly HTML, but may be other data such as XML. Released in June 1997 by Sun Microsystems.
  - Technically speaking a Servlet is a Java class in Java EE that conforms to the Java Servlet API, a protocol by which a Java class may respond to requests. .
- Java Beans

Struts MVC

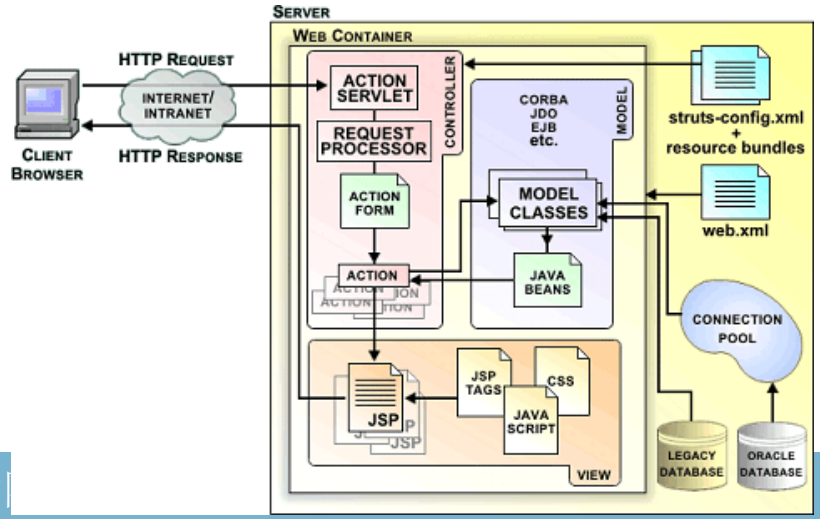
- Apache Struts is an open-source web application framework for developing Java EE web applications.
- It uses and extends the Java Servlet API to encourage developers to adopt a model-view-controller (MVC) architecture.
- It was originally created by Craig McClanahan and donated to the Apache Foundation in May, 2000.

Struts MVC


SERVER
WEB CONTAINER
HTTP REQUEST
INTERNET/
INTRANET
CLIENT
BROWSER
HTTP RESPONSE
ACTION
SERVLET
REQUEST
PROCESSOR
ACTION
FORM
ACTION
CONTROLLER
CORBA
JDO
EJB
etc.
MODEL
MODEL
CLASSES
JAVA
BEANS
struts-config.xml
+
resource bundles
web.xml
CONNECTION
POOL
JSP
JSP
TAGS
CSS
JAVA
SCRIPT
VIEW
LEGACY
DATABASE
ORACLE
DATABASE

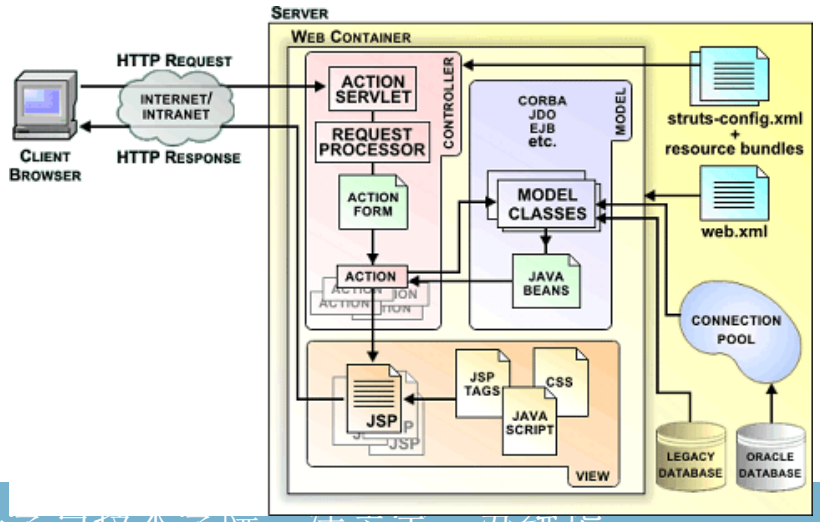
Components of Struts

- struts-config.xml　　Struts配置文件
- ActionServlet　控制器（Struts 的核心模块）
- Action Class　　包含事务逻辑
- ActionForm　　显示模块数据
- ActionMapping　　帮助控制器将请求映射到操作
- ActionForward　　用来指示操作转移的对象
- ActionError　　用来存储和回收错误
- JSP　　用于构造界面
- Struts Tags　　JSP标记库



ActionServlet

- 截获用户的Http请求(含Form数据),把这个请求映射到相应的Action类
- 创建一个ActionForm bean实例,将Form数据放入ActionForm中
- 调用Action实例的perform()方法
- 将ActioForm bean,Action Mapping对象,request和response对象传给Action的execute()方法
- Action的execute方法返回一个ActionForword对象，根据方法执行情况转到相应的jsp页面



How to build a simple Struts application?

1. Update the web.xml file(optional)
2. Create the struts-config.xml file
3. Build the View Components (JSPs)
4. Build the Model Components (ActionForm Class)
5. Build the Controller Components (Action Class)

Update the web.xml file

- Config the ActionServlet and the URL Pattern

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
</servlet>

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

Create the struts-config.xml file

- Within this file you' ll define the action-mappings and form-beans.(配置action-mappings 和 form-beans)
  - The action-mappings tell the ActionServlet which action classes to include within the application.
  - The form-beans define the ActionForm classes required within the application.

```
<action-mappings type="org.apache.struts.action.ActionMapping">
    <action path="/simpleAction"
        type="edu.depaul.struts.SimpleAction"
        name="simpleBean"
        scope="request"
</action-mappings>
```

```
<form-beans type="org.apache.struts.action.ActionFormBean">

    <form-bean name="simpleBean" type="edu.depaul.struts.SimpleForm"/>

</form-beans>
```

Building a JSP

- Must include a form in the jsp file (必须包含一个Form)
- To create a JSP using Struts you may use the tag libraries provided by the Struts Framework (可以使用taglib构造界面).
  - struts-bean taglib：包含在访问bean和bean属性时使用的tag，也包含一些消息显示的tag。

```
<html:form action="simpleAction" method="POST">

First Name:
<html:text property="firstName"/>
<font color="red"><html:errors property="firstName"/>
</font>

Last Name:
<html:text property="lastName"/>
<font color="red"><html:errors property="lastName"/>
</font>
<html:submit value="Submit"/>

</html:form>
```

Building an ActionForm

- The ActionForm is a JavaBean which contains the data entered trough an HTML form (是一个JavaBean,存储HTML Form中的数据).
- The ActionForm should also contain the get & set methods for each property.
- To create a ActionForm, you will need to extend the class org.a… …endent on t…
- The … …d to perf… …ing the …

```
public class SimpleForm extends ActionForm
{
   private String firstName;
   private String lastName;

   public String getFirstName()
   {
      return firstName;
   }

            ... other getters and setters
}
```

Building an Action Class

- Action is a bridge to connect the ActionServlet and the

```
public class SimpleForm extends Action{
…
public ActionForward execute(
            ActionMapping actionMapping,
            ActionForm actionForm,
            HttpServletRequest httpServletRequest,
            HttpServletResponse
            httpServletResponse) throws Exception{
          //getting posted information
            SimpleForm simpleForm = (SimpleForm) actionForm;

            ///... - business – related processing (调用相关的业务逻辑处理)

            httpServletRequest.setAttribute("simpleForm", simpleForm);
            //passing control to the next page
            return actionMapping.findForward("next");
          }
}
```

2.9
本章作业

作业2

- 针对KWIC问题，我们要提供一个全球用户可用的Web应用，请给出基于MVC风格的设计方案
  - 用户可以从多种终端访问应用，e.g. 手机端、PC端
  - 可以通过网络上传待处理文件，文件可能超过10000行
  - 设计方案应给出构件、连接件的设计细节并讨论设计的理由，比如构件的功能规格是什么，构件有哪些属性、方法，连接件如何保证稳定性等
  - 提交时间：+2周

“请你来讲”征集

- 主题：
  - MVC架构的演进：过去、现在与未来
- 要求：
  - 2周后课上分享给大家
  - 时长15分钟到30分钟
  - 内容上包括经典MVC、应用模型MVC、MVP、Web MVC、Web MVP等以及其他大家感兴趣的“故事”
- 资料：
  - eg http://huoding.com/2011/05/02/64

第2章

软件架构的传统风格

Thanks for listening

涂志莹、苏统华
哈尔滨工业大学计算机学院
企业与服务计算研究中心