SpringMVC

Programming
Your Future

-----控制层顶级框架

Beta版

本章内容

Programming Your Future

节	知识点	掌握程度	难易程度
SpringMVC概述	什么是SpringMVC	了解	普通
	MVC在B/S系统下的应用	了解	普通
	SpingMVC框架	理解	难
	SpingMVC组件	掌握	普通
入门程序	环境搭建	掌握	普通
	开发步骤	掌握	普通

SpringMVC概述

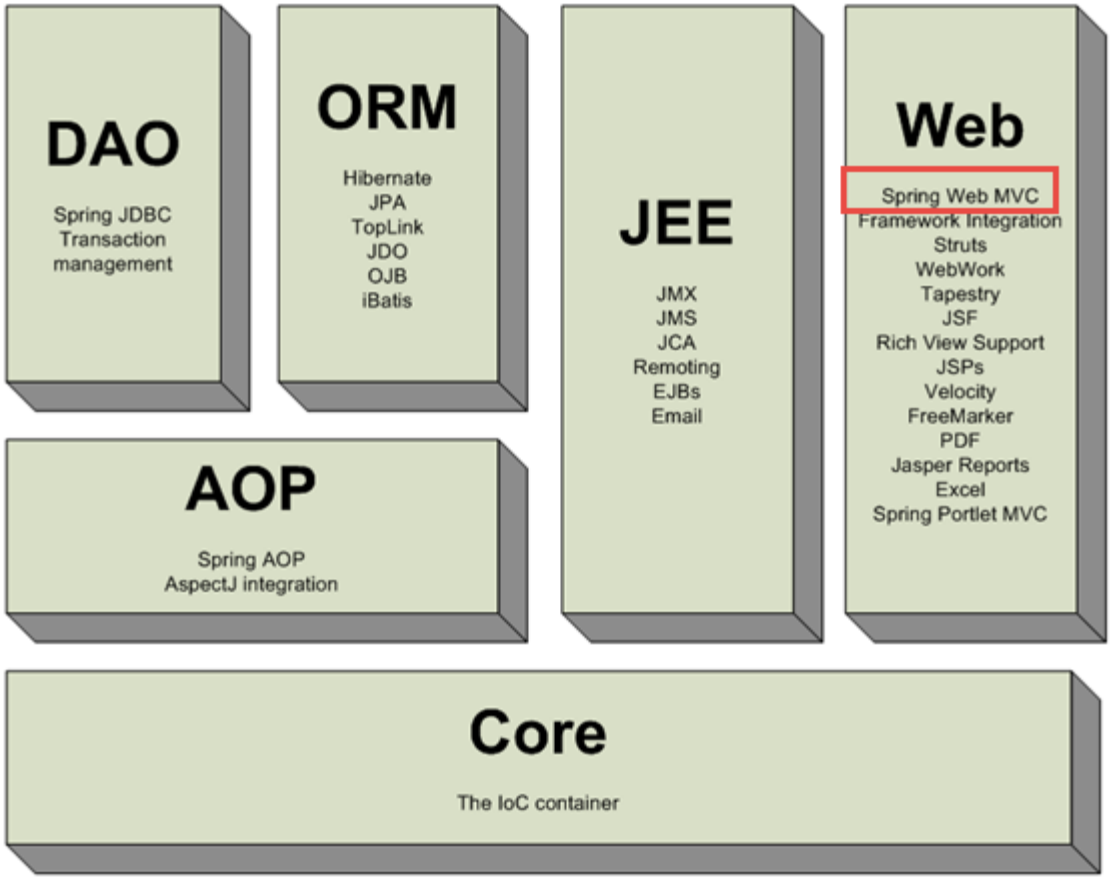
Programming Your Future

- 什么是SpringMVC
  - springmvc是spring框架的一个模块，springmvc和spring无需通过中间整合层进行整合。
  - springmvc是一个基于mvc的web框架。

SpringMVC概述

Programming Your Future

- 什么是SpringMVC



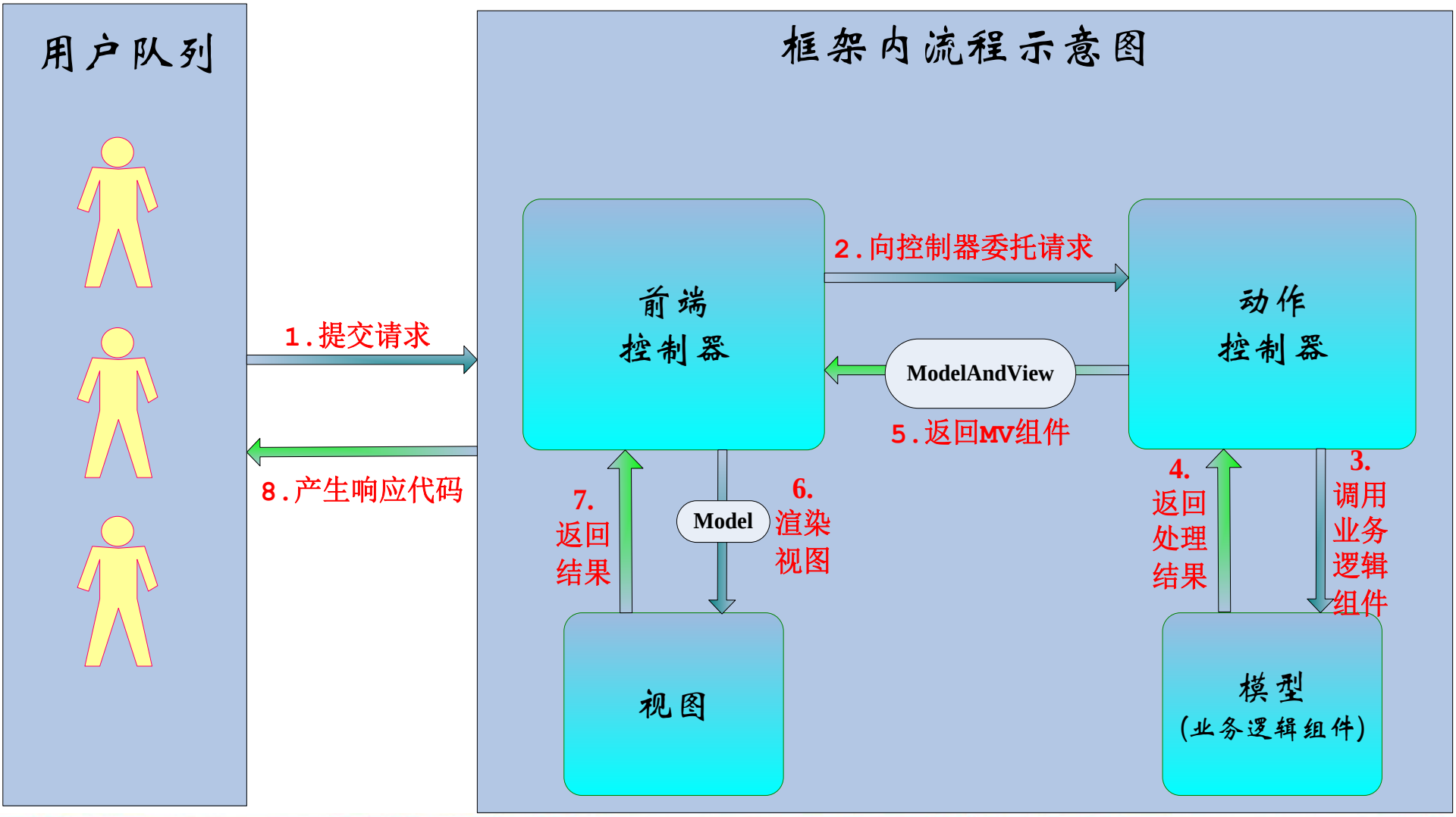
SpringMVC概述

- MVC在B/S系统下的应用
  - **用户发起request请求至控制器(Controller)**
    - 控制接收用户请求的数据，委托给模型进行处理
  - **控制器通过模型(Model)处理数据并得到处理结果**
    - 模型通常是指业务逻辑
  - **模型处理结果返回给控制器**
  - **控制器将模型数据在视图(View)中展示**
    - web中模型无法将数据直接在视图上显示，需要通过控制器完成。如果在C/S应用中模型是可以将数据在视图中展示的。
  - **控制器将视图response响应给用户**
    - 通过视图展示数据或提示信息给客户

SpringMVC概述

- SpringMVC框架
  - Spring 为展现层提供的基于 MVC 设计理念的优秀的Web 框架，是目前最主流的 MVC 框架之一
  - Spring3.0 后全面超越 Struts2，成为最优秀的 MVC 框架
  - Spring MVC 通过一套 MVC 注解，让 POJO 成为处理请求的控制器，而无须实现任何接口
  - 采用了松散耦合可插拔组件结构，比其他 MVC 框架更具扩展性和灵活性

SpringMVC概述--工作原理



SpringMVC示例程序

Programming Your Future

- 环境搭建
  - java环境：
    - jdk1.7
    - Eclipse Java EE IDE
  - springmvc版本：
    - spring3.2
  - Web服务器：
    - tomcat7
  - 数据库环境：
    - Oracle11gR2

SpringMVC示例程序

Programming Your Future

- 版本1
  - 基于Controller接口的原始实现
  - 基于XML文件配置

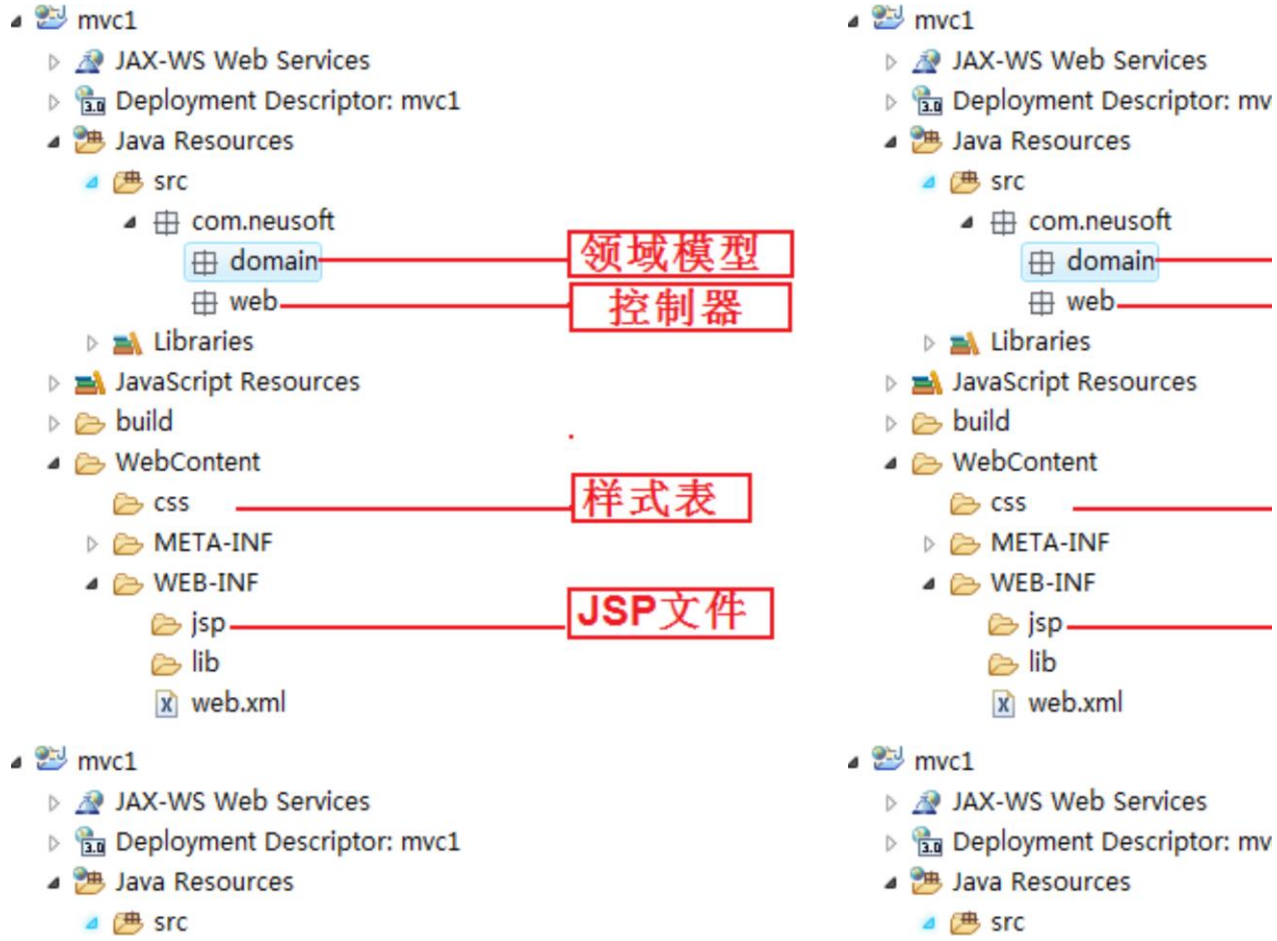
SpringMVC示例程序-版本1

Programming Your Future

- 开发步骤总纲：
  - 创建工程及相关包路径
  - 导入SpringMVC相关jar包
  - 在 web.xml 中配置 DispatcherServlet
  - 编写控制器实现类及初始页面
  - 完成springmvc-servlet.xml文件配置过程
  - 编写领域模型Person类
  - 完成人员添加控制器及显示页面
  - 部署运行

SpringMVC示例程序-版本1

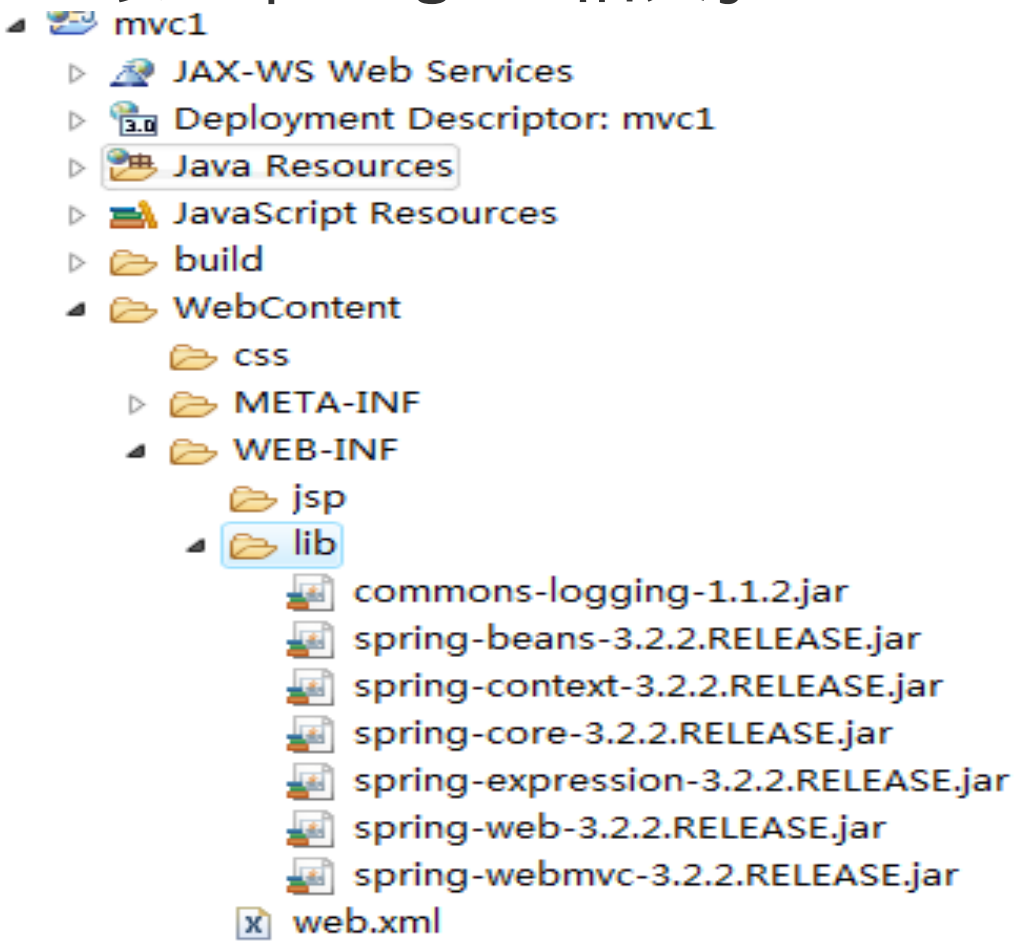
- 创建工程及相关包路径,工程结构如下图：



SpringMVC示例程序-版本1

Programming Your Future

- 导入SpringMVC相关jar包：



右键单击获取**jar**包

SpringMVC基础Jar包.rar

SpringMVC示例程序-版本1

Programming Your Future

- 配置Web.xml：
  - 在 web.xml 中配置 DispatcherServlet

```
<servlet>
    <servlet-name>springmvc</servlet-name>
    <servlet-class>
        org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>springmvc</servlet-name>
    <!-- 所有控制器的默认扩展名,相当于Struts2的.action -->
    <url-pattern>*.html</url-pattern>
</servlet-mapping>
```

web.xml

SpringMVC示例程序-版本1

- 编写控制器实现类及初始页面：
  - 编写控制器实现类
  - 控制器类需要实现如下接口
    - org.springframework.web.servlet.mvc.Controller
  - 初始页面

SpringMVC示例程序-版本1

Programming Your Future

- 初始化控制器实现类代码:

```
public class A1010Controller implements Controller
{
    @Override
    public ModelAndView handleRequest(HttpServletRequest request,
            HttpServletResponse response) throws Exception
    {
        //ModelAndView通过模型视图组件,制定默认打开的页面
        ModelAndView mv=new  ModelAndView("/WEB-INF/jsp/A1010.jsp");
        return mv;
    }

}
```

SpringMVC示例程序-版本1

- 完成springmvc-servlet.xml文件配置过程：
  - 该文件需要放置到工程的/WEB-INF下，
  - 文件名必须是springmvc-servlet.xml
  - 代码如下图：

springmvc-servlet.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xmlns:p="http://www.springframework.org/schema/p"
5      xsi:schemaLocation="http://www.springframework.org/schema/beans
6      http://www.springframework.org/schema/beans/spring-beans-3.2.xsd">
7
8      <!-- 配置控制器 -->
9      <bean name="/a1010.html" class="com.neusoft.web.A1010Controller"/>
10
11 </beans>
12
```

空文件下载
右侧图标

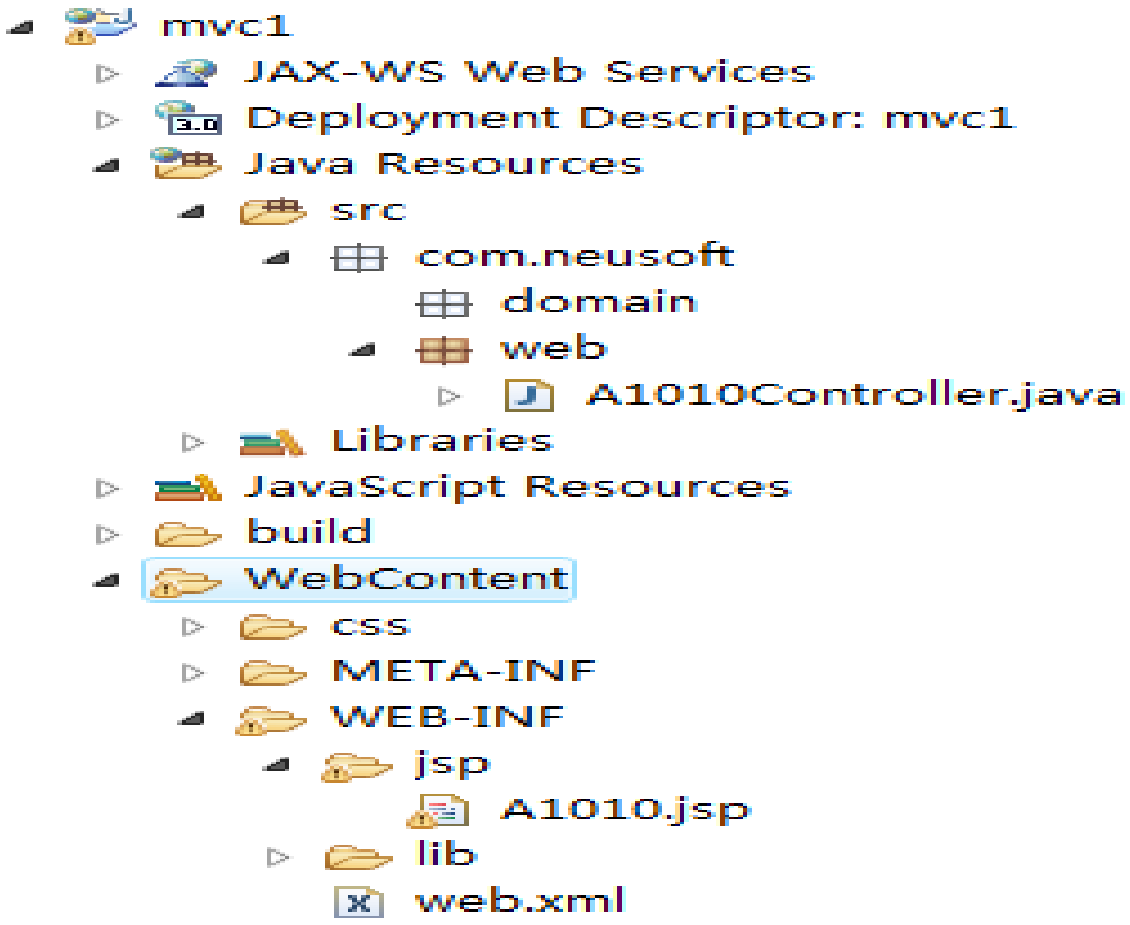
springmvc-servlet.xml

初始页面A1010.jsp代码

```jsp
<%@ page language="java" pageEncoding="GBK"%>
<html>
<head>
  <title>Insert title here</title>
  <style type="text/css">@IMPORT url("css/style.css");</style>
</head>
<body>
<form action="<%=request.getContextPath()%>/a1011.html" method="post">
  <div class="edit">
    <table>
      <caption>
        员工管理
        <hr>
      </caption>
      <tr class="title">
        <td colspan="100" >员工信息</td>
      </tr>
      <tr>
        <td>姓名</td>
        <td> <input type="text" name="pname">  </td>
      </tr>
      <tr>
        <td>身份证</td>
        <td>  <input type="text" name="pnumber">   </td>
      </tr>
      <tr>
        <td>月薪</td>
        <td>  <input type="number" step="0.01" name="pmoney">    </td>
      </tr>
      <tr class="title">
        <td colspan="100" align="center">
          <input type="submit" name="next" value="提交">
        </td>
      </tr>
    </table>
  </div>
</form>
</body>
</html>
```

SpringMVC示例程序-版本1

- 编写完毕A1010.jsp后工程结构图



SpringMVC示例程序-版本1

- 部署运行测试：

http://localhost:8080/mvc1/a1010.html

运行效果图

SpringMVC示例程序-版本1

- 编写领域模型Person类：
  - 创建工程及相关包路径
  - 该类需要实现序列化接口
  - 代码截图见下页

领域模型Person类代码

Programming Your Future

```
1  package com.neusoft.domain;
2
3  public class Person implements java.io.Serializable
4  {
5      private String pname=null;
6      private String pnumber=null;
7      private String pmoney=null;
8
9      // 设置子与访问子略,懒鬼们需要自己补充上
10     @Override
11     public String toString()
12     {
13         StringBuilder text=new StringBuilder()
14         .append("Person[")
15         .append("pname="+this.pname)
16         .append(",pnumber="+this.pnumber)
17         .append(",pmoney="+this.pmoney)
18         .append("]")
19         ;
20         return text.toString();
21     }
22 }
```

SpringMVC示例程序-版本1

Programming Your Future

- 员工添加动作点控制器实现类代码：

```
1  package com.neusoft.web;
2
3  import javax.servlet.http.HttpServletRequest;
8
9  public class A1011Controller implements Controller
10 {
11     @Override
12     public ModelAndView handleRequest(HttpServletRequest request,
13                         HttpServletResponse response) throws Exception
14     {
15         Person p=new Person();
16         p.setPname(request.getParameter("pname"));
17         p.setPnumber(request.getParameter("pnumber"));
18         p.setPmoney(request.getParameter("pmoney"));
19         ModelAndView mv=new ModelAndView("/WEB-INF/jsp/A1011.jsp","person",p);
20         return mv;
21     }
22 }
```

知识点:

- ModelAndView:
  - 负责将数据模型与目标视图建立绑定关系
- 参数格式:

```
new ModelAndView(String viewName,
                 String modelName,
                 Object modelObject
                 )
```

- 参数说明
  - String viewName,　　目标视图及路径
  - String modelName,　　绑定的模型名称
  - Object modelObject　　模型对象
- 用法概述:
  - 通过模型名称,将模型对象与目标页面绑定

SpringMVC示例程序-版本1

- 配置A1011Controller控制器

A1011.jsp | springmvc-servlet.xml | A1011Controller.java

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xmlns:p="http://www.springframework.org/schema/p"
5      xsi:schemaLocation="http://www.springframework.org/schema/beans
6      http://www.springframework.org/schema/beans/spring-beans-3.2.xsd">
7
8      <!-- 配置控制器 -->
9      <bean name="/a1010.html" class="com.neusoft.web.A1010Controller"/>
10     <!-- 员工添加控制器 -->
11     <bean name="/a1011.html" class="com.neusoft.web.A1011Controller"/>
12
13 </beans>
```

SpringMVC示例程序-版本1

- 员工信息显示页面

```jsp
<%@ page language="java" pageEncoding="GBK"%>
<html>
<head>
  <title>A1011.jsp</title>
  <style type="text/css">@IMPORT url("css/style.css");</style>
</head>
<body>
  <div class="edit">
    <table>
      <caption>
        员工信息
        <hr>
      </caption>
      <tr class="title">
        <td colspan="100" >员工信息</td>
      </tr>
      <tr>
        <td width="35%">姓名</td>
        <td width="65%"> ${person.pname }  </td>
      </tr>
      <tr>
        <td>身份证</td>
        <td> ${person.pnumber }  </td>
      </tr>
      <tr>
        <td>月薪</td>
        <td> ${person.pmoney }  </td>
      </tr>
    </table>
  </div>
</form>
</body>
</html>
```

SpringMVC示例程序-版本1

- 版本1部署运行：

工程完整代码

mvc1.rar

版本1的缺陷

- 1.原始ServletAPI的侵入性
- 2.一个控制器只能对应页面一个动作
- 3.XML文件配置极其相似,但啰嗦,重复,无聊
- 4.ModelAndView目标路径过长

- 对于第四个问题我们将在版本2中通过视图解析器予以优化
- 前三个问题我们将在版本3中给出解决方案

SpringMVC示例程序-版本2

- 为版本一配置视图解析器，简化ModelAndView的视图路径.
- 将如下代码复制到版本1的springmvc-servlet.xml中：

```
<!-- 视图解析器配置 -->
<bean id="viewResolver"
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
   <property name="prefix" value="/WEB-INF/jsp/"/>
   <property name="suffix" value=".jsp"/>
</bean>
```

- **参数说明：**
  - prefix -- 视图文件的存储路径
  - suffix -- 视图的扩展名
- **视图路径转换：**

```
//视图解析器配置前
ModelAndView mv=new ModelAndView("/WEB-INF/jsp/A1011.jsp","person",p);
//视图解析器配置后
ModelAndView mv=new ModelAndView("A1011","person",p);
```

SpringMVC示例程序-版本2

- 视图解析器配置完毕后的springmvc-servlet.xml

springmvc-servlet.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.2.xsd">
    <bean id="viewResolver"
       class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/jsp/"/>
        <property name="suffix" value=".jsp"/>
    </bean>

    <!-- 配置控制器 -->
    <bean name="/a1010.html" class="com.neusoft.web.A1010Controller"/>
    <!-- 员工添加控制器 -->
    <bean name="/a1011.html" class="com.neusoft.web.A1011Controller"/>
</beans>
```

SpringMVC示例程序-版本2

- **修改原有控制器的代码**
  - 配置完毕视图解析器以后，控制器中在指定视图路径时候，将很大程度上得以简化
- **代码参见后继两页PPT**

SpringMVC示例程序-版本2

- 初始化控制实现类代码

```
1  package com.neusoft.web;
2
3  import javax.servlet.http.HttpServletRequest;
8
9  public class A1010Controller implements Controller
10 {
11     @Override
12     public ModelAndView handleRequest(HttpServletRequest request,
13             HttpServletResponse response) throws Exception
14     {
15         //ModelAndView通过模型视图组件,制定默认打开的页面
16         ModelAndView mv=new  ModelAndView("A1010");
17         return mv;
18     }
19
20 }
```

SpringMVC示例程序-版本2

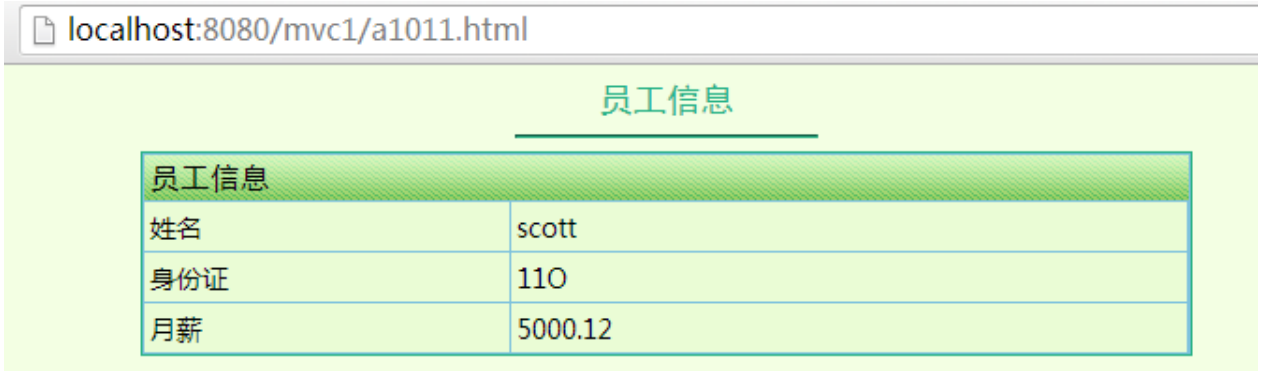
- 员工添加动作点控制器实现类代码：

A1011Controller.java

```
1  package com.neusoft.web;
2
3  import javax.servlet.http.HttpServletRequest;
8
9  public class A1011Controller implements Controller
10 {
11     @Override
12     public ModelAndView handleRequest(HttpServletRequest request,
13                         HttpServletResponse response) throws Exception
14     {
15         Person p=new Person();
16         p.setPname(request.getParameter("pname"));
17         p.setPnumber(request.getParameter("pnumber"));
18         p.setPmoney(request.getParameter("pmoney"));
19         //视图解析器配置后
20         ModelAndView mv=new ModelAndView("A1011","person",p);
21         return mv;
22     }
23 }
```

SpringMVC示例程序-版本2

- 版本2部署运行：



工程完整代码

回顾版本1的缺陷

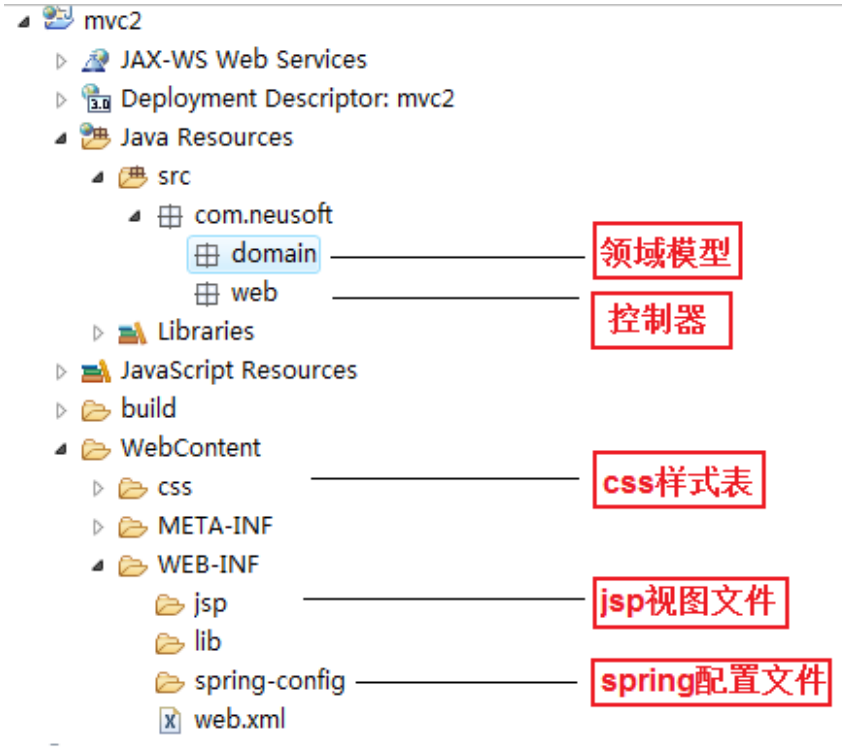
Programming Your Future

- 1.原始ServletAPI的侵入性
- 2.一个控制器只能对应页面一个动作
- 3.XML文件配置繁琐
- 4.ModelAndView目标路径过长

- 在版本2中我们已经解决了第四个问题,下面我们来解决前三个问题,工程进入到版本3
  - 基于元数据注解的配置方式

SpringMVC示例程序-版本3

- 创建工程mvc2,工程结构如下所示:



- 复制上一工程的jar包,领域模型,JSP及样式表当本工程,不赘述

SpringMVC示例程序-版本3

Programming Your Future

- 编写控制器,完成初始化页面调用代码如下

```
1  package com.neusoft.web;
2
3  import org.springframework.stereotype.Controller;
4  import org.springframework.web.bind.annotation.RequestMapping;
5
6  //该注解表示该类是SpringMVC的控制器类,无需实现任何接口
7  @Controller
8  public class PersonController
9  {
10     //该方法与请求路径 /a1010.html 对应
11     @RequestMapping(value="/a1010.html")
12     public String openWindow()
13     {
14         //返回A1010.jsp,路径为基于视图解析器简化
15         return "A1010";
16     }
17 }
```

SpringMVC示例程序-版本3

- 编写Spring配置文件,
  - 文件名为spring-config.xml
  - 该文件位于/WEB-INF/springConfig目录下,代码如下

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xmlns:p="http://www.springframework.org/schema/p"
5      xmlns:mvc="http://www.springframework.org/schema/mvc"
6      xmlns:context="http://www.springframework.org/schema/context"
7      xsi:schemaLocation="
8          http://www.springframework.org/schema/beans
9          http://www.springframework.org/schema/beans/spring-beans.xsd
10         http://www.springframework.org/schema/mvc
11         http://www.springframework.org/schema/mvc/spring-mvc.xsd
12         http://www.springframework.org/schema/context
13         http://www.springframework.org/schema/context/spring-context.xsd">
14
15     <!-- 指定 注解映射器扫描的控制器基础包,该包的所有子包都会被扫描 -->
16     <context:component-scan base-package="com.neusoft.web"/>
17     <!-- 注解映射器 -->
18     <mvc:annotation-driven/>
19     <!-- 无需过滤的静态资源,该部分内容,不需要DispatcherServlet 进行拦截 -->
20     <mvc:resources mapping="/css/**" location="/css/"/>
21
22
23     <bean id="viewResolver"
24     class="org.springframework.web.servlet.view.InternalResourceViewResolver">
25         <property name="prefix" value="/WEB-INF/jsp/"/>
26         <property name="suffix" value=".jsp"/>
27     </bean>
28 </beans>
```

SpringMVC示例程序-版本3

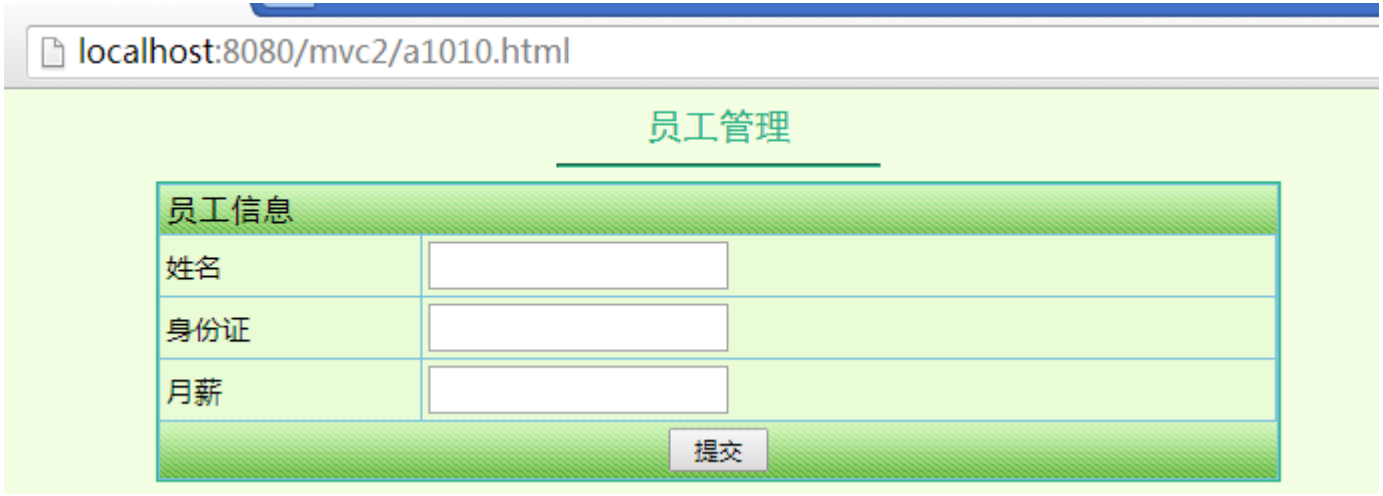
- 编写Web.xml增加对Spring的支持,代码如下

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="3.0"
3          xmlns="http://java.sun.com/xml/ns/javaee"
4          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5          xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
6          http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
7      <servlet>
8          <servlet-name>springmvc</servlet-name>
9          <servlet-class>
10             org.springframework.web.servlet.DispatcherServlet
11         </servlet-class>
12         <init-param>
13             <param-name>contextConfigLocation</param-name>
14             <param-value>/WEB-INF/springConfig/spring-config.xml</param-value>
15         </init-param>
16         <load-on-startup>1</load-on-startup>
17     </servlet>
18
19     <servlet-mapping>
20         <servlet-name>springmvc</servlet-name>
21         <url-pattern>/</url-pattern>
22     </servlet-mapping>
23 </web-app>
```

SpringMVC示例程序-版本3

Programming Your Future

- 部署运行



SpringMVC示例程序-版本3

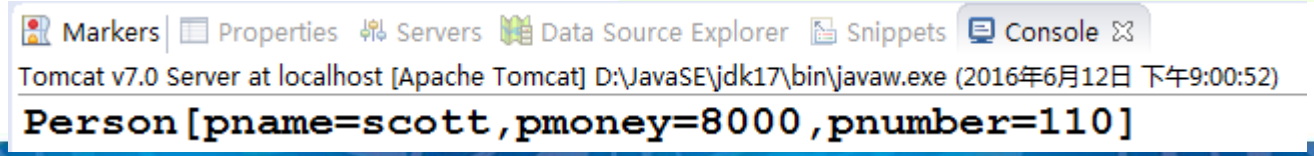
- 编写Person控制器的数据添加方法,代码如下:

```
@RequestMapping("/a1011.html")
public String add(Person p)
{
    System.out.println(p);
    return "A1011";
}
```

- 参数说明
  - person 表示数据输入页面的数据接收模型

SpringMVC示例程序-版本3

- 部署运行,请注意控制台的输出



SpringMVC示例程序-版本3

Programming Your Future

- 代码背后的秘密(1):领域模型的自动填充
  - 在上页PPT中,我们为add()方法提供了一个接收页面数据的领域模型,当页面段向该方法跳转时候,只要页面段控件名字与领域模型中某个属性名称相同,那么就会自动调用同名属性的设置子(setXXX方法),将页面数据自动填充到模型中
  - 为验证这一点我们,修改模型的设置子,代码见下页PPT

SpringMVC示例程序-版本3

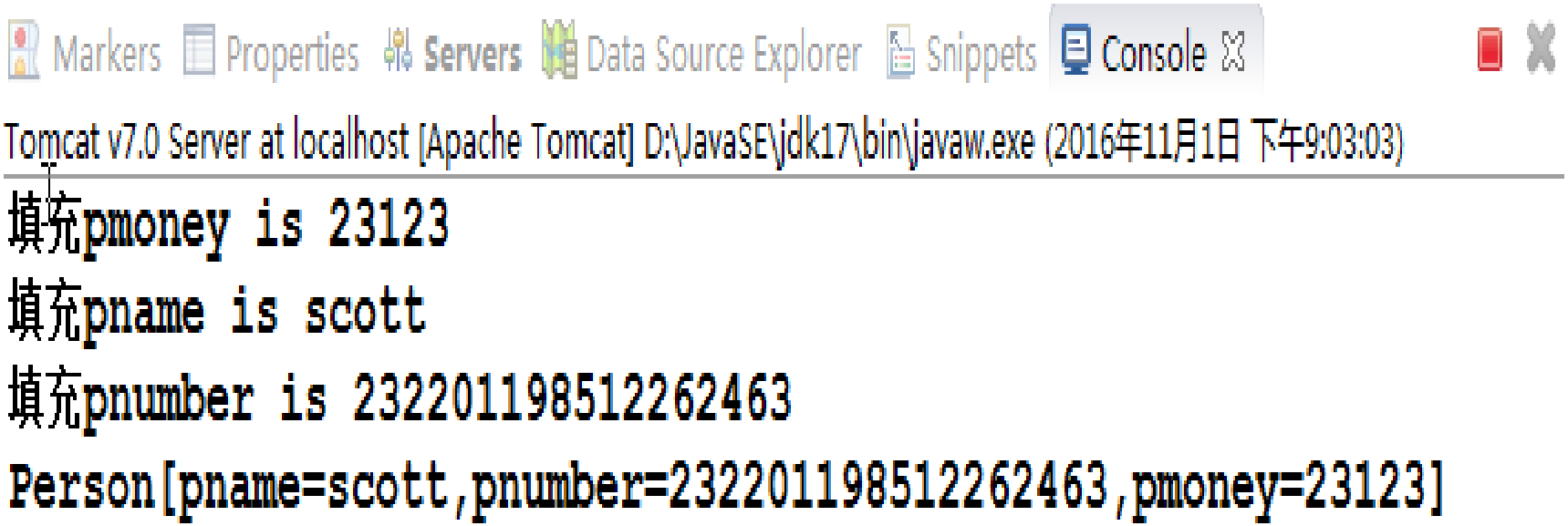
Programming Your Future

- 代码背后的秘密(1):领域模型设置子的改进

```
public void setPname(String pname)
{
     System.out.println("填充pname is "+pname);
     this.pname = pname;
}
public String getPnumber() {
     return pnumber;
}
public void setPnumber(String pnumber)
{
     System.out.println("填充pnumber is "+pnumber);
     this.pnumber = pnumber;
}
public String getPmoney() {
     return pmoney;
}
public void setPmoney(String pmoney)
{
     System.out.println("填充pmoney is "+pmoney);
     this.pmoney = pmoney;
}
```

SpringMVC示例程序-版本3

- 代码背后的秘密(1)：控制台的输出



SpringMVC示例程序-版本3

Programming Your Future

- 代码背后的秘密(2):领域模型的自动存储
  - 当方法接收到领域模型以后,会将领域模型自动存储为request对象的一个属性,
  - 属性名称为领域模型名称且第一个字母小写,其余字母不变,这样到页面段只要按照该名称就可以用EL语言进行request属性读取
  - 为验证这个问题,我们在程序中定义新的领域模型,名称为
    - PersonInfo　代码同Person
    - 并且编写方法add2(),代码见下页

SpringMVC示例程序-版本3

- 代码背后的秘密(2):新的添加方法

```
@RequestMapping("/a1012.html")
public String add2(PersonInfo pi)
{
      System.out.println(pi);
      return "A1012";
}
```

- 修改A1010.jsp的表单地址为

```
action="<%=request.getContextPath()%>/a1012.html"
```

SpringMVC示例程序-版本3

- 代码背后的秘密(2)：A1012.jsp页面代码片段

```
<tr class="title">
   <td colspan="100" >员工信息</td>
</tr>
<tr>
   <td>姓名</td>
   <td>${personInfo.pname }   </td>
</tr>
<tr>
   <td>身份证</td>
   <td> ${personInfo.pnumber }     </td>
</tr>
<tr>
   <td>月薪</td>
   <td> ${personInfo.pmoney }       </td>
</tr>
```

- 运行效果证明秘密是真实的

SpringMVC示例程序-版本3番外篇

Programming Your Future

- 对ServletAPI的访问
  - Web程序开发离不开对ServletAPI的访问,SpringMVC中提供了两种访问方式
    - A.带有侵入性的访问方式
    - B.非侵入性的访问方式

SpringMVC示例程序-版本3番外篇

- 带有侵入性的访问方式
  - 如版本1的方式,直接在控制器的方法中,引入ServletAPI做参数.这种方式的缺陷在于,会产生ServletAPI对控制器的侵入性,请看代码

```
@RequestMapping("/a1013.html")
public String add3(Person p,
                   HttpServletRequest request,
                   HttpSession session
                   )
{
    request.setAttribute("msg", "request数据保存!");
    session.setAttribute("msg", "session数据保存!");
    session.getServletContext().setAttribute("msg", "app数据保存");
    return "A1011";
}
```

SpringMVC示例程序-版本3番外篇

Programming Your Future

- 非侵入性的访问方式:
  - 侵入性会造成控制器的无法独立测试运行(这个真的很重要吗?),为此SpringMVC中提供了一个名字的Model的类,通过该类同样可以实现对ServletAPI的操作,并且没有侵入性,
  - 请看代码

```
@RequestMapping("/a1014.html")
public String add(Person p,Model model)
{
    model.addAttribute("msg", "model的属性,默认为request作用域");
    System.out.println(p);
    return "A1011";
}
```

SpringMVC示例程序-版本3番外篇

Programming Your Future

- 非侵入性写入Session:
  - 一如上页PPT所言,model属性的默认作用域是request,那么如果需要将属性存入session咋整啊?小意思,我们告诉它存入session就可以了,看代码:

```
@Controller
@SessionAttributes("msg2")
public class A1010Controller
{
    @RequestMapping("/a1014.html")
    public String add(Person p,Model model)
    {
        model.addAttribute("msg", "model的属性,默认为request作用域");
        model.addAttribute("msg2","这是存入session的数据");
        System.out.println(p);
        return "A1011";
    }
```

@RequestMapping的类级映射与子方法调用

- @RequestMapping除可以将方法映射成请求路径外,还可以对控制器类进行映射,见代码

```
14 @Controller
15 @RequestMapping("/pc2")
16 public class PersonController2
17 {
18     @RequestMapping(value="/open.html")
19     public String openWindow()
20     {
21         return "A1020";
22     }
23
24     @RequestMapping("/add.html")
25     public String addPerson(Person person,Model model)
26     {
27         System.out.println(person);
28         model.addAttribute("person", person);
29         return "A1021";
30     }
31 }
```

localhost:8080/mvc2/pc2/open.html

```
<form id="myform" action="<%=request.getContextPath()%>/pc2/add.html" method="post">
```

SpringMVC示例程序-版本3

- 404问题的解决方案:
- 这个问题在开发实物中完全可以交给web.xml来进行管理.
- 在web.xml增加如下选项配置

```
<error-page>
    <error-code>404</error-code>
    <location>/NotFound.jsp</location>
</error-page>
```

- 目标页面的上的所有图片资源需要放弃过滤:

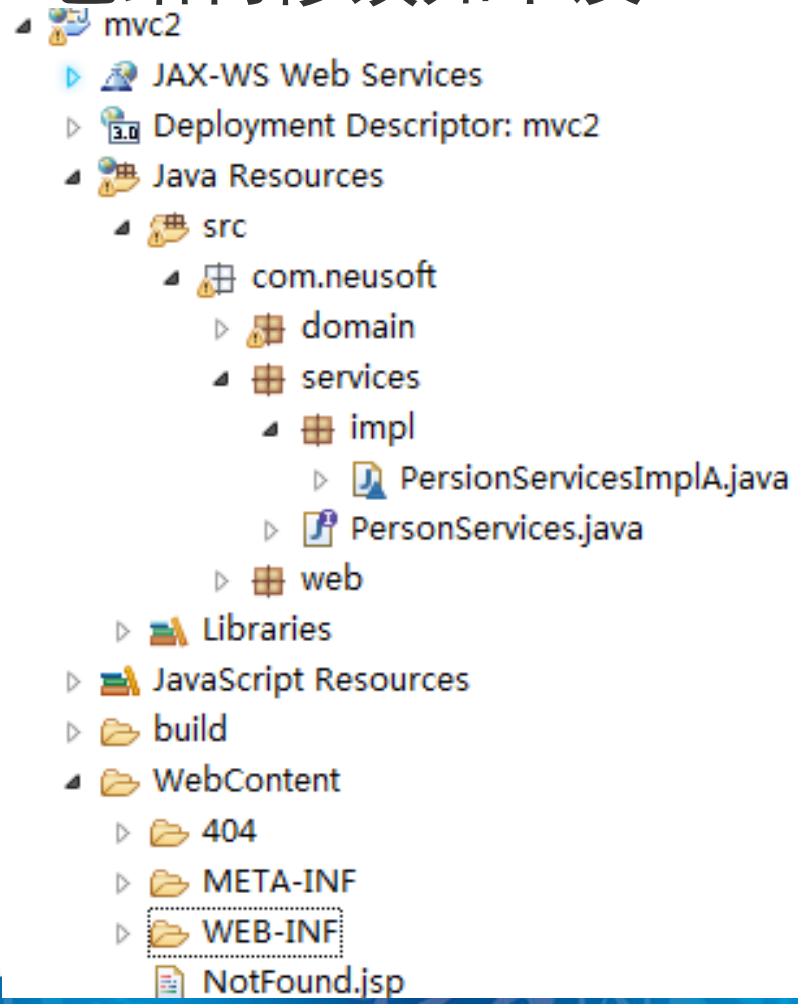
```
<mvc:resources mapping="/404/**" location="/404/" />
```

SpringMVC示例程序-版本3

- 控制器的模型载入问题
- 依据分层体系原则,控制器直接打交道的对象是Services组件.下面我们编写SpringMVC模式下的业务逻辑组件PersonServices接口及其实现类,并通过注解和xml两种方式完成以来注入

SpringMVC示例程序-版本3

- 包结构修改如下及Services层实现方案如下



SpringMVC示例程序-版本3

Programming Your Future

- PersonServices接口代码如下

```
1 package com.neusoft.services;
2
3 public interface PersonServices
4 {
5     public boolean addPerson()throws Exception;
6
7 }
8
```

SpringMVC示例程序-版本3

- 实现类PersonServicesImplA代码如下

```
1  package com.neusoft.services.impl;
2
3  import org.springframework.stereotype.Service;
4
5  import com.neusoft.services.PersonServices;
6
7  @Service   //该注解指定该类为Services组件
8  class PersionServicesImplA implements PersonServices
9  {
10     @Override
11     public boolean addPerson() throws Exception
12     {
13         System.out.println("完成数据添加");
14         return true;
15     }
16
17 }
```

SpringMVC示例程序-版本3

- 修改控制器PersonController,完成services注入

```
@Autowired  //安装类型自动装配
private PersonServices services=null;

@RequestMapping("/a1011.html")
public String addPerson(Person person,Model model)
{
        try
        {
            boolean tag=this.services.addPerson();
            System.out.println("tag is "+tag);
            System.out.println(person);
            model.addAttribute("person", person);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }

    return "A1011";
}
```

SpringMVC示例程序-版本3

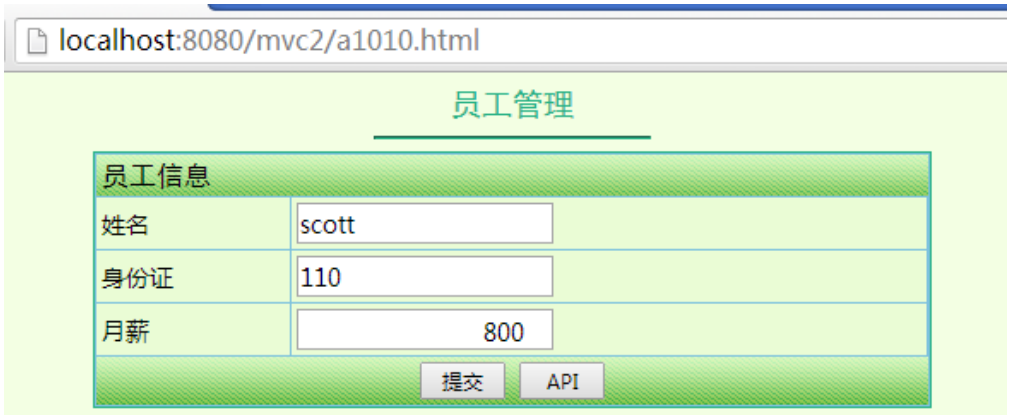
- 添加注解扫描器,扫描services包

```
<context:component-scan base-package="com.neusoft.services"/>
```

SpringMVC示例程序-版本3

Programming Your Future

- 部署运行,注意看控制台输出




```
Tomcat v7.0 Server at localhost [Apache Tomcat] D:\JavaSE\jdk17\bin\javaw.exe (2016年6月12日 下午10:03:36)
完成数据添加
tag is true
Person[pname=scott,pmoney=800,pnumber=110]
```

SpringMVC示例程序-版本3

Programming Your Future

- 自动装配引发的问题
- @Autowired
  - 自动装配
  - 默认安装类型自动进行装配,
  - 在本工程中,程序会自动搜索PersonServices的子类,自动对PersonController的属性personServices进行注入.
- 于是麻烦了.....
  - 我们无法保证一个Services接口只有一个实现类,
  - 也就是说,在一个系统中,一个Services接口完全是可能存在多个实现类的,尤其在软件架构的设计中,为了简化处理,我们一般会抽象所有Services的共同接口,从这个角度说,所有Services的实现类,实际上是共用一个接口的.

SpringMVC示例程序-版本3

Programming Your Future

- 自动装配引发的问题(续)
- 在本工程中,如果PersonServices存在不止一个的实现类,安装自动装配原则,到底将那个实现类注入到控制器PersonController的personServices属性呢?
- **实践是检验真理的唯一标准,走两步,试试**

SpringMVC示例程序-版本3

Programming Your Future

- 自动装配引发的问题(续)
- 在本工程中,如果PersonServices存在不止一个的实现类,安装自动装配原则,到底将那个实现类注入到控制器PersonController的personServices属性呢?
- **实践是检验真理的唯一标准,走两步,试试**

SpringMVC示例程序-版本3

- 自动装配引发的问题(再续)
- 为PersonServices编写新的实现类,PersonServicesImplB,代码如下:

```
1  package com.neusoft.services.impl;
2
3  import org.springframework.stereotype.Service;
4
5  import com.neusoft.services.PersonServices;
6
7  @Service
8  class PersonServicesImplB implements PersonServices
9  {
10     @Override
11     public boolean addPerson() throws Exception
12     {
13         System.out.println("add Person impl B");
14         return true;
15     }
16
17 }
```

SpringMVC示例程序-版本3

Programming Your Future

- 自动装配引发的问题(继续)
- 启动程序:报错!!!!

- 错误原因:
  - 准备了一桌子酒宴,来了两桌子客人,这酒怎么喝?Spring也蒙圈了!

SpringMVC示例程序-版本3

Programming Your Future

- 自动装配的安全方案
  - 为每个Services指定名称
  - 在Controller中注入Services实现类时候,增加按名称注入选项
  - 代码见后继三页ppt

SpringMVC示例程序-版本3

Programming Your Future

- 自动装配的安全方案
  - PersonServicesImplA代码修改如下:

```
1  package com.neusoft.services.impl;
2
3  import org.springframework.stereotype.Service;
4
5  import com.neusoft.services.PersonServices;
6
7  //指定services实现类名称,相当于xml文件中Bean的ID
8  @Service("personServicesA")
9  class PersionServicesImplA implements PersonServices
10 {
11     @Override
12     public boolean addPerson() throws Exception
13     {
14         System.out.println("完成数据添加");
15         return true;
16     }
17
18 }
```

SpringMVC示例程序-版本3

Programming Your Future

- 自动装配的安全方案
  - PersonServicesImplB代码修改如下:

```
1  package com.neusoft.services.impl;
2
3  import org.springframework.stereotype.Service;
4
5  import com.neusoft.services.PersonServices;
6
7  @Service("personServicesB")
8  class PersonServicesImplB implements PersonServices
9  {
10     @Override
11     public boolean addPerson() throws Exception
12     {
13         System.out.println("add Person impl B");
14         return true;
15     }
16
17 }
```

SpringMVC示例程序-版本3

Programming Your Future

- 自动装配的安全方案
  - 控制器注入方式上增加名称选项:

```
@Autowired  //安装类型自动装配
@Qualifier("personServicesB")  //按照ID注入
private PersonServices services=null;

@RequestMapping("/a1011.html")
public String addPerson(Person person,Model model)
{
        try
        {
            boolean tag=this.services.addPerson();
            System.out.println("tag is "+tag);
            System.out.println(person);
            model.addAttribute("person", person);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }

    return "A1011";
}
```

SpringMVC示例程序-版本3

- 自动装配的安全方案
  - 部署运行,单击页面提交按钮,注意控制台输出:

```
Tomcat v7.0 Server at localhost [Apache Tomcat] D:\JavaSE\jdk17\bin\javaw.exe (2016年6月12日 下午10:32:59)
add Person impl B
tag is true
Person[pname=scott,pmoney=800,pnumber=110]
```

SpringMVC示例程序-版本3

Programming Your Future

- 自动装配的安全方案
  - 在软件开发实务中,一般而言,对于Services组件,DAO组件我们建议还是使用传统的XML配置方式
  - 这相当于是说,在工程中,为业务层以下的代码,提供一份代码及调用关系清单,项目维护会更加方便
  - 而对于SpringMVC的控制器,一般采用基于注解的方式配置.这更加方便.
  - **物尽其力而不达终极,是为知其雄而守其雌**
  - **任何技术都不是尽善尽美的,要学会变通,程序才会圆融**

SpringMVC示例程序-版本3

Programming Your Future

- 混合兵团的实现方案
- 很简单
  - 对于Services和DAO,采用XML文件配置
  - 在工程Spring配置文件中,导入Services和DAO配置文件
  - 控制器层在做注解注入时候,指定注入对象的名称

- OK,就到这里吧.

- 哦对了,DAO的注解需要告诉你们
- @Repository