SpringMVC

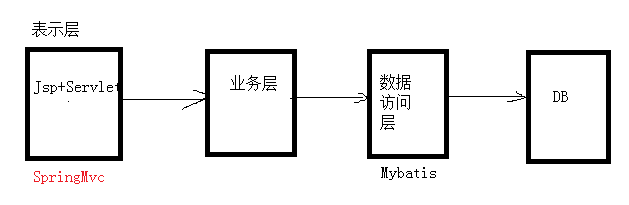
课程目标

1. 理解mvc设计模式
2. (重点)介绍Springmvc和Springmvc工作原理
3. 使用Idea创建Maven web工程并添加Tomcat插件运行工程
4. (重点)搭建Springmvc的开发环境
5. 使用Springmvc实现登入

>>理解控制器的创建

>>理解ModelAndView对象

回顾:三层架构



1. MVC设计模式(思想)
2. 理解MVC的术语

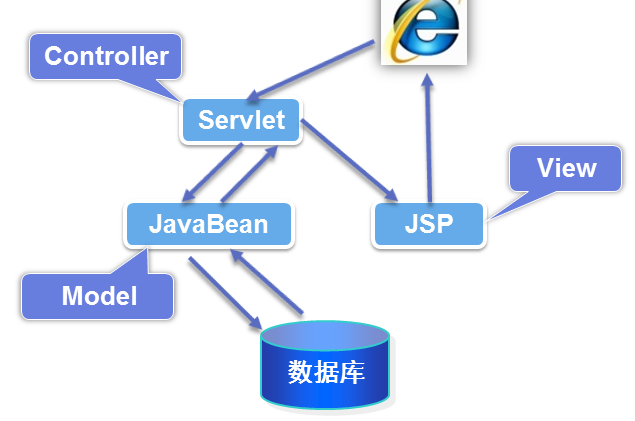
M(Model):模型层 对应组件:JavaBean

V(VIEW):视图层 对应组件:HTML或者 Jsp

C(Controller):控制器 对应的组件:Servlet

1. Jsp+servlet+Javabean(Model 2模式)开发模式与Mvc的关系

Jsp+servlet+javabean是实现mvc思想



1. mvc的优点和**缺点**:

**MVC 优点**

* 1. **多视图共享一个模型，大大提高代码的可重用性**
  2. **MVC三个模块相互独立，松耦合架构**
  3. **控制器提高了应用程序的灵活性和可配置性**
  4. **有利于软件工程化管理**

**完美的系统架构 = 松耦合+高重用性+高扩展性**

**MVC 缺点**

* + **原理复杂**
  + **增加了系统结构和实现的复杂性**
  + **视图对模型数据的低效率访问**

1. springmvc框架
2. 介绍springmvc框架

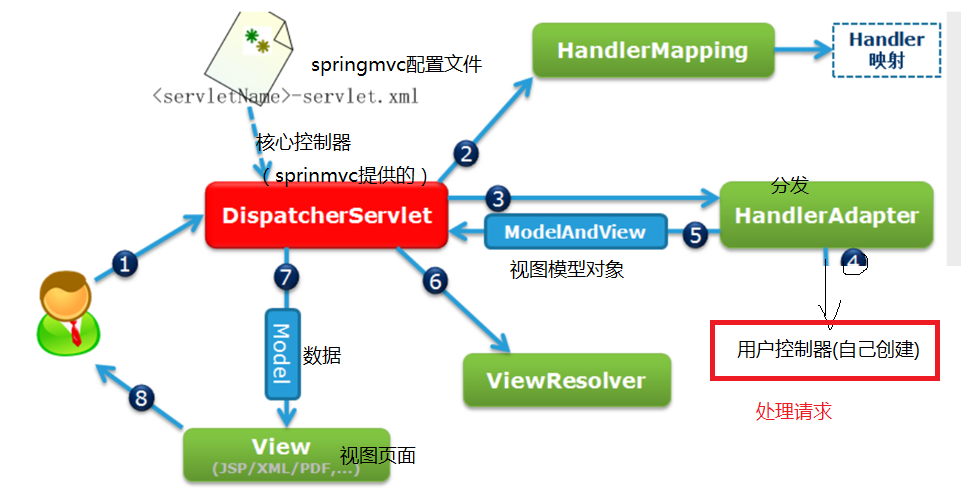
Springmvc框架是spring框架的子集，实现MVC思想的产物.

注意:常用mvc框架:Springmvc、struts2

1. (重点)Springmcv的工作原理

用户—发请求🡪核心控制器(DispatcherServlet)—分发请求🡪用户控制器--🡪调用模型层

视图进行呈现<--ModelAndView(视图模型对象)--返回-----------



1. (重点)使用springmvc框架（版本4.3.3）的步骤：
2. 导入springmvc相关的jar包(添加依赖)

|  |
| --- |
| *<!-- 初始化框架的版本号 -->* <**properties**>  <**spring.version**>4.3.3.RELEASE</**spring.version**> </**properties**>  <**dependencies**>  *<!--spring支持的相关依赖和springmvc依赖(spring-web、spring-webmvc)-->* <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-web</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>   <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-webmvc</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>   <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-core</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>   <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-oxm</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>   <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-tx</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>   <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-jdbc</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>   <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-aop</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>   <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-context-support</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>   <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-test</**artifactId**>  <**version**>${spring.version}</**version**>  </**dependency**>  *<!-- 引入Spring 依赖 end -->* |

2． 添加springmvc的配置文件(springmvc.xml)：存于resources目录

复制现有的springmvc配置文件

|  |
| --- |
| *<?***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"  xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="  http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-4.3.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context-4.3.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd  "**>  *<!--配置用户控制器的-->* </**beans**> |

1. 修改web.xml文件:配置核心控制器

|  |
| --- |
| *<!--核心控制器  1.拦截所有的用户请求，并根据请求的名称分发到用户控制  2.读取sprignmvc配置文件 -->* <**servlet**>  <**servlet-name**>springmvc</**servlet-name**> <**servlet-class**>org.springframework.web.servlet.DispatcherServlet</**servlet-class**>  <**init-param**>  <**param-name**>contextConfigLocation</**param-name**>  <**param-value**>classpath:springmvc.xml</**param-value**>  </**init-param**>  <**load-on-startup**>1</**load-on-startup**> </**servlet**> <**servlet-mapping**>  <**servlet-name**>springmvc</**servlet-name**>  *<!--拦截的请求-->* <**url-pattern**>/</**url-pattern**> </**servlet-mapping**> |

注意 ：

问题一:如果web.xml文件存在以下错误:没有servlet的环境



解决:在pom文件中添加以下依赖

|  |
| --- |
| *<!--添加servlet的依赖-->* <**dependency**>  <**groupId**>javax.servlet</**groupId**>  <**artifactId**>javax.servlet-api</**artifactId**>  <**version**>3.0.1</**version**>  <**scope**>provided</**scope**> </**dependency**> |

1. 创建springmvc控制器的步骤

注意:控制器类以后都统一放到:xxx.xxx.controller包中.

1. 新建一个类，实现org.springframework.web.servlet.mvc.Controller接口，并重写handleRequest方法

|  |
| --- |
| *//创建控件器类* **public class** LoginController **implements** Controller {  *//处理请求的方法* @Override  **public** ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse response) **throws** Exception {  处理请求的代码  System.***out***.println(**"我是控制器,可以处理请求啦"**);  **return null**;  } } |

1. 在springmvc配置文件中，定义控制器

|  |
| --- |
| *<!--配置用户控制器的--> <!--定义控制器 配置控制器的请求路径 <bean name="请求路径" class="控制器类限定名称"></bean> -->* <**bean name="/login" class="com.kgc.controller.LoginController"**></**bean**> |

1. 发请求访问控制器
2. (综合练习)实现登入的业务:

用户是admin 密码:123

1. 制作表单

|  |
| --- |
| <%@ **page language**="**java**" **pageEncoding**="**utf-8**" **contentType**="**text/html; utf-8**" %> <**html**> <**body**> <**form name="form" method="post" action="login"**>  用户名:<**input type="text" name="uname"**><**br**/>  密码:<**input type="text" name="password"**><**br**/>  <**input type="submit" value="登入" name="sub"**><**br**/> </**form**> </**body**> </**html**> |

1. 编写控制器实现登入的验证

|  |
| --- |
| **public class** LoginController **implements** Controller {  @Override *//处理请求的方法* **public** ModelAndView handleRequest(HttpServletRequest request,  HttpServletResponse response) **throws** Exception {  *//System.out.println("可以处理请求啦....");  //1.取值* String uname=request.getParameter(**"uname"**);  String password=request.getParameter(**"password"**);  *//2.比较验证* ModelAndView mav=**null**;  **if**(uname.equals(**"admin"**)&&password.equals(**"123"**)){  *//返回成功页  //ModelAndView mav=new ModelAndView("网页路径");* mav=**new** ModelAndView(**"success.jsp"**);  }**else**{  *//返回登入页* mav=**new** ModelAndView(**"index.jsp"**);  }  **return** mav;  } }  注意:控制器类需要在springmvc配置文件中进行配置  <**bean name="/login" class="com.kgc.controller.LoginController"**></**bean**> |

1. 视图模型对象(ModelAndView)
2. 创建模型视图对象

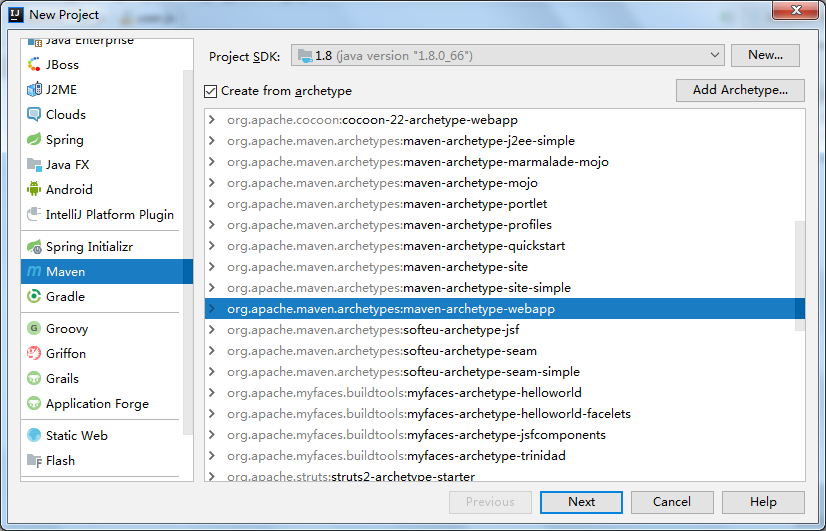
ModelAndView mav=**new** ModelAndView(**"网页路径"**);

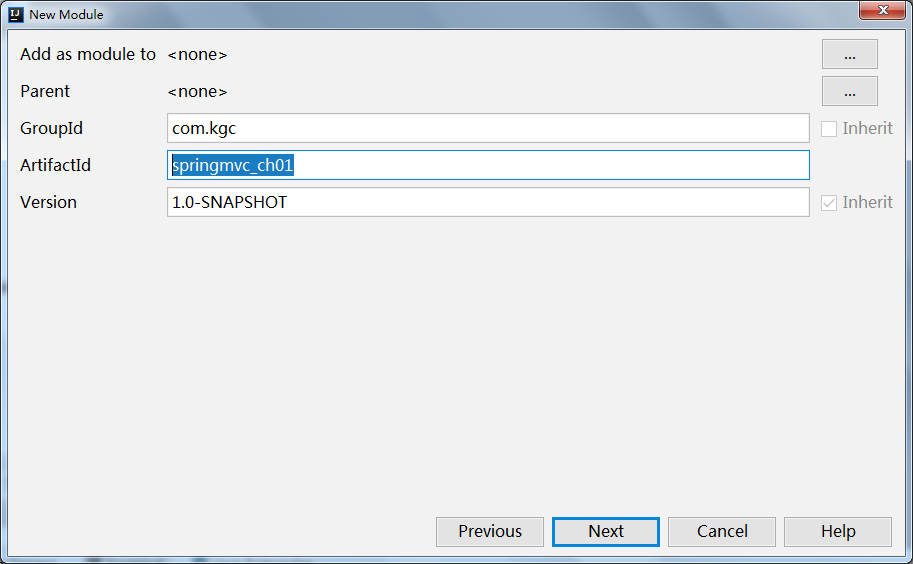
2. 给模型视图对象添加数据 (底层还是将数据放到作用域)  
mav.getModel().put("键",值);  
 mav.getModel().put(**"name"**,**"我会使用模型对象"**);

注意:视图页面中还是通过el表过式获取数据

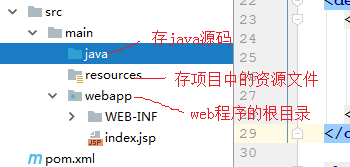
七、其它

1. 使用idea创建maven的web工程





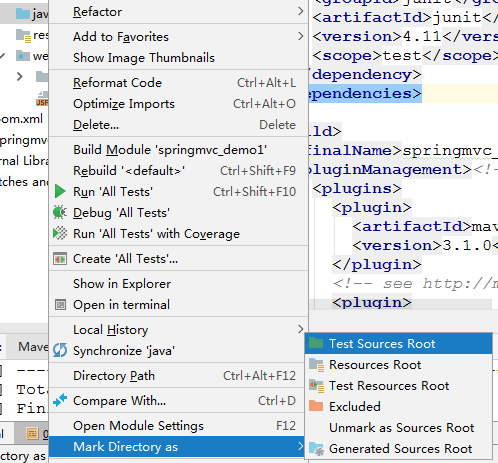
1. 在工程中创建java目录、resources目录



1. 修改java目录和resouces目录的作用

在java目录上右键🡪Mark Directory AS🡪Source Root

resouces上右键🡪Mark Directory AS🡪reSource Root



4．在maven的web工程中添加tomcat插件运行项目

4.1 修改pom文件，添加Tomcat插件

|  |
| --- |
| <bulid>  *<!--添加tomcat插件-->* <**plugins**>  <**plugin**>  <**groupId**>org.apache.tomcat.maven</**groupId**>  <**artifactId**>tomcat7-maven-plugin</**artifactId**>  <**version**>2.2</**version**>  *<!--如果configuration不加，默认端口为8080  path 项目名称  port 表示端口  servlet 表示服务器名称tomcat7  -->* <**configuration**>  <**path**>/</**path**>  <**port**>8080</**port**>  <**server**>tomcat7</**server**>  </**configuration**>  </**plugin**>  </**plugins**>  <bulid> |

2.使用内嵌的tomcat运行工程

