Spring MVC 开发技术

邹国兵，博士

副教授、博导/硕导

服务计算与数据挖掘实验室

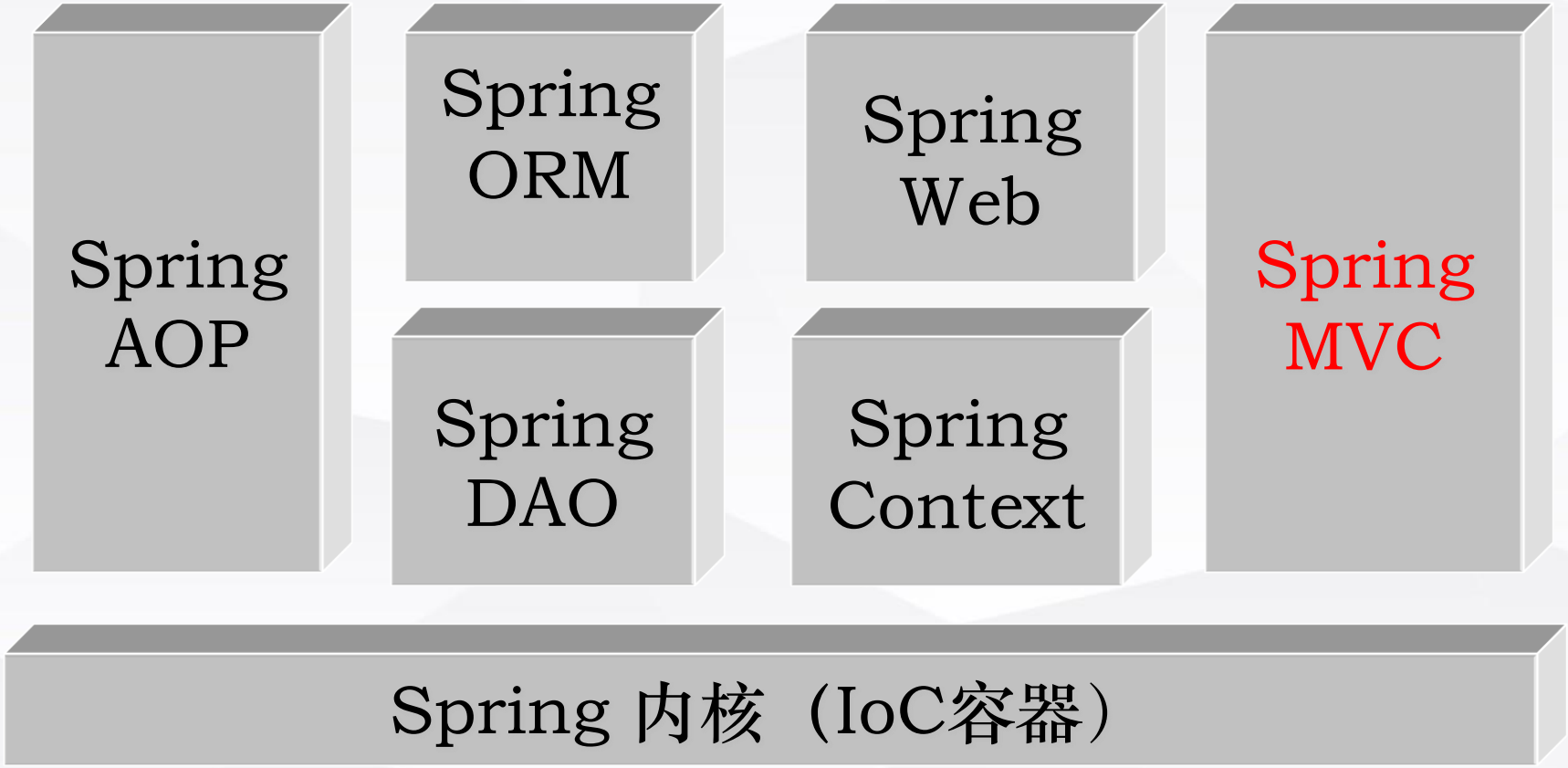
http://scdm-shu.github.io

课程内容安排

- Spring MVC简介
- Spring MVC体系架构
- Spring MVC工作原理
- 基于Controller接口的Spring MVC开发
- 基于注释的Spring MVC开发
- Spring MVC与Struts框架比较

Spring MVC 简介

Spring MVC属于Spring整个框架中的Web部分。



Spring框架中的7个主要模块

Spring MVC简介

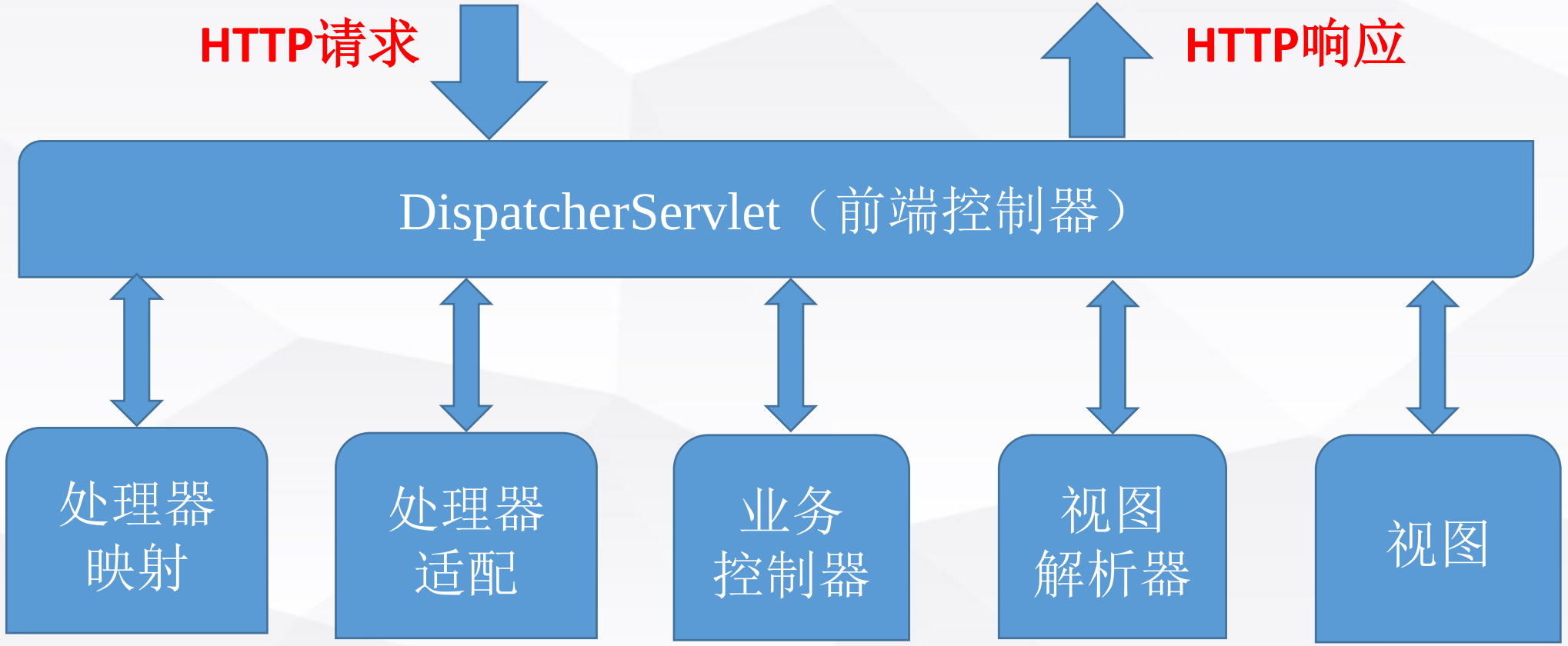
- Spring 框架提供了构建Web应用程序的全功能MVC模块，即Spring MVC。
- Spring MVC是Spring整个框架中实现的一个轻量级MVC框架。
- Spring MVC属于Spring Framework的衍生产品。
- Spring MVC与Strut 2的功能相同，但其使用方便性和性能优于Struts 2。
- Strut 2基于配置文件进行控制，配置Action进行拦截控制；Spring MVC基于注释进行控制，通过注释@Controller可将一个类说明为业务控制器。

课程内容安排

- Spring MVC简介
- Spring MVC体系架构
- Spring MVC工作原理
- 基于Controller接口的Spring MVC开发
- 基于注释的Spring MVC开发
- Spring MVC与Struts框架比较

Spring MVC框架的体系架构

Spring MVC 体系框架



Spring MVC 的体系框架

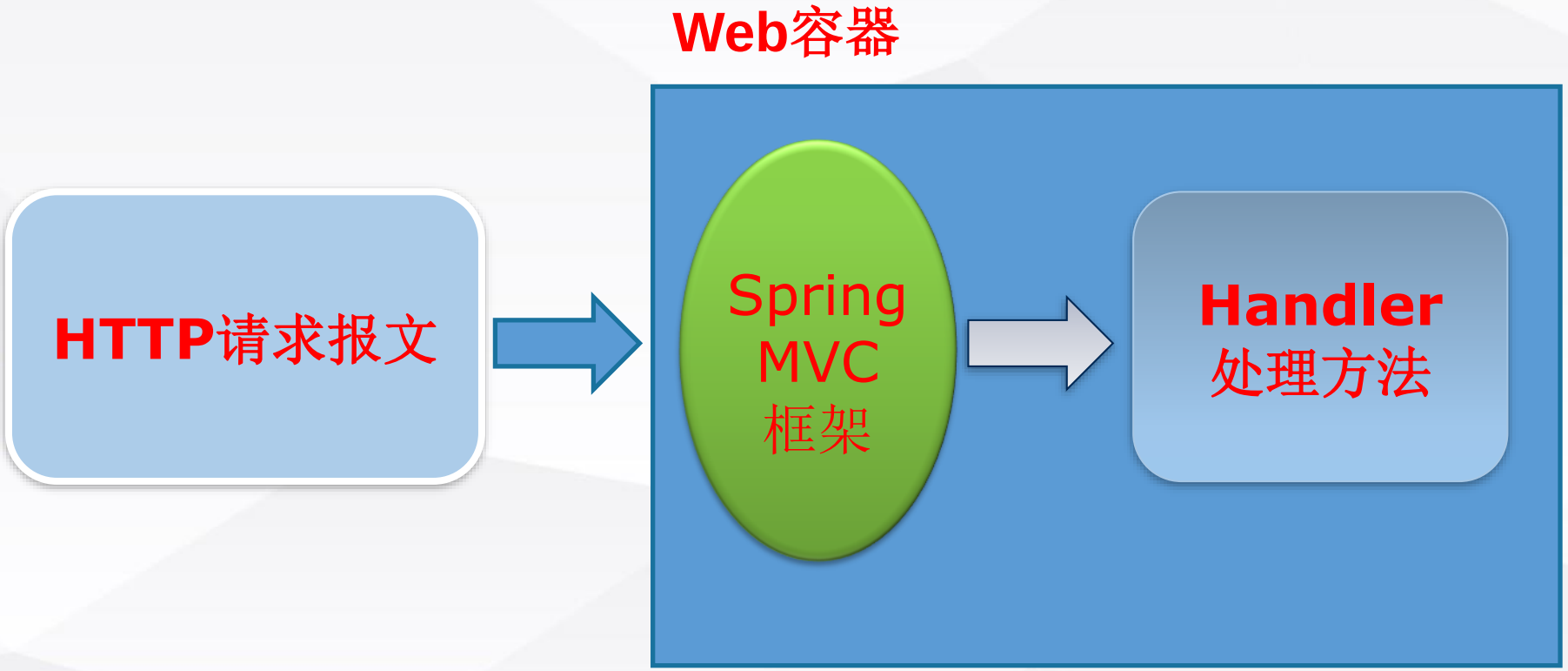
Spring MVC框架的M、C和V的具体实现：

- ✓ 在Spring MVC框架中，模型是Model或ModelAndView封装的Map。模型中的数据可来自于程序、数据库、文件、外部服务等。
- ✓ Spring MVC的控制器有两种实现方式：
  1）实现接口Controller；
  2）通过@Controller注释，带有@Controller注释的类可以被Spring检索为控制器。
- ✓ 视图有多种，如JSP、JSTL、FreeMarker、PDF等。

课程内容安排

- Spring MVC简介
- Spring MVC体系架构
- Spring MVC工作原理
- 基于Controller接口的Spring MVC开发
- 基于注释的Spring MVC开发
- Spring MVC与Struts框架比较

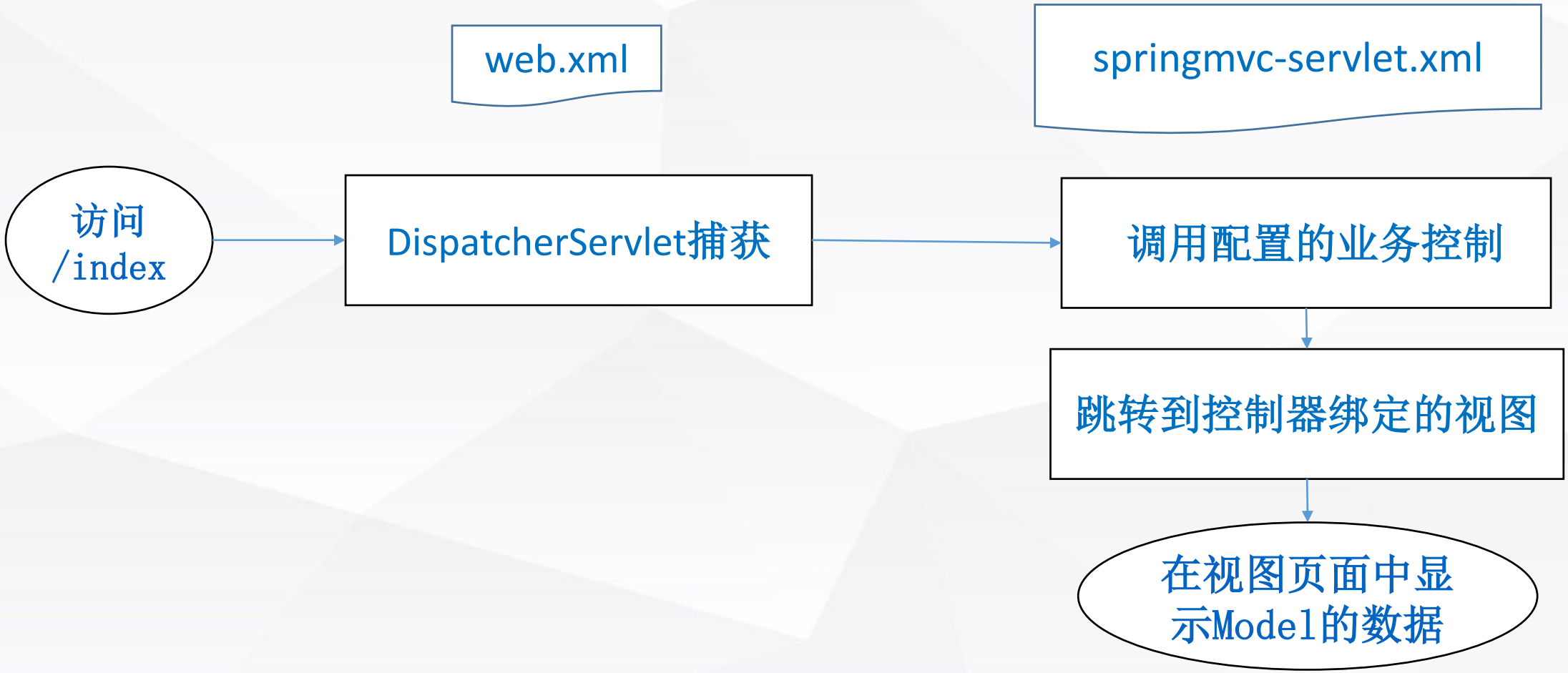
Spring MVC 框架工作原理



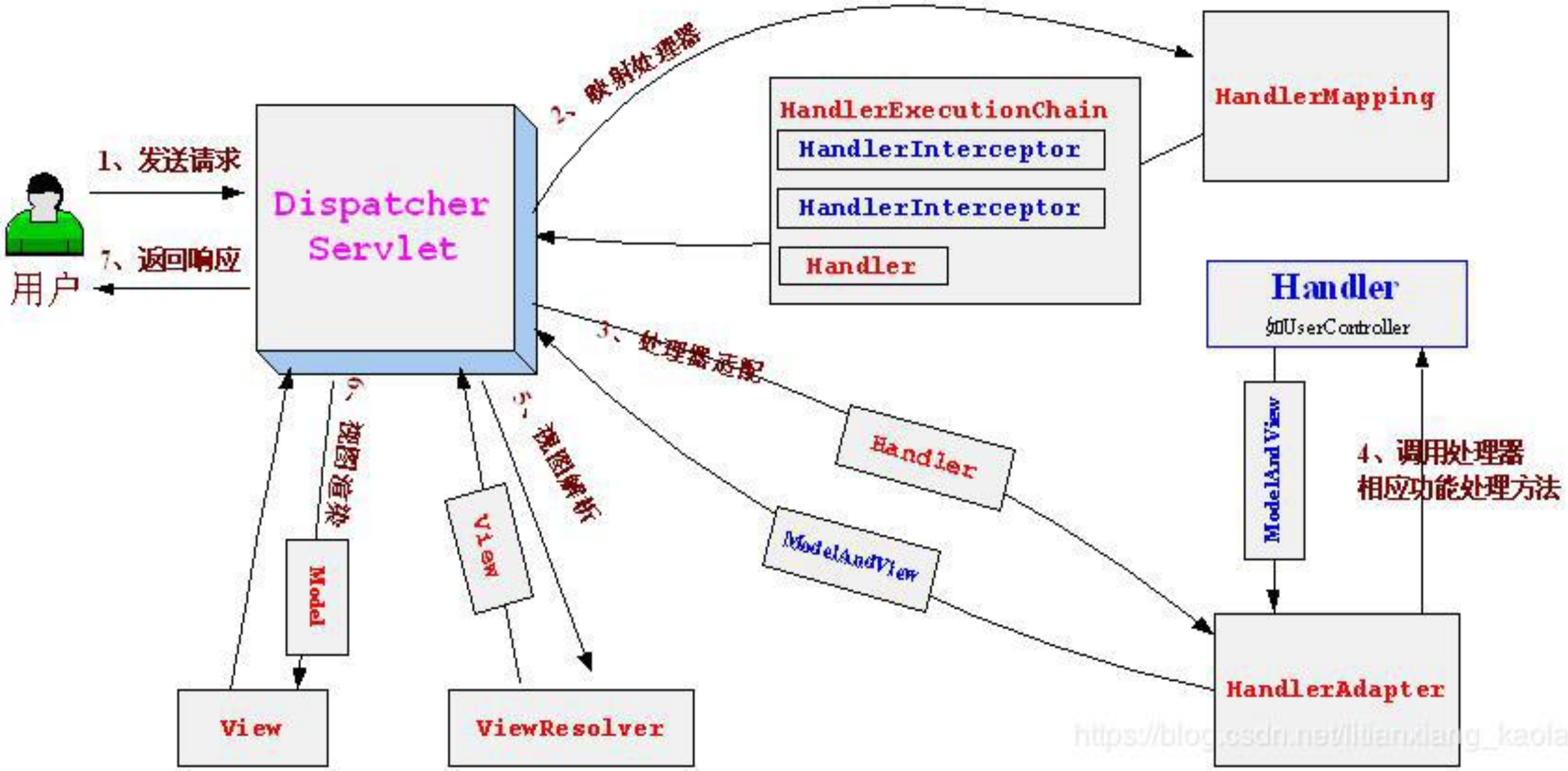
DispatcherServlet的定义方式与Struts中定义核心控制器的方式相同。

Spring MVC 框架工作原理

基于Controller控制器的执行流程：



Spring MVC 框架工作原理



Spring MVC 框架工作原理

具体流程：

（1）用户发送请求—>DispatcherServlet，前端控制器作为统一访问点，进行全局的流程控制，所以它收到请求后自己不进行处理，而是委托给其它的解析器进行处理。

（2）DispatcherServlet—>HandlerMapping，处理器映射根据请求URL得到URI，再找到具体的处理器，这样把请求映射为HandlerExecutionChain对象（包含一个Handler处理器对象、多个HandlerInterceptor拦截器对象），并返回给DispatcherServlet。

（3）DispatcherServlet根据获得的Handler，选择合适的HandlerAdapter。处理器适配器把处理器(Handler)包装为适配器，从而支持处理多种Handler，调用Handler实际处理数据的方法。

Spring MVC 框架工作原理

（4）执行处理器(Handler，也叫后端控制器)，具体处理方法：

在填充Handler方法的入参过程中，根据配置，Spring将帮做一些额外的工作：

1）**消息转换**。将请求消息数据（如JSON、XML等）转换成对象，或将对象转换成指定的响应消息。

2）**数据类型转换**。对请求消息进行数据类型转换，如String转换成Integer等。

3）**数据格式化**。对请求消息进行数据格式化，如将字符串转化成格式化数字或格式化日期。

4）**数据验证**。验证数据的有效性（如长度、格式等。

处理器（Handler）执行完成后，返回ModelAndView（包含业务对象返回的模型数据、逻辑视图名）给DispatcherServlet。

Spring MVC 框架工作原理

（5）DispatcherServlet将ModelAndView传给ViewReslover（以应对多种视图）。

（6）视图解析器ViewReslover把逻辑视图名（根据视图解析器的配置文件，加上前缀和后缀）解析为具体的View，返回给DispatcherServlet。

（7）DispatcherServlet对物理View根据传进来的Model模型数据进行渲染（即将模型数据填充至视图中，此处的Model实际是一个Map数据结构），最后将渲染后的视图返回响应给用户。

课程内容安排

- Spring MVC简介
- Spring MVC体系架构
- Spring MVC工作原理
- 基于Controller接口的Spring MVC开发
- 基于注释的Spring MVC开发
- Spring MVC与Struts框架比较

基于Controller接口的Spring MVC开发

基于Controller接口的开发步骤

第一步：创建Web项目并导入相应的jar包

第二步：配置前端控制器

第三步：创建控制器组件

第四步：创建index.jsp

基于Controller接口的Spring MVC开发

1.支持Spring MVC开发的jar包体现在以下依赖坐标

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
</dependency>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
</dependency>
```

注意：使用注解方式时，项目需要依赖Spring的AOP包，因此要能保证在lib目录中添加spring-aop的包，否则程序运行时会报错！

基于Controller接口的Spring MVC开发

2. 前端控制器DispatcherServlet

- 在多数MVC架构中，都有一个用于调度控制的Servlet。
- Spring MVC架构中的调度控制器为DispatcherServlet，它充当前端控制器。
- DispatcherServlet继承自HttpServlet,其核心功能是捕获用户的请求，然后分发给相应的业务控制器处理(实际上是先分发给HandlerMapping，它将请求的资源（URL）与业务控制器关联）。

基于Controller接口的Spring MVC开发

```
//web.xml
<servlet>
  <servlet-name>appServlet</servlet-name>
  <servlet-class>
      org.springframework.web.servlet.DispatcherServlet
  </servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/springmvc-config.xml</param-value>
  </init-param>
</servlet>
<servlet-mapping>
   <servlet-name>appServlet</servlet-name>
   <url-pattern>/</url-pattern>
</servlet-mapping>
```

基于Controller接口的Spring MVC开发

- 加载前端控制器时，需要Spring MVC 配置文件。
- 默认在应用文件夹的WEB-INF下找对应的[servlet-name]-config.xml，也可以放到servlet元素的子元素init-param指定的地方。

基于Controller接口的Spring MVC开发

springmvc-servlet.xml

```
<beans>
<bean id="simpleUrlHandlerMapping"  //1.定义一个处理器映射器
   class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
     <property name="mappings">  //配置映射
           <props>
              <prop key="/index">MyController</prop>//根据URL确定具体的Handler
           </props>
      </property>
   </bean>
   <bean id="MyController" class="com.shu.ces.controller.MyController">//2.定义控制器
   </bean>
</beans>
```

该配置文件定义了一个业务控制器MyController，并映射到与/index。

• 基于Controller接口的Spring MVC开发

3.处理器映射器HandlerMapping

•在上述Spring MVC的配置文件中，配置了一个处理器映射器**simpleUrlHandlerMapping**，告诉以简单的**URL**（其它可为**bean**）作为**URL**进行查找

•处理器映射器对请求的URL进行解析，得到对应的URI。

• 然后根据URI，调用HandlerMapping获得一个Handle配置的所有相关对象，包括Handle对象以及Handle对象对应的拦截器，这些对象被封装到一个HandlerExecutionChain对象中返回。

• Spring MVC提供了不同的映射器实现，支持不同的映射方式，如实现接口方式、注解方式等。

• 基于Controller接口的Spring MVC开发

3.处理器映射器HandlerMapping （续）

•处理器映射器将会把请求映射为 HandlerExecutionChain 对象（包含一个 Handler 处理器对象、多个 HandlerInterceptor 拦截器对象），通过这种策略模式，很容易添加新的映射策略。

• 处理器映射器有三个，且可共存，相互不影响，分别是 SimpleUrlHandlerMapping、BeanNameUrlHandlerMapping和 ControllerClassNameHandlerMapping；

• 基于Controller接口的Spring MVC开发

3.处理器映射器HandlerMapping （续）

• 三个处理器映射器中BeanNameUrlHandlerMapping默认映射器，即使不配置，默认就使用该处理器映射器进行映射请求。

```
<bean
class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping">
</bean>
//映射器把hello请求映射到该处理器
<bean id="testController" name="/hello"
class="com.shu.ces.controller.TestController" >
</bean>
```

• 基于Controller接口的Spring MVC开发

3.处理器映射器HandlerMapping （续）

SimpleUrlHandlerMapping处理器映射器可以配置多个映射对应一个处理器。

```
<bean
class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
 <property name="mappings">
        <props>
        <prop key="/ss">testController</prop>
        <prop key="/abc">testController</prop>
        </props>
  </property>
</bean>//上面的这个映射配置表示多个应用请求访问同一个Controller。
<bean id="testController" class="org.shu.ces.controller.TestController">
</bean>
```

• 基于Controller接口的Spring MVC开发

3.处理器映射器HandlerMapping （续）

ControllerClassNameHandlerMapping处理器映射器可以不用手动配置映射，通过[类名]来访问对应的处理器。

//通过这个Mapping配置，就可以使用Controller的 [类名]来访问这个Controller。

```
<bean
class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping">
</bean>
```

• 基于Controller接口的Spring MVC开发

4.业务控制器-Handler

- 相当于Struts中的Action。
- Spring2.5以前版本，只能通过实现Controller接口来开发Handler，必须实现的方法是:

```
ModelAndView handleRequest(HttpServletRequest req,
                           HttpServletResponse res)throws Exception
```

其中ModelAndView是包含视图名或视图名和模型的对象。

- Controller接口的实现类只能处理单一请求动作，而Spring2.5之后新增的基于注释的控制器可以支持同时处理多个请求动作，并且无需实现任何接口。

基于Controller接口的Spring MVC开发

4.业务控制器-Handler的实现

4.1 通过实现接口Controller来实现业务控制器。

Controller接口提供方法handleRequest()来处理请求，Spring MVC通过ModelAndView对象把模型和视图结合在一起。

基于Controller接口的Spring MVC开发

```
//MyController.java
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.Controller;
public class MyController implements Controller {
    public ModelAndView handleRequest(HttpServletRequest request,
HttpServletResponse response) throws Exception {
        ModelAndView mav = new ModelAndView("index.jsp");  //使之包含视图名
        mav.addObject("message", "Hello Spring MVC");  //使之包含模型
        return mav;
         }
}
```

基于Controller接口的Spring MVC开发

5. HandlerAdapter：处理器适配器

- DispatcherServlet根据获得的具体Handler，选择一个合适的HandlerAdapter来处理它（HandlerAdapter用于处理多种Handler，这是适配器模式的应用）；
- HandlerAdapter调用Handler实际处理请求的方法；
- 控制器在这个方法中会设置模型数据并将视图名称返回给前端控制器DispatcherServlet，DispatcherServlet将会查询ViewResolver接口（视图解析器），得到视图的实现。

基于Controller接口的Spring MVC开发

5. HandlerAdapter：处理器适配器

- 处理器适配器有两种，可以共存，分别是SimpleControllerHandlerAdapter和HttpRequestHandlerAdapter。

基于Controller接口的Spring MVC开发

5. HandlerAdapter：处理器适配器

- SimpleControllerHandlerAdapter是默认的适配器，所有实现了org.springframework.web.servlet.mvc.Controller 接口的处理器都是通过此适配器适配执行。

基于Controller接口的Spring MVC开发

5. HandlerAdapter：处理器适配器

- HttpRequestHandlerAdapter适配器将http请求封装成HttpServletResquest和HttpServletResponse对象。所有实现了org.springframework.web.HttpRequestHandler接口的处理器都是通过此适配器适配执行的。

实例：

（1）配置HttpRequestHandlerAdapter适配器

```
<!-- 配置HttpRequestHandlerAdapter适配器 -->
<bean
  class="org.springframework.web.servlet.mvc.HttpRequestHandlerAdapter">
</bean>
```

基于Controller接口的Spring MVC开发

5. HandleAdapter：处理器适配器

（2）编写处理器

```
public class HttpController implements HttpRequestHandler {

public void handleRequest(HttpServletRequest request, HttpServletResponse
   response) throws ServletException, IOException {

//给Request设置值，在页面进行回显

 request.setAttribute(“hello”, “这是HttpRequestHandler！”);          //跳转页面

 request.getRequestDispatcher("/WEB-INF/jsps/index.jsp").forward(request,
   response);}}
```

基于Controller接口的Spring MVC开发

5. HandleAdapter：处理器适配器

（3）index页面

```
<html>
  <body>
    <h1>${hello}
    </h1>
  </body>
</html>
```

基于Controller接口的Spring MVC开发

6. ViewResolver：视图解析器

- ViewResolver负责将处理结果ModelAndView生成View视图。
- ViewResolver首先根据逻辑视图名解析成物理视图名即具体的页面地址，再生成View视图对象，最后对View进行渲染（将model中的数据写入视图）将处理结果通过页面展示给用户。

• 基于Controller接口的Spring MVC开发

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

上述代码定义了一个id为viewResolver的视图解析器，并设置了视图的前缀和后缀。这样设置后，在方法中定义的视图即可简化。例如，return “success”，而不必为return “/WEB-INF/jsp/success.jsp”，在访问时，视图解析器会自动地添加前缀和后缀。

• 基于Controller接口的Spring MVC开发

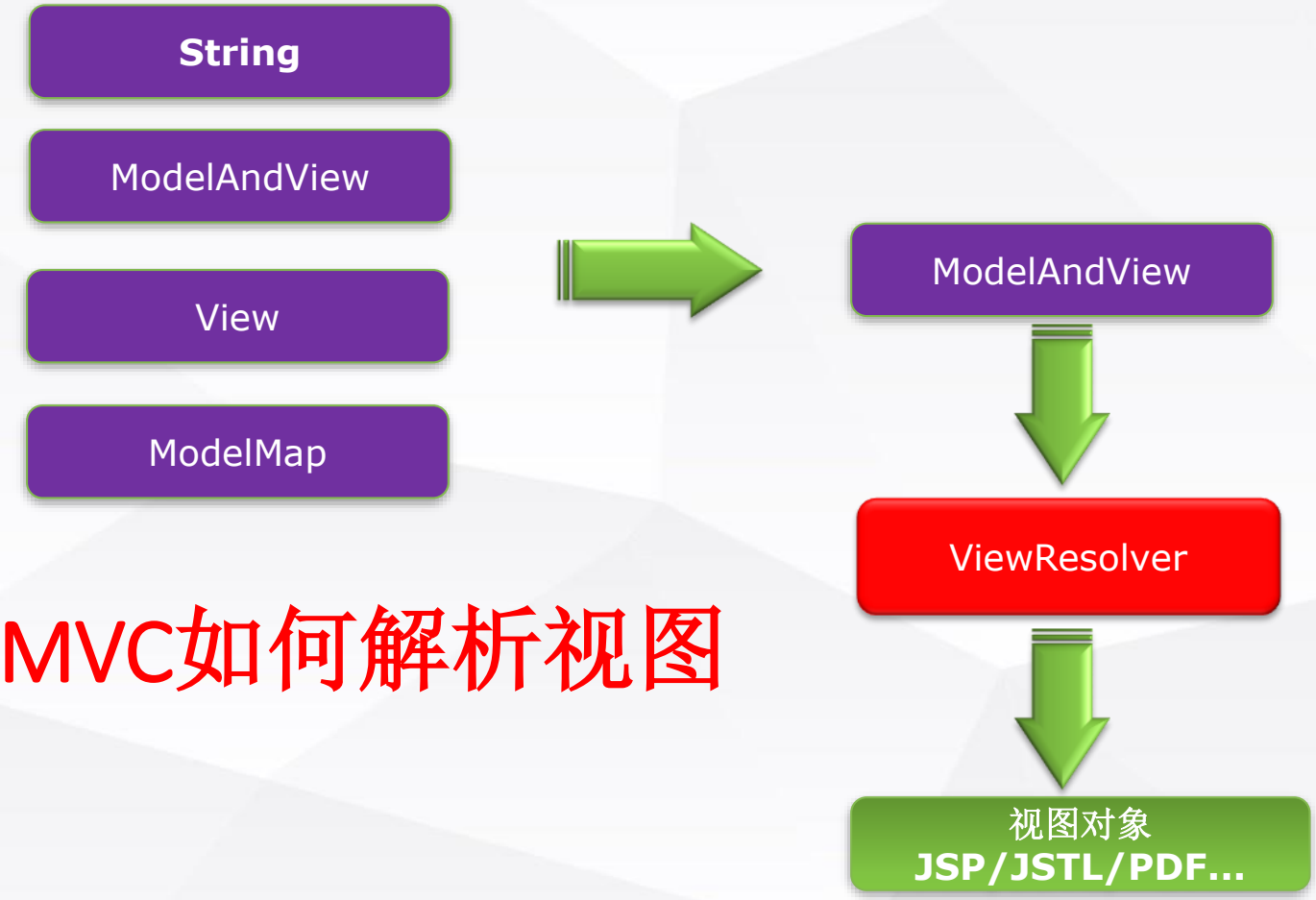
```
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
      <property name="viewClass"
        value="org.springframework.web.servlet.view.JstlView" />
      <property name="prefix" value="/WEB-INF/jsp/" />
      <property name="suffix" value=".jsp" />
</bean>
```

InternalResourceViewResolver：支持JSP视图解析

viewClass：JstlView表示JSP模板页面需要使用JSTL标签库。此属性可以不设置，默认为JstlView。

基于Controller接口的Spring MVC开发

请求处理方法返回值类型



Spring MVC如何解析视图

• 基于Controller接口的Spring MVC开发

6. View：视图

Spring MVC框架支持多种View视图，包括JSTL View、freeMarker View、PDF View等。

大都使用页面标签或页面模版技术将模型数据通过页面展示给用户，这需要根据业务需求开发具体的页面。

```
//index.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" isELIgnored="false"%>
<h1>${message}</h1>
```

• 基于Controller接口的Spring MVC开发

基于Controller接口的控制器

测试方法：

http://localhost:8080/index

IntelliJ IDEA演示

使用MVC框架就应该严格遵循MVC设计模式的思想。MVC框架不赞成浏览器直接访问Web应用的视图页面，用户的所有请求都只应项控制器发送，由控制器调用模型组件、视图组件向用户呈现数据。

• 基于Controller接口的Spring MVC开发

基于Controller控制器的执行流程：

（1）用户访问 /index；

（2）根据web.xml中的配置，所有的访问都会经过DispatcherServlet；

（3）根据配置文件springmvc-servlet.xml中的配置，前端控制器会捕获（URL Pattern）以/index开始的所有URL请求，并调用业务控制器MyController；

（4）在MyController中指定跳转到页面index.jsp（通常在创建**ModelAndView**对象时指定初始化参数的方式的 **new ModelAndView(“index.jsp”)**），并传递message数据（说明是在指定的页面中直接显示Model中的message数据）；

（5）在index.jsp中显示message信息！

课程内容安排

- Spring MVC简介
- Spring MVC体系架构
- Spring MVC工作原理
- 基于Controller接口的Spring MVC开发
- 基于注释的Spring MVC开发
- Spring MVC与Struts框架比较

• 基于注释的Spring MVC开发

基于注释的控制器Web应用开发过程：

第一步：创建Web项目并导入相应的jar包

第二步：配置前端控制器

第三步：创建控制器组件

第四步：创建index.jsp

• 基于注释的Spring MVC开发

基于注释的控制器

- 从Spring2.5版起，新增了基于注释的控制器，即控制器不用实现Controller接口，而可以通过注释类型来描述。

- 特点:1)通过大量的注释描述控制器；
  2)方便且不局限于一个控制器类只响应一个请求。

- 不同于实现接口的开发方法：
  1).通过注释实现Controller，而不用通过实现Controller接口开发控制器；
  2).控制器的配置不同。

• 基于注释的Spring MVC开发

Spring MVC框架的实现

- ✓ 基于注释的控制器通过使用@RequestMapping注释，根据请求URL，将请求映射控制器的处理方法上。
- ✓控制器也由此确定是直接在Web响应页面中写入内容，还是将请求路由到一个视图并将属性注入到该视图中，视图可由模板引擎进行渲染。

• 基于注释的Spring MVC开发

注释类控制器的结构

```
...
@Controller          //将本类注释为控制器类
@RequestMapping("/index") //指定控制器映射的URL，即哪个路径
public class UserController {
@RequestMapping(value = "/register") //处理方法对应的URL，相对于指定的URL
      public String register() {
                return "index/register"; //返回逻辑视图名
      }
}
```

• 基于注释的Spring MVC开发

为了保证Spring能够找到控制器类，还需要在Spring MVC的配置文件中添加相应的组件扫描(**component-scan**)配置信息，这可省去在spring容器中配置每个controler类的繁琐。

```
<?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:context="http://www.springframework.org/schema/context" //引入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">
    <context:component-scan base-package="com.shu.ces.controller" />//指定需要扫描的包
</beans>
```

• 基于注释的Spring MVC开发

RequestMappingHandlerMapping

注解式处理器映射器对类中标记@ResquestMapping的方法进行映射。它根据ResquestMapping定义的URL（即访问该URL）匹配ResquestMapping标记的方法（即调用标记的方法），匹配成功返回HandlerMethod对象给前端控制器，HandlerMethod对象中封装了URL对应的方法。

配置如下：

```
<!--注解映射器 -->
<bean
   class="org.springframework.web.servlet.mvc.method.annotation.Request
   MappingHandlerMapping"/>
```

• 基于注释的Spring MVC开发

RequestMappingHandlerAdapter
注解式处理器适配器，对标记@ResquestMapping的方法进行适配。
从spring3.1版本开始，采用RequestMappingHandlerAdapter完成注解式处理器适配。

配置如下：

```
<!--注解适配器 -->
<bean
class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"/>
```

基于注释的Spring MVC开发

Spring通过@Controller注解找到相应的控制器类后，还需要知道控制器内部对每一个请求是如何处理的，这就需要使用@RequestMapping注解类型，它用于映射一个请求或一个方法。使用时，可以标注在一个方法或一个类上。

1. 标注在方法上：作为请求处理方法在程序接收到对应的URL请求时被调用：

```
package com.shu.ces.controller;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
...
@Controller
public class FirstController{
    @RequestMapping(value="/firstController")
    public ModelAndView handleRequest(HttpServletRequest request,
                                      HttpServletResponse response) {
        ...
        return mav;
    }
```

此时，可以通过地址：http://localhost:8080/springMVCPro/firstController访问该方法！

基于注释的Spring MVC开发

2. 标注在类上：该类中的所有方法都将映射为相对于类级别的请求，表示该控制器所处理的所有请求都被映射到value属性值所指定的路径下。

```
package com.shu.ces.controller;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
...
@Controller
@RequestMapping(value="/hello")
public class FirstController{
    @RequestMapping(value="/firstController")
    public ModelAndView handleRequest(HttpServletRequest request,
                                      HttpServletResponse response) {
        ...
        return mav;
    }
}
```

由于在类上添加了@RequestMapping注解，并且其value属性值为“/hello”，所以上述代码方法的请求路径将变为：http://localhost:8080/springMVCPro/hello/firstController。

基于注释的控制器

- 在Spring MVC框架的所有组件中：

  处理器映射器（HandlerMapping）

  处理器适配器（HandlerAdapter）

  视图解析器（ViewResolver）

  被称为Spring MVC框架的三大组件。

基于注释的控制器

- 修改Spring MVC的配置文件springmvc-servlet.xml

```
<context:component-scan base-package=“com.shu.ces.controller”/>//扫描范围
<bean class="org.springframework.web.servlet.mvc.method.
                    annotation.RequestMappingHandlerMapping" />
<bean class="org.springframework.web.servlet.mvc.method.
                    annotation.RequestMappingHandlerAdapter"/>
<bean id="viewResolver" class="org.springframework.web.servlet.
                    view.InternalResourceViewResolver"/>
</beans>
```

基于注释的控制器

```
  <?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: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.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">
<context:component-scan base-package="com.shu.ces.controller" />

<bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping" />

<bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter" />

<bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver"/>

    <mvc:annotation-driven/>

</beans>
```

基于注释的控制器

- RequestMappingHandlerMapping 和 RequestMappingHandlerAdapter 两个 bean 是 Spring MVC为@Controller分发请求所必须的。

基于注释的控制器

- 需要注意的是：在Spring 4.0后，如果不配置处理器映射器、处理器适配器和视图解析器，也会使用默认的来完成Spring内部MVC工作。

基于注释的控制器

IntelliJ IDEA演示

处理/hello请求的Controller：

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;
//基于注释的控制器类，可以同时处理多个请求动作，并且无需实现任何接口
@Controller
public class HelloController{
//RequestMapping注释用来映射请求的URL和请求的方法。这里用来映射/hello，下面的hello只是一个普通的方法，请求由它来处理。
    @RequestMapping(value="/hello")
    public ModelAndView hello() {
 //ModelAndView对象通常包含返回视图名、模型名及模型对象
        ModelAndView mav = new ModelAndView("welcome.jsp");
        mav.addObject("message", "Hello Spring MVC");
        return mav;
    }
}
```

Spring MVC与Struts 2

Spring MVC与Struts 2的区别

拦截机制上

1. Struts2

✓Struts2是类级别的拦截，每次请求创建一个Action，和Spring整合时Struts2的ActionBean注入作用域是原型模式prototype（否则会出现线程并发问题），然后通过setter，getter将request数据注入到属性。

✓Struts2中，一个Action对应一个request，response上下文，在接收参数时，可以通过属性接收，即属性参数是让多个方法共享的。

✓Struts2中Action的一个方法可以对应一个URL，其类属性被所有方法共享，无法用注解或其它方式标识其所属方法。

拦截机制上

2. SpringMVC

✓SpringMVC是方法级别的拦截，一个方法对应一个Request上下文，所以方法之间是独立的，独享request，response数据。每个方法和一个URL对应，参数的传递是直接注入到方法中的，是方法所独有的。处理结果通过ModelMap返回给框架。

✓与Spring整合时，Spring MVC的Controller Bean默认单例模式，所以默认对所有的请求，只会创建一个Controller，又因为没有共享的属性，所以是线程安全的。

Spring MVC与Struts 2

Spring MVC与Struts 2的区别

性能方面

✓Spring MVC实现了零配置，由于Spring MVC基于方法的拦截，由加载一次单例模式bean注入。而Struts 2是类级别的拦截，每次请求对应实例化一个新的Action，需要加载所有的属性值注入，所以，Spring MVC开发效率和性能高于Struts 2。

Spring MVC与Struts 2

Spring MVC与Struts 2的区别

配置方面

- ✓Spring MVC与Spring是无缝的，项目的管理和安全上比Struts 2高。
- ✓Spring MVC可以认为已经100%零配置。
- ✓Sturts 2需要在配置文件中做Action和结果的大量配置。