企业级框架 Spring

Unit03

Sping Web MVC简介

Sping Web MVC

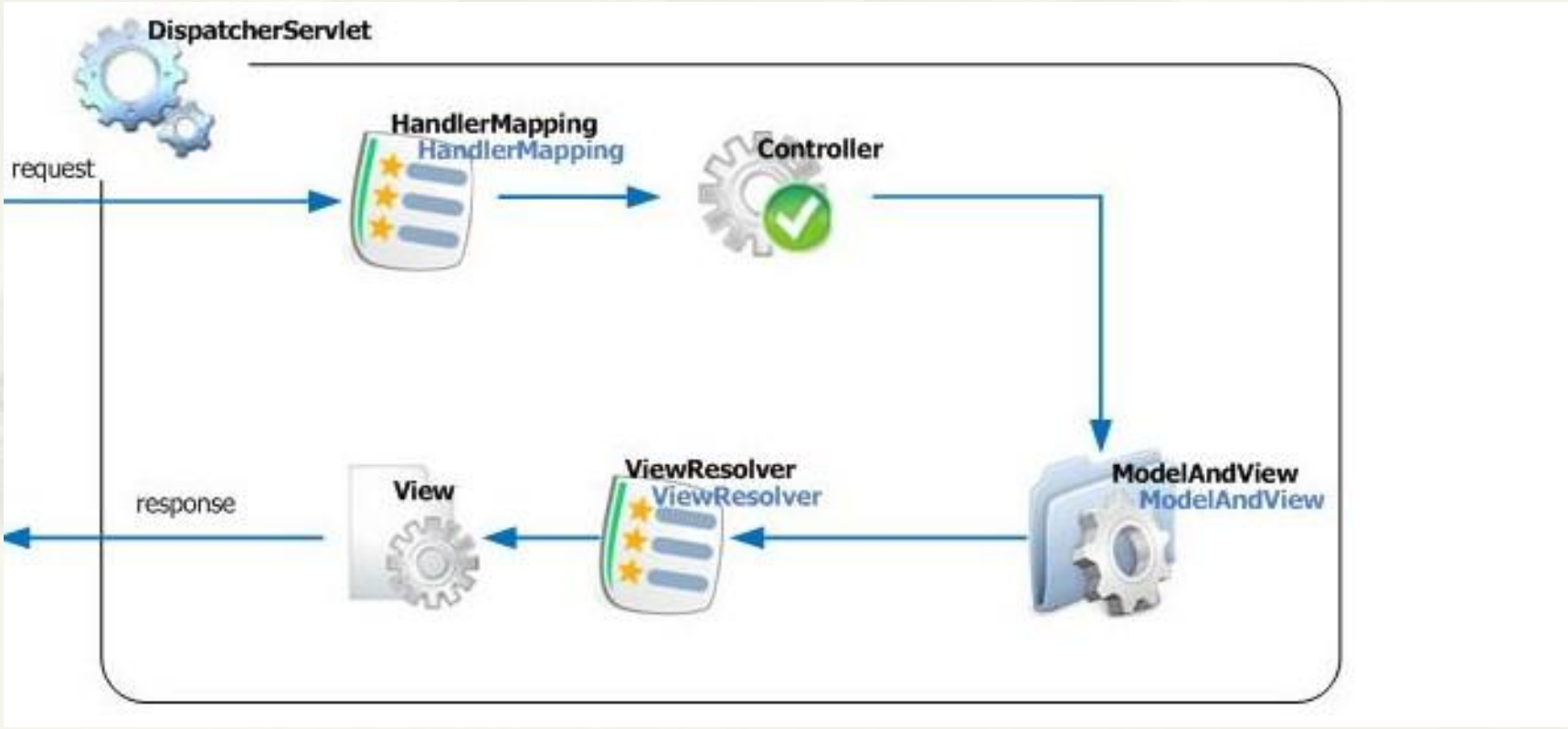
* Spring 提供了一个Web MVC框架，便于开发MVC结构的Java Web程序。
* Spring MVC框架控制器为DispatcherServlet，DispatcherServlet负责接收请求，然后将请求分发到不同的处理器进行业务处理，最后由控制器完成转发动作。

Sping Web MVC

- Spring Web MVC提供了M、V和C相关的实现，主要实现组件如下
  - DispatcherServlet（控制器, 请求入口）
  - HandlerMapping（控制器, 请求派发）
  - Controller（控制器, 请求处理流程）
  - ModelAndView（模型, 封装处理结果和视图）
  - ViewResolver（视图, 视图显示处理器）

Sping Web MVC

* Spring Web MVC主要处理流程如下



Sping Web MVC应用

Sping Web MVC应用

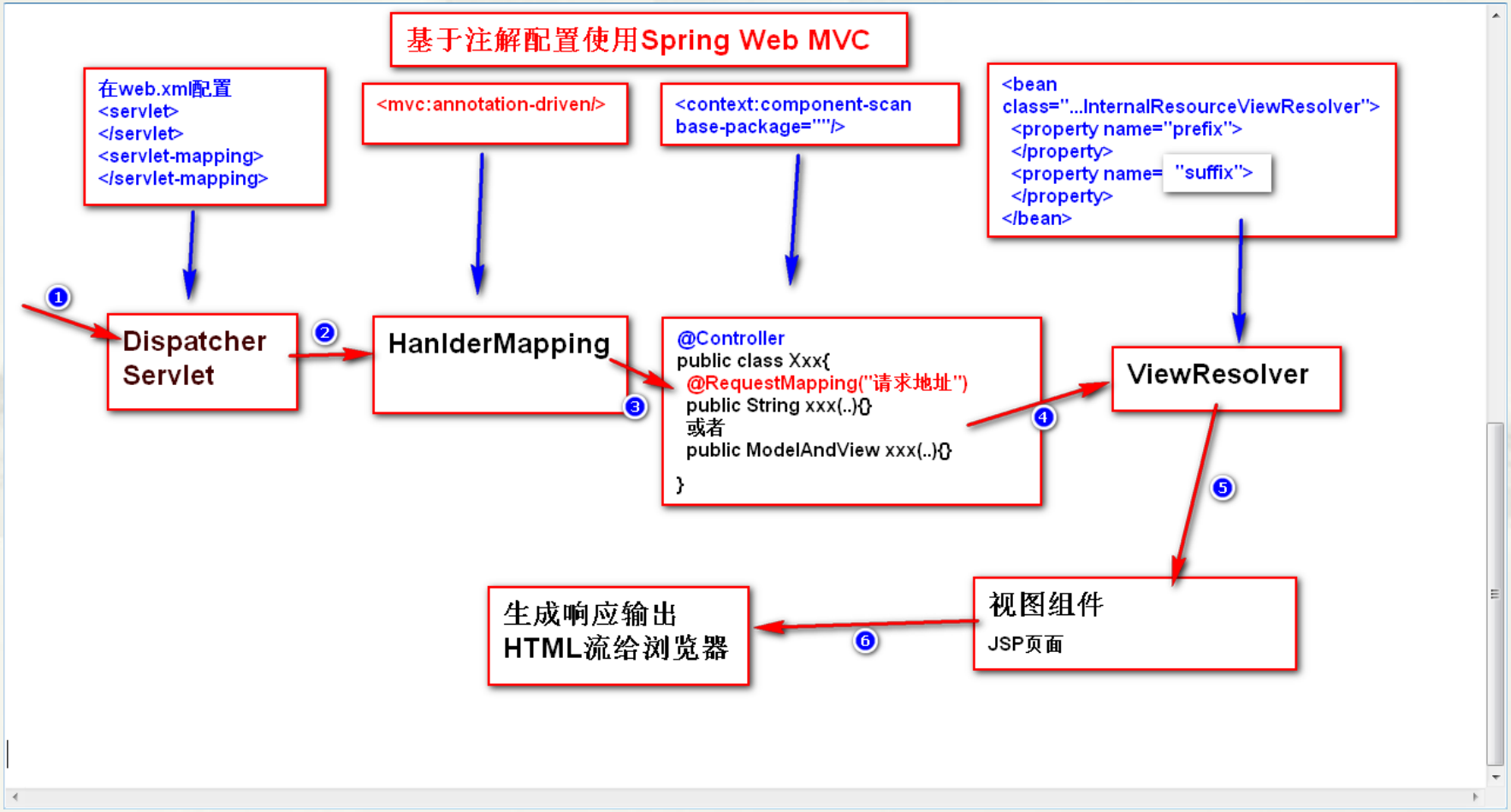
- 搭建Spring Web MVC环境的步骤如下：
  - 创建web项目工程
  - 添加ioc和webmvc开发包
  - 添加Spring配置文件applicationContext.xml
  - 在web.xml中配置DispatcherServlet控制器组件

Sping Web MVC应用

- 开发Spring Web MVC功能的步骤如下：
  - 设计请求到响应处理流程
  - 编写Controller组件和JSP页面
  - 在applicationContext.xml配置Controller组件
  - 在applicationContext.xml配置HandlerMapping组件
  - 在applicationContext.xml配置配置ViewResolver组件

Sping Web MVC应用

* 注解配置应用结构：



Sping Web MVC应用

- 在Controller组件处理中，如果需要接收请求参数，可以使用下面的方法
  - 使用HttpServletRequest获取
  - 使用@RequestParam注解
  - 使用自动机制封装成Bean对象

Sping Web MVC应用

- 当Controller组件处理后，如果需要向JSP传值，可以使用下面的方法
  - 直接使用HttpServletRequest和Session
  - 使用ModeAndView对象
  - 使用ModelMap参数对象

Sping Web MVC应用

- 在Controller组件处理中，如果需要使用Session,可以使用下面方法
  - 添加方法参数HttpServletRequest，然后使用getSession方法获取
  - 添加方法参数HttpSession，直接使用

Sping Web MVC应用

- 在Controller组件处理后，默认采用转发方式调用视图JSP页面，如果需要重定向，可以使用下面方法
  - 使用RedirectView
  - 使用redirect:前缀

Sping Web MVC应用

- Spring提供了一个CharacterEncodingFilter过滤器，可用于解决Post提交参数的乱码问题。
- CharacterEncodingFilter使用时需要注意以下问题
  - 表单数据以POST方式提交
  - 在web.xml中配置CharacterEncodingFilter过滤器
  - 页面编码和过滤器指定编码要保持一致

Sping Web MVC应用

- Spring的HandlerMapping处理器支持拦截器应用。当需要为某些请求提供特殊功能时，例如对用户进行身份认证、登录检查等
- 拦截器必须实现HandlerInterceptor接口

Sping Web MVC应用

- HandlerInterceptor接口有以下3个方法
  - preHandle()

  处理器执行前被调用。方法返回true表示会继续调用其他拦截器和处理器；返回false表示中断流程，不会执行后续拦截器和处理器
  - postHandle()

  处理器执行后、视图处理前调用。
  - afterCompletion()

  整个请求处理完毕后调用。

Sping Web MVC应用

* 自定义拦截器的配置如下

```
<mvc:interceptors>
    <mvc:interceptor>
        <mvc:mapping path="/**"/>
        <mvc:exclude-mapping path="/login.do"/>
        <bean class="com.xdl.interceptor.SomeInterceptor"/>
    </mvc:interceptor>
</mvc:interceptors>
```

Sping Web MVC应用

- Spring MVC处理异常方式有以下三种
  - 使用简单异常处理器SimpleMappingExceptionResolver，处理所有Controller异常
  - 自定义异常处理器，实现HandlerExceptionResolver接口，处理所有Controller异常
  - 使用@ExceptionHandler注解实现异常处理，处理某一个Controller异常

```
public String execute(HttpServletRequest request, Exception ex)
```

Sping Web MVC应用

- SimpleMappingExceptionResolver使用时，只需要在Spring的XML配置文件中定义下就可以了，定义示例如下。

```
<bean class="org.springframework.web.servlet.handler
 .SimpleMappingExceptionResolver">
<property name="exceptionMappings">
      <props>
         <prop key="java.lang.Exception">error1</prop>
         <prop key=“java.lang.IOException">error2</prop>
      </props>
   </property>
</bean>
```

Sping Web MVC应用

* 上传文件，Spring提供了一个CommonsMultipartResolver组件，用于整合第三方解析器完成文件上传操作

```
<bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
        <property name="maxUploadSize"  value="10240">
        </property>
        <property name="resolveLazily" value="true">
        </property>
</bean>
```

Sping Web MVC应用

* 上传文件操作步骤
  * 引入第三方解析器组件，commons-fileupload.jar和commons-io.jar
  * 在Spring配置文件中定义CommonsMultipartResolver

  提示：必须指定id=“multipartResolver”
  * 编写Controller，在controller类方法中定义MultipartFile参数，用于接收解析器解析出来的文件。

```
public String upload(@RequestParam("file") MultipartFile file)
```

提示：在JSP表单中的<form>元素一定要添加enctype=“multipart/form-data”属性

Sping Web MVC应用

- 为了便于接收和处理Ajax请求，Spring MVC提供了JSON响应的支持，可以很方便地将数据自动转成JSON格式字符串给客户端
- 与JSON响应相关的注解为@ResponseBody，该注解主要用于Controller组件的处理方法前

```
@RequestMapping("/load")
@ResponseBody
public Object f4(){
    Emp emp = new Emp();
    ... ...
    return emp;
}
```

Sping Web MVC应用

- @ResponseBody注解具体使用方法如下：
  - 引入jackson开发包，后面示例采用的是jackson-annotations-2.4.1.jar、jackson-core-2.4.1.jar 、jackson-databind-2.4.1.jar
  - 在Spring配置文件中定义<mvc:annotation-driven />，开启对@ResponseBody应用的支持
  - 在Controller处理方法前定义@ResponseBody注解

Sping Web MVC应用

* REST架构是一个抽象的概念，目前主要是基于HTTP协议实现，其目的是为了提高系统的可伸缩性、降低应用之间的耦合度、便于架构分布式处理程序。

Sping Web MVC应用

- REST主要对以下两个方面进行了规范
  - 定位资源的URL风格，例如

  http://itxdl.com/customers/1234

  http://itxdl.com/orders/2007/10/776654
  - 如何对资源操作

  采用HTTP协议规定的GET、POST、PUT、DELETE动作处理资源的增删改查操作

Sping Web MVC应用

Method	CRUD
POST	Create update, delete
GET	Read
PUT	Update create
DELETE	Delete

Sping Web MVC应用

* 什么是RESTful？

  符合REST约束风格和原则的应用程序或设计就是RESTful

/blog/1　　HTTP　GET　=>　　查询id=1的blog
/blog/1　　HTTP　DELETE　=>　删除id=1的blog
/blog/1　　HTTP　PUT　=>　　更新blog
/blog/new　HTTP　POST　=>　　新增blog

Sping Web MVC应用

- Spring MVC对RESTful应用提供了以下支持
  - 利用@RequestMapping指定要处理请求的URI模板和HTTP请求的动作类型
  - 利用@PathVariable将URI请求模板中的变量映射到处理方法的参数上

Sping Web MVC应用

- 在RESTful应用中，@RequestMapping可以采用以下使用格式：

```
@RequestMapping(value="/user/{id}", method = RequestMethod.GET)

@RequestMapping   (value="/user/new", method = RequestMethod.POST)

@RequestMapping(value="/user/{id}", method = RequestMethod.DELETE)

@RequestMapping(value="/user/{id}", method = RequestMethod.PUT)
```

Sping Web MVC应用

- @PathVariable作用是将URI请求模板中的变量解析出来，映射到处理方法的参数上，使用示例如下：

```
@RequestMapping(value="/user/{id}",method=RequestMethod.GET)
public User execute(@PathVariable("id") int id){
    //查询user操作处理
    return user;
}
```

Sping Web MVC应用

- 采用RESTful架构后，需要将web.xml中控制器拦截的请求设置为/，这样会将css,js等静态资源进行拦截，发生404错误。
- 解决上述问题的方法如下：
  - 配置<mvc:resources/>

```
<mvc:resources mapping="请求URI" location="资源位置" />
```

  - 配置<mvc:default-servlet-handler/>

```
<mvc:default-servlet-handler/>
```

总结和答疑