[今日课程大纲]

SpringMVC 简介

SpringMVC 容器和 Spring 容器介绍(源码跟踪)

使用纯配置文件方式搭建环境

基于注解方式搭建环境

接收参数的几种方式

视图解析器

@ResponseBody

[知识点详解]

一.SpringMVC 简介

1.SpringMVC 中重要组件

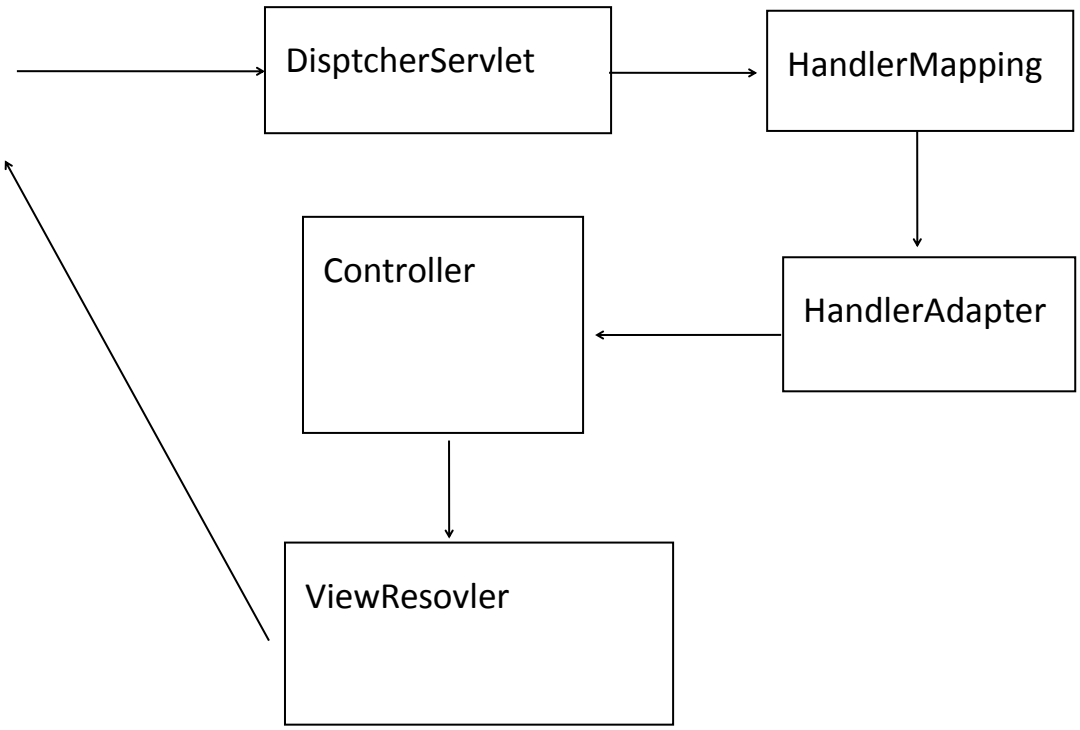
1.1 DispatcherServlet： 前端控制器,接收所有请求(如果配置/不包含 jsp)

1.2 HandlerMapping: 解析请求格式的.判断希望要执行哪个具体的方法.

1.3 HandlerAdapter: 负责调用具体的方法.

1.4 ViewResovler:视图解析器.解析结果,准备跳转到具体的物理视图

2.SpringMVC 运行原理图



3.Spring 容器和 SpringMVC 容器的关系

3.1 代码

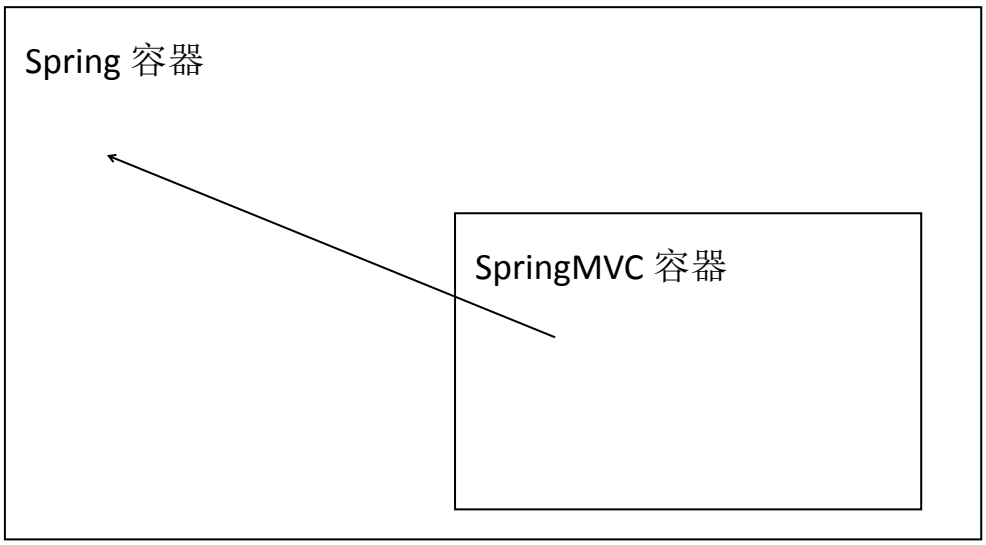
```
WebApplicationContext rootContext =
        WebApplicationContextUtils.getWebApplicationContext(getServletContext());
WebApplicationContext wac = null;

if (this.webApplicationContext != null) {
    // A context instance was injected at construction time -> use it
    wac = this.webApplicationContext;
    if (wac instanceof ConfigurableWebApplicationContext) {
        ConfigurableWebApplicationContext cwac = (ConfigurableWebApplicationContext) wac;
        if (!cwac.isActive()) {
            // The context has not yet been refreshed -> provide services such as
            // setting the parent context, setting the application context id, etc
            if (cwac.getParent() == null) {
                // The context instance was injected without an explicit parent -> set
                // the root application context (if any; may be null) as the parent
                cwac.setParent(rootContext);
            }
            configureAndRefreshWebApplicationContext(cwac);
        }
    }
}
```

3.2 Spring 容器和 SpringMVC 容器是父子容器.

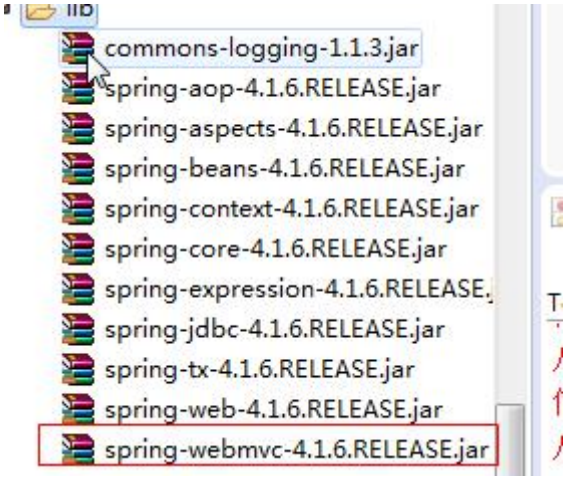
3.2.1 SpringMVC 容器中能够调用 Spring 容器的所有内容.

3.2.2 图示



二.SpringMVC 环境搭建

1. 导入 jar



2. 在 web.xml 中配置前端控制器 DispatcherServlet

2.1 如果不配置 <init-param> 会在 /WEB-INF/<servlet-name>-servlet.xml

```
<servlet>
    <servlet-name>jqk</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>jqk</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
```

3. 在 src 下新建 springmvc.xml

3.1 引入 xmlns:mvc 命名空间

```
<?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/context
        http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd">
    <!-- 扫描注解 -->
    <context:component-scan
base-package="com.bjsxt.controller"></context:component-scan>
    <!-- 注解驱动 -->
    <!--
org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandler
Mapping -->
    <!--
org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerA
dapter -->
    <mvc:annotation-driven></mvc:annotation-driven>
    <!-- 静态资源 -->
    <mvc:resources location="/js/" mapping="/js/**"></mvc:resources>
    <mvc:resources location="/css/" mapping="/css/**"></mvc:resources>
    <mvc:resources location="/images/"
mapping="/images/**"></mvc:resources>
</beans>
```

4. 编写控制器类

```
@Controller
public class DemoController {
   @RequestMapping("demo")
   public String demo(){
      System.out.println("执行 demo");
      return "main.jsp";
   }
   @RequestMapping("demo2")
   public String demo2(){
      System.out.println("demo2");
      return "main1.jsp";
   }
}
```

三.字符编码过滤器

1.在 web.xml 中配置 Filter

```
<!-- 字符编码过滤器 -->
   <filter>
      <filter-name>encoding</filter-name>

   <filter-class>org.springframework.web.filter.Charac
```

```
terEncodingFilter</filter-class>
      <init-param>
         <param-name>encoding</param-name>
         <param-value>utf-8</param-value>
      </init-param>
   </filter>
   <filter-mapping>
      <filter-name>encoding</filter-name>
      <url-pattern>/*</url-pattern>
   </filter-mapping>
```

四.传参

1. 把内容写到方法(HandlerMethod)参数中,SpringMVC 只要有这个内容,注入内容.

2. 基本数据类型参数

2.1 默认保证参数名称和请求中传递的参数名相同

```
@Controller
public class DemoController {
   @RequestMapping("demo")
   public String demo(String name,int age){
      System.out.println("执行 demo"+"  "+name+"
"+age);
```

```
        return "main.jsp";
    }
}
```

2.2 如果请求参数名和方法参数名不对应使用@RequestParam()赋值

```
    @RequestMapping("demo")
    public String demo(@RequestParam(value="name1")
String name,@RequestParam(value="age1")int age){
        System.out.println("执行 demo"+"  "+name+"
"+age);
        return "main.jsp";
    }
```

2.3 如果方法参数是基本数据类型(不是封装类)可以通过@RequestParam 设置默认值.

2.3.1 防止没有参数时 500

```
    @RequestMapping("page")
    public String page(@RequestParam(defaultValue="2")
int pageSize,@RequestParam(defaultValue="1") int
pageNumber){
        System.out.println(pageSize+" "+pageNumber);
        return "main.jsp";
    }
```

2.4 如果强制要求必须有某个参数

```
@RequestMapping("demo2")
public String demo2(@RequestParam(required=true)
String name){
    System.out.println("name 是 SQL 的查询条件,必须要传
递 name 参数"+name);
    return "main.jsp";
}
```

3. HandlerMethod 中参数是对象类型

3.1 请求参数名和对象中属性名对应(get/set 方法)

```
@RequestMapping("demo4")
public String demo4(People peo){
    return "main.jsp";
}
```

4. 请求参数中包含多个同名参数的获取方式

4.1 复选框传递的参数就是多个同名参数

```
@RequestMapping("demo5")
public String demo5(String name,int
age,@RequestParam("hover")List<String> abc){
    System.out.println(name+"  "+age+"  "+abc);
    return "main.jsp";
}
```

5. 请求参数中对象.属性格式

5.1 jsp 中代码

```
<input type="text" name="peo.name"/>
<input type="text" name="peo.age"/>
```

5.2 新建一个类

5.2.1 对象名和参数中点前面名称对应

```
public class Demo {
   private People peo;
```

5.3 控制器

```
@RequestMapping("demo6")
public String demo6(Demo demo){
   System.out.println(demo);
   return "main.jsp";
}
```

6. 在请求参数中传递集合对象类型参数

6.1 jsp 中格式

```
<input type="text" name="peo[0].name"/>
<input type="text" name="peo[0].age"/>
<input type="text" name="peo[1].name"/>
<input type="text" name="peo[1].age"/>
```

6.2 新建类

```
public class Demo {
```

```
private List<People> peo;
```

6.3 控制器

```
@RequestMapping("demo6")
public String demo6(Demo demo){
    System.out.println(demo);
    return "main.jsp";
}
```

7. restful 传值方式.

7.1 简化 jsp 中参数编写格式

7.2 在 jsp 中设定特定的格式

```
<a href="demo8/123/abc">跳转</a>
```

7.3 在控制器中

7.3.1 在@RequestMapping 中一定要和请求格式对应

7.3.2 {名称} 中名称自定义名称

7.3.3 @PathVariable 获取@RequestMapping 中内容,默认按照方法参数名称去寻找.

```
@RequestMapping("demo8/{id1}/{name}")
public String demo8(@PathVariable String name,@PathVariable("id1") int age){
    System.out.println(name +"   "+age);
    return "/main.jsp";
}
```

四.跳转方式

1. 默认跳转方式请求转发.
2. 设置返回值字符串内容

   2.1 添加 redirect:资源路径　　重定向

   2.2 添加 forward:资源路径 或省略 forward:　转发

五.视图解析器

1. SpringMVC 会提供默认视图解析器.
2. 程序员自定义视图解析器

```
   <bean id="viewResolver"
class="org.springframework.web.servlet.view.InternalR
esourceViewResolver">
      <property name="prefix" value="/"></property>
      <property name="suffix" value=".jsp"></property>
   </bean>
```

3. 如果希望不执行自定义视图解析器,在方法返回值前面添加 forward:或 redirect:

```
@RequestMapping("demo10")
public String demo10(){
    return "forward:demo11";
}

@RequestMapping("demo11")
public String demo11(){
    System.out.println("demo11");
    return "main";
}
```

六.@ResponseBody

1. 在方法上只有@RequestMapping 时,无论方法返回值是什么认为需要跳转

2. 在方法上添加@ResponseBody(恒不跳转)

   2.1 如果返回值满足 key-value 形式(对象或 map)

      2.1.1 把响应头设置为 application/json;charset=utf-8

      2.1.2 把转换后的内容输出流的形式响应给客户端.

   2.2 如果返回值不满足 key-value,例如返回值为 String

      2.2.1 把相应头设置为 text/html

      2.2.2 把方法返回值以流的形式直接输出.

      2.2.3 如果返回值包含中文,出现中文乱码

         2.2.3.1 produces 表示响应头中 Content-Type 取值.

```
@RequestMapping(value="demo12",produces="text/html;charset=utf-8")
```

```
@ResponseBody
public String demo12() throws IOException{
   People p = new People();
   p.setAge(12);
   p.setName("张三");
   return "中文";
}
```

3. 底层使用 Jackson 进行 json 转换,在项目中一定要导入 jackson 的 jar

   3.1 spring4.1.6 对 jackson 不支持较高版本,jackson 2.7 无效.