一.SpringMVC简介

1.SpringMVC中重要组件

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

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

1.3HandlerAdapter:负责调用具体的方法

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

3.Spring容器与springMVC的关系

3.1代码

3.2Spring容器和SpringMVC容器是父子容器

3.2.1SpringMVC容器能够调用Spring容器的所有内容

3.2.2图示

二.环境搭建

1.导入jar包

spring-webmvc.jar

2.在web.xml中配置前端控制器

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

```
<servlet>
   <servlet-name>dispatcherServlet</servlet-name>
   <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
   <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:com/annotation/springmvc.xml</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
</servlet>
```

```
 <servlet-mapping>
    <servlet-name>dispatcherServlet</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

```
<!--扫描注解-->
   <context:component-scan base-package="com.annotation.controller">
</context:component-scan>
   <!--注解驱动-->
   <!--mvc.annotation.DefaultAnnotationHandlerMapping-->
   <!--mvc.annotation.AnnotationMethodHandlerAdapter-->
   <mvc:annotation-driven></mvc:annotation-driven>
   <!--静态资源-->
<!--    <mvc:resources location="/js/" mapping="/js/*" ></mvc:resources>-->
   <mvc:resources mapping="/js/**/**" location="/js/"></mvc:resources>
   <mvc:resources mapping="/css/**" location="/css/"></mvc:resources>
   <mvc:resources mapping="/img/**" location="/img/"></mvc:resources>
```

3.2编写控制器类

```
@org.springframework.stereotype.Controller
public class DemoController implements Controller {
   @RequestMapping("demo")
   public String Demo(){
      System.out.println("执行demo");
      return "/Annotation/main.jsp";
   }

   @Override
   public ModelAndView handleRequest(HttpServletRequest request,
HttpServletResponse response) throws Exception {
      return null;
   }
}
```

三.字符编码过滤器

1.在web.xml中配置filter

```
<filter>
   <filter-name>encoding</filter-name>
   <filter-class>org.springframework.web.filter.CharacterEncodingFilter</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只要有这个内容就会注入内容(也可以是对象,自动进行匹配。或者是Servlet得参数(Request或Response))

2.基本数据类型参数

2.1默认保证参数名称和请求传递的参数名相同

2.2如果请求参数名和方法参数名不同应使用@RequestParam(value="")

```
public String Demo(People peo,@RequestParam(value = "name1") String
name,@RequestParam(value="age1") int age, HttpServletRequest request){
    System.out.println(peo+name+age);
    request.setAttribute("demo123","测试");
    return "/param/index.jsp";
}
```

2.3如果方法参数是基本数据类型(不是封装类)可以通过注解

@RequestParam(defaultValue="")设置默认值

2.3.1没有参数时 400错误505错误

```
public String page(@RequestParam(defaultValue ="2") int
pageSize,@RequestParam(defaultValue = "1") int pageNumber){
```

2.4强制要求必须有某个参数 @RequestParam(required="")

```
public String demo2(@RequestParam(required = true) String name){
```

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)

4.请求参数中包含多个同名参数的获取方法

4.1复选框传递的参数就是多个同名参数

```
@RequestParam(value = "hover") List<String> list){
```

5.请求参数中是对象或属性格式

5.1jsp代码

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

5.2新建一个类

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

5.3.控制器

```
public String demo5(Demo demo){
```

6.在请求参数中传递集合对象类型参数

6.2jsp代码

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

6.3新建类

```
public class Demo {
  private List<People> peo;
```

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="/SpringMVC/demo8/abc/123" >跳转3</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中获取内容，默认按照参数名称

去寻找

```
@RequestMapping("demo8/{name1}/{id}")//名称可以任意取
public String demo8(@PathVariable(value = "name1") String name,@PathVariable
int id){
```

五.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(默认)

```
return "redirect:/param/index.jsp";
```

六.视图跳转方式

1.SpringMV会提供默认的视图解析器

2.程序员自定义视图解析器

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

3.如果不希望执行视图解析器,在方法返回值前添加forward:或者redirect:

七.@ResponseBody

1.在方法上只有@ResponseMapping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把对应响应头设置为application/json;charset=utf-8

2.1.2把转换后的内容以输出流的形式响应给客户端

2.2如果返回值类型不满足key-value.例如返回值为String

2.2.1把响应头设置为text/html

2.2.3把方法返回值以流的形式直接输出

```
@RequestMapping(value="demo12")
@ResponseBody
public People demo12() throws IOException {
    People people=new People();
    people.setName("张三");
    people.setAge(12);
    return people;
}

@RequestMapping(value="demo13",produces = "text/html;charset=utf-8")
@ResponseBody
public String demo13() throws IOException {
   String str="hahahhah";
   return str;
}
```

八.SpringMVC作用域传值的几种方式

1.使用原生servlet

1.1在HandlerMethod参数中添加作用域对象

2使用map集合

2.1.把map中内容放在request作用域中

2.2.spring会对map集合通过BindingAwareModelMap进行实例化

3使用SpringMVC中的model接口

3.1把内容最终放入到request作用域里

4.使用SpringMVC中ModelAndView类

5.使用注解

九.文件下载

三种方式(直接访问,inline,attachment.其中直接访问和inline受类型影响)

1.访问资源时响应头如果没有设置Content-Disposition,浏览器默认按照inline值进

行处理

1.1inline能显示就显示,不能显示就下载

2.只需要修改响应头中Content-Disposition="attachment;filename=文件名"

2.1attachment下载,以附件形式下载.

2.2filename=值就是下载时显示的下载文件名

3.实现步骤

3.1导入两个jar包 commons-io

3.2在jsp中添加超链接,设置要下载文件(一般不需要)

3.2.1在sprinmvc中放行静态资源files文件夹

3.3编写控制器方法

十.文件上传

1.基于apache的commons-fileupload.jar完成文件下载

2.MuiltipartResovler作用

2.1把客户端上传的文件流转换为MuilpartFile封装类

2.2通过MuilpartFile封装类获取到文件流

3.表单数据类型的分类(提交为post的话,提交上限为2GB,get为2kb)

3.1在form的entype属性控制表单的类型

3.2默认值:application/x-www-form-urlencoded,普通表单数据(上传少量文字

信息)

3.3text/plain 大文字量时使用的类型,邮件,论文

3.4muitipart/form-data 表单中包含二进制文件内容

4.实现步骤

4.1导入两个jar包commons-io和commons-fileupload

4,2编写jsp代码

4.3配置springmvc.xml

4.4编写控制器类

4.4.1MultipartFile对象名必须和表单里的属性名相同