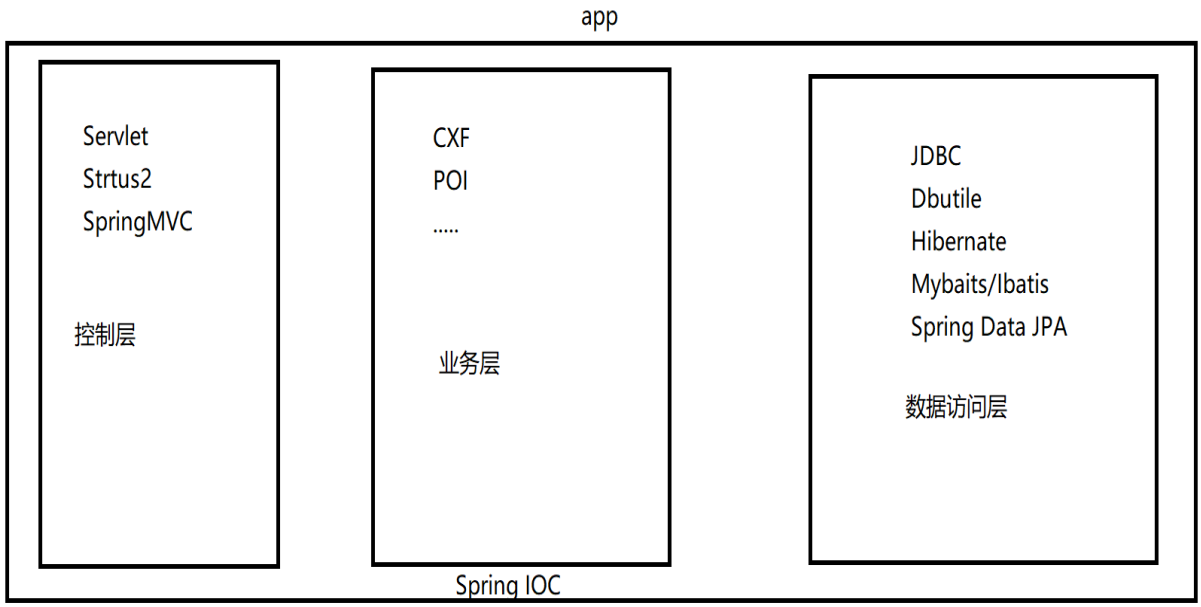
SpringMVC 系列课程

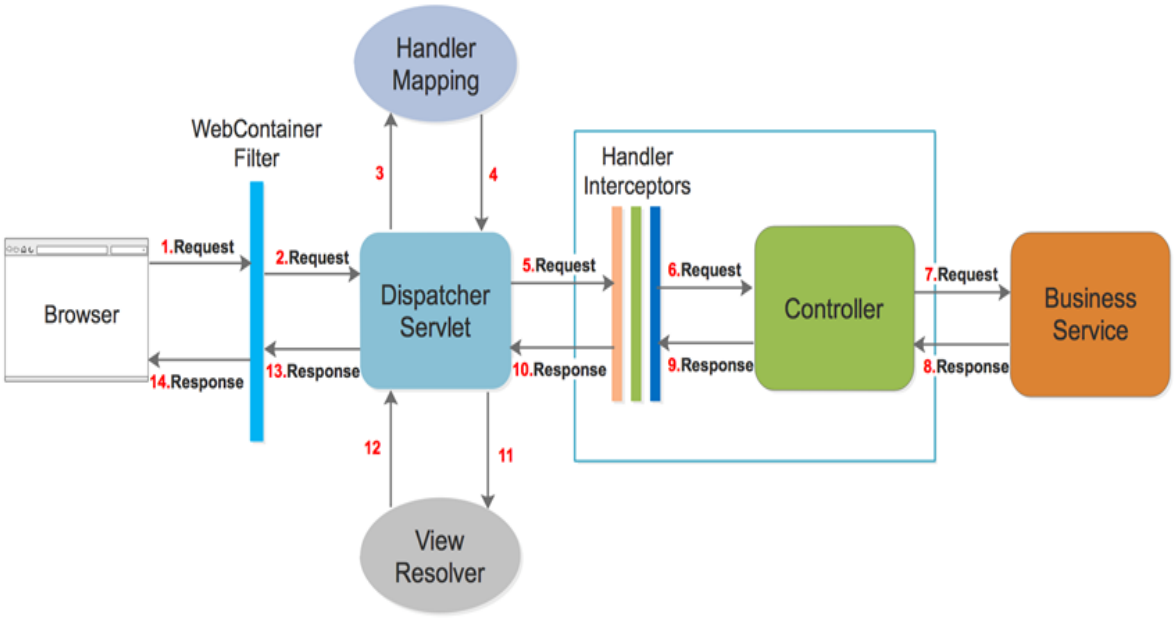
时间: 2017/11/14
作者:袁毅雄
邮箱: 896778954@qq.com

目录

定位

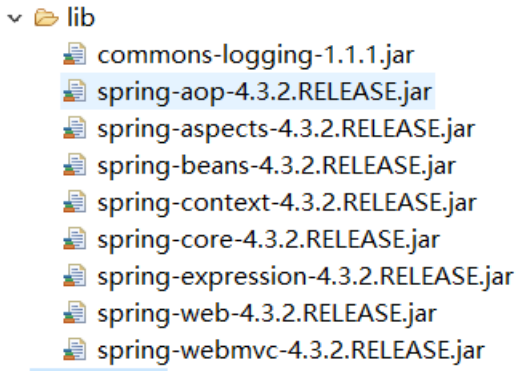


原理



搭建项目骨架

导包



web.xml

```
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <!-- http://localhost:80/News/springmvc/ -->
        <url-pattern>/springmvc/*</url-pattern>
    </servlet-mapping>
    <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-servlet.xml</param-value>
        </init-param>
    </servlet>
```

springmvc-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:context="http://www.springframework.org/schema/context"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-4.0.xsd ">
    <!-- 扫描包 -->
    <context:component-scan base-package="online.qsx.controller"/>
    <!-- 启用MVC注解,默认使用 DefaultAnnotationHandlerMapping -->
    <mvc:annotation-driven/>
    <!-- 视图解析器 -->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="suffix" value=".jsp"/>
        <property name="prefix" value="/"/>
    </bean>
</beans>
```

MyController.java

```
package online.qsx.controller;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;
@Controller
```

```java
@RequestMapping(value = "demo")
// http://localhost:80/News/springmvc/demo
public class MyController {
     // http://localhost:80/News/springmvc/demo/index
     @RequestMapping(value = "index", method = RequestMethod.GET)
     public ModelAndView to_index() {
          System.out.println("MyController....");
          return new ModelAndView("index");
     }
}
```

数据绑定

基本数据类型

```java
// dataTypeBase?age=12&avg=50
     @RequestMapping(value = "dataTypeBase", method = RequestMethod.GET)
     public ModelAndView dataTypeBase(int age,double avg) {
          System.out.println("age:"+age);
          System.out.println("avg:"+avg);
          return new ModelAndView("index");
     }
```

引用数据类型

```java
// dataTypeRef?age=12&avg=50
     @RequestMapping(value = "dataTypeRef", method = RequestMethod.GET)
     public ModelAndView dataTypeRef(Integer age,Double avg) {
          System.out.println("age:"+age);
          System.out.println("avg:"+avg);
          return new ModelAndView("index");
     }
```

@RequestParam

```java
// dataTypeBase?age=12&avg=50
     @RequestMapping(value = "dataTypeBase", method = RequestMethod.GET)
     public ModelAndView dataTypeBase(
```

```
            @RequestParam(defaultValue="18",name="a") int age,
            @RequestParam(defaultValue="50",name="b") double avg
    ) {
        System.out.println("age:"+age);
        System.out.println("avg:"+avg);
        return new ModelAndView("index");
    }

    // dataTypeRef?age=12&avg=50
    @RequestMapping(value = "dataTypeRef", method = RequestMethod.GET)
    public ModelAndView dataTypeRef(
            @RequestParam(required=true) Integer age,
            @RequestParam(required=true) Double avg
    ) {
        System.out.println("age:"+age);
        System.out.println("avg:"+avg);
        return new ModelAndView("index");
    }
```

自定义对象类型

```
package online.qsx.controller;
public class Student {
    private int id;
    private String name;
    //此处省略get方法,set方法,构造方法
    @Override
    public String toString() {
        return "Student [id=" + id + ", name=" + name + "]";
    }
}
```

```
// dataTypeObject1?id=12&name=arvin
    @RequestMapping(value = "dataTypeObject1", method = RequestMethod.GET)
    public ModelAndView dataTypeObject1(Student student) {
        System.out.println(student);
        return new ModelAndView("index");
    }
```

自定义对象名称冲突

```
package online.qsx.controller;
public class Student {
    private int id;
    private String name;
    //此处省略get方法,set方法,构造方法
    @Override
    public String toString() {
        return "Student [id=" + id + ", name=" + name + "]";
    }
}
```

```
package online.qsx.controller;
public class Teacher {
    private int id;
    private String name;
    //此处省略get方法,set方法,构造方法
    @Override
    public String toString() {
        return "Teacher [id=" + id + ", name=" + name + "]";
    }
}
```

```
// dataTypeObject2?stu.id=1&stu.name=arvin&tea.id=2&tea.name=jack
    @RequestMapping(value = "dataTypeObject2", method = RequestMethod.GET)
    public ModelAndView dataTypeObject2(Student student, Teacher teacher) {
        System.out.println(student);
        System.out.println(teacher);
        return new ModelAndView("index");
    }

    //扩展DataBinder
    @InitBinder(value="student")
    public void initStudent(WebDataBinder webDataBinder){
        webDataBinder.setFieldDefaultPrefix("stu.");
    }

    @InitBinder(value="teacher")
    public void initTeacher(WebDataBinder webDataBinder){
        webDataBinder.setFieldDefaultPrefix("tea.");
    }
```

集合类型

List

```
package online.qsx.controller;
import java.util.List;
public class Param {
	private List<String> list;

	//此处省略 get 方法,set 方法,构造方法
}
```

```
// dataTypeList?list[0]=a&list[1]=b
    @RequestMapping(value = "dataTypeList", method = RequestMethod.GET)
    public ModelAndView dataTypeList(Param param) {
        for (String temp : param.getList()) {
            System.out.println(temp);
        }
        return new ModelAndView("index");
    }
```

Set

```
package online.qsx.controller;
import java.util.Set;
public class Param {
	private Set<String> set;
	//此处省略 get 方法,set 方法,构造方法
}
```

```
// dataTypeSet?set=a&set=b&set=c
    @RequestMapping(value = "dataTypeSet", method = RequestMethod.GET)
    public ModelAndView dataTypeSet(Param param) {
        for (String temp : param.getSet()) {
            System.out.println(temp);
        }
        return new ModelAndView("index");
    }
```

Map

```
package online.qsx.controller;
import java.util.Map;
public class Param {
	private Map<String, String> map;

	//此处省略 get 方法,set 方法,构造方法
}
```

```
// dataTypeMap?map[a]=aa&map[b]=bb
    @RequestMapping(value = "dataTypeMap", method = RequestMethod.GET)
    public ModelAndView dataTypeMap(Param param) {
        for (String key : param.getMap().keySet()) {
            System.out.println(key+":"+param.getMap().get(key));
        }
        return new ModelAndView("index");
    }
```

多种返回结果

String

```
// to_index_string
    @RequestMapping(value = "to_index_string", method = RequestMethod.GET)
    public String to_index_string(Model model) {
        System.out.println("to_index_string");


        model.addAttribute("mess", "to_index_string");
        model.addAttribute("obj",new Student(1,"arvin"));

        List<Student> students=new ArrayList<Student>();
        students.add(new Student(1,"arvin"));
        students.add(new Student(2,"jack"));
        model.addAttribute("list",students);

        return "index";
    }
```

ModelAndView

```
// to_index_modelandview
    @RequestMapping(value = "to_index_modelandview", method =
RequestMethod.GET)
    public ModelAndView to_index_modelandview() {

        ModelAndView mav=new ModelAndView("index");

        mav.addObject("mess", "to_index_modelandview");
        mav.addObject("obj",new Student(1,"arvin"));

        List<Student> students=new ArrayList<Student>();
        students.add(new Student(1,"arvin"));
        students.add(new Student(2,"jack"));
        mav.addObject("list",students);

        return mav;
    }
```

AJAX/JSON

```
    //(异步)AJAX 只能返回数据  js css jquery UI
    //(同步)返回的是静态页面
    //一次请求,只会响应一次结果
    // get_json
    @ResponseBody
    @RequestMapping(value = "get_json", method = RequestMethod.GET)
    public String json(){
        return "[{'id':'12','name':'arvin'},{'id':'13','name':'jack'}]";
    }
```

转发: forward

```
    // '/' = WebContent
    @RequestMapping(value = "to_forward_string_jsp", method = RequestMethod.GET)
    public String to_forward_string_jsp() {
        return "forward:/index.jsp";
    }
```

```
    // '/' = WebContent
    @RequestMapping(value = "to_forward_modelandview_jsp", method = RequestMethod.GET)
    public ModelAndView to_forward_modelandview_jsp() {
        return new ModelAndView("forward:/index.jsp");
    }

    // '/' = http://localhost:80/News/
    @RequestMapping(value = "to_forward_string_controller", method =
RequestMethod.GET)
    public String to_forward_string_controller() {
        return "forward:to_index";
    }

    // '/' = http://localhost:80/News/
    @RequestMapping(value = "to_forward_modelandview_controller", method =
RequestMethod.GET)
    public ModelAndView to_forward_modelandview_controller() {
        return new ModelAndView("forward:to_index");
    }
```

重定向

带参数

注意事项

注意：

1.使用 RedirectAttributes 的 addAttribute 方法传递参数会跟随在 URL 后面 ，如上代码即为 http:/index.action?a=a

2.使用 addFlashAttribute 不会跟随在 URL 后面，会把该参数值暂时保存于 session，待重定向 url 获取该参数后从 session 中移除，这里的 redirect 必须是方法映射路径，jsp 无效。你会发现 redirect 后的 jsp 页面中 b 只会出现一次，刷新后 b 再也不会出现了，这验证了上面说的，b 被访问后就会从 session 中移除。对于重复提交可以使用此来完成.

3.另外，如果使用了 RedirectAttributes 作为参数，但是没有进行 redirect 呢？这种情况下不会将 RedirectAttributes 参数传递过去，默认传 forward 对应的 model

案例

```
// '/' = http://localhost:80/News/
    @RequestMapping(value = "/testRedirect_controller", method = RequestMethod.GET)
    public String testRedirect_controller(RedirectAttributes attr) {
        attr.addAttribute("name", "url");// 地址栏中
        attr.addFlashAttribute("age", "session");// 会话中
        return "redirect:to_index";
    }
```

不带参数

```
// '/' = http://localhost:80/News/
    @RequestMapping(value = "/testRedirect_controller", method = RequestMethod.GET)
    public String testRedirect_controller() {
        return "redirect:to_index";
    }
```

文件上传

同步文件上传

commons-io-2.4.jar
commons-fileupload-1.2.2.jar
commons-logging-1.1.1.jar

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:context="http://www.springframework.org/schema/context"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
           http://www.springframework.org/schema/mvc
           http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd
           http://www.springframework.org/schema/context
           http://www.springframework.org/schema/context/spring-context-4.0.xsd ">

    <!-- 扫描包 -->
```

```
    <context:component-scan base-package="online.qsx.controller" />

    <!-- 启用MVC注解,默认使用 DefaultAnnotationHandlerMapping -->
    <mvc:annotation-driven />

    <!-- 视图解析器 -->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="suffix" value=".jsp" />
        <property name="prefix" value="/" />
    </bean>

    <!-- 文件下载 -->
    <bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
        <property name="maxUploadSize" value="10240000000"/>
        <property name="defaultEncoding" value="utf-8"/>
    </bean>
</beans>
```

```
// 单文件上传
    //http://localhost:80/News/springmvc/demo/fileupload_single
    @RequestMapping(value = "fileupload_single", method = RequestMethod.POST)
    public ModelAndView fileupload(MultipartFile file, HttpServletRequest request) {
        String url = request.getSession().getServletContext().getRealPath("upload");

        // 如果目录不存在我就创建
        if (!new File(url).isDirectory()) {
            new File(url).mkdirs();
        }

        StringBuffer fileName = new StringBuffer(UUID.randomUUID().toString());
        fileName.append(file.getOriginalFilename());

        try {
            file.transferTo(new File(url + File.separator + fileName));
        } catch (Exception e) {
            e.printStackTrace();
        }
        return new ModelAndView("index");
    }
```

异步文件上传

springmvc-servlet.xml

```
    <!-- 文件上传 -->
    <bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
        <property name="maxUploadSize" value="1024000000000"/>
        <property name="defaultEncoding" value="UTF-8" />
    </bean>
```

UploadFileController

```
// 异步文件上传
    @ResponseBody
    @RequestMapping(value = "do_uploadFile_ajax", method = RequestMethod.POST)
    public String do_uploadFile_ajax(MultipartFile file, HttpServletRequest request)
{

        boolean mess=false;
        // 解决文件名冲突
        String fileName = file.getOriginalFilename();
        String newFileName = UUID.randomUUID().toString()+
fileName.substring(fileName.indexOf("."), fileName.length());
        String url = request.getServletContext().getRealPath("/upload");
        System.out.println(url);
        if (!new File(url).exists()) {
            new File(url).mkdirs();
        }
        try {
            FileUtils.copyInputStreamToFile(file.getInputStream(), new File(url +
File.separator + newFileName));
            mess=true;
        } catch (IOException e) {
            e.printStackTrace();
            mess=false;
        }



        JSONObject jsonObject=new JSONObject();
```

```
        jsonObject.put("mess", mess);
        jsonObject.put("url", File.separator+"upload"+File.separator + newFileName);
        return jsonObject.toJSONString();
    }
```

upload.html[JS]

```
<script type="text/javascript">
    function upload(){
        var fromData=new FormData();//js 的 from
        fromData.append("file",$("#file")[0].files[0]);
        var opt={
                url: "do_uploadFile_ajax",
                type:"post",
                data:fromData,//文件
                dataType:"json",
                processData:false,
                contentType:false,
                success: function(json){
                    var img="<img src='http://localhost/UploadSystem"+json.url+"'
width='100' height='100'/>";
                    $("#img_div").html(img);
                }
        }
        $.ajax(opt);
    }
</script>
```

upload.html[HTML]

```
<div id="img_div" style="width: 100px;height: 100px;"></div>
    <form>
        <input id="file" type="file"/>
        <input type="button" value="异步上传" onclick="upload()"/>
    </form>
```

文件下载

commons-io-2.4.jar

commons-fileupload-1.2.2.jar

commons-logging-1.1.1.jar

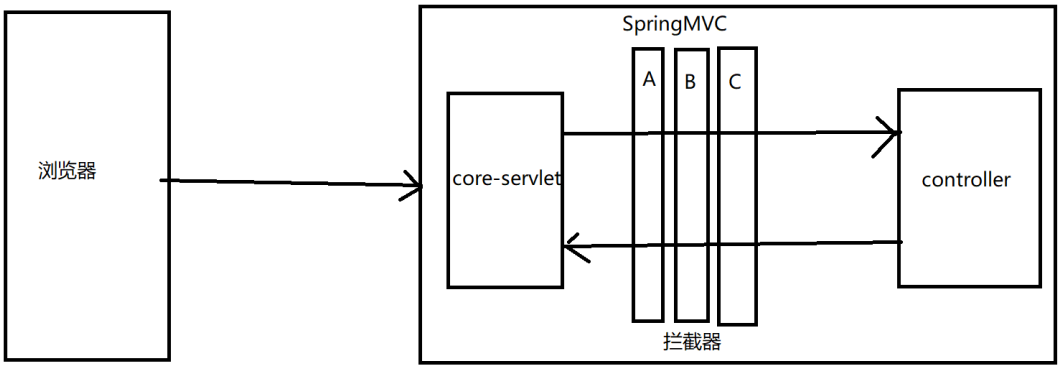
```java
// http://localhost:80/News/springmvc/demo/download
    @RequestMapping(value = "download", method = RequestMethod.GET)
    public ResponseEntity<byte[]> download() throws Exception {
        //将文件转换为字节
        byte[] file=FileUtils.readFileToByteArray(new File("G:/Java课程练习.xlsx"));
        //将文件名进行转码
        String fileName = new String("Java课程练习.xlsx".getBytes("UTF-8"), "ISO-8859-1");

        //构建响应头
        HttpHeaders headers = new HttpHeaders();
        //设置下载的附件名称
        headers.setContentDispositionFormData("attachment", fileName);
        //设置响应的内容为字节流
        headers.setContentType(MediaType.APPLICATION_OCTET_STREAM);

        //组装响应的实体
        return new ResponseEntity<byte[]>(file,headers, HttpStatus.CREATED);
    }
```

拦截器

原理



实现方案

Interceptors 配置

```
<!-- 拦截器 -->
    <mvc:interceptors>
        <mvc:interceptor>
            <mvc:mapping path="/**" />
            <bean class="online.qsx.interceptors.MyInterceptor"/>
        </mvc:interceptor>
    </mvc:interceptors>
```

MyInterceptor

```
package online.qsx.interceptors;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.HandlerInterceptor;
import org.springframework.web.servlet.ModelAndView;

public class MyInterceptor implements HandlerInterceptor    {

    @Override
    public void afterCompletion(HttpServletRequest arg0, HttpServletResponse arg1, Object arg2, Exception arg3)
                throws Exception {
        System.out.println("afterCompletion.....");
        //无所谓请求是否正常结束都会调用

        //异常处理
    }

    @Override
    public void postHandle(HttpServletRequest arg0, HttpServletResponse arg1, Object arg2, ModelAndView arg3)
                throws Exception {
        System.out.println("postHandle.....");
        //请求正常结束调度的后续处理
        //加密
```

```
    }

    @Override
    public boolean preHandle(HttpServletRequest arg0, HttpServletResponse arg1, Object arg2) throws Exception {
        System.out.println("preHandle.....");
        //请求服务前调度的方法，true,调度后面的拦截器或控制器/false,终止后续调度
        //解密
        return true;
    }

}
```

国际化

Interceptors 配置

```
    <!-- 存储区域设置信息 -->
    <bean id="localeResolver" class="org.springframework.web.servlet.i18n.SessionLocaleResolver" />
    <!-- 国际化资源文件 -->
    <bean id="messageSource"
class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
        <property name="basename" value="classpath:messages" />
    </bean>

    <mvc:interceptors>
        <bean id="localeChangeInterceptor"
class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor">
            <property name="paramName" value="lang" />
        </bean>
    </mvc:interceptors>
```

messages_zh_CN.properties

name=阿文

messages_en_US.properties

name=arvin

jsp

```
<%@taglib uri="http://www.springframework.org/tags" prefix="spring" %>
<spring:message code="name"/>
```

Servlet API 注入

```
// http://localhost:80/UploadSystem/springmvc/uploadFile/to_index
    @RequestMapping(value = "to_index", method = RequestMethod.GET)
    public String to_index(
             HttpServletRequest request,
            HttpServletResponse response,
            HttpSession session,
            Locale locale,
            InputStream inputStream,
            //OutputStream outputStream,
            //Reader reader,
            Writer writer,
            Principal principal
              ) {

         System.out.println(request);
        System.out.println(response);
        System.out.println(session);
        System.out.println(locale);
        System.out.println(inputStream);
        //System.out.println(outputStream);
        //System.out.println(reader);
        System.out.println(writer);
        System.out.println(principal);

         System.out.println("to_index");
         return "upload";
    }
```

异常处理

全局异常

```
<!-- 全局异常 -->
    <bean class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
        <property name="exceptionMappings">
            <props>
                <prop key="java.lang.Exception">error1</prop>
            </props>
        </property>
    </bean>
```

局部异常

```
    @ExceptionHandler(value = { ArithmeticException.class,
NullPointerException.class })
    public ModelAndView disposeException(Exception ex) {
        if(ex instanceof ArithmeticException){
            System.out.println("ArithmeticException...");
            return new ModelAndView("error1");
        }else if(ex instanceof NullPointerException){
            System.out.println("NullPointerException...");
            return new ModelAndView("error2");
        }
        return new ModelAndView("error2");
    }
```