1 文件上传

1）导入springMVC上传文件所依赖的第三方jar包

```
<dependency>
  <groupId>commons-fileupload</groupId>
  <artifactId>commons-fileupload</artifactId>
  <version>1.3.3</version>
</dependency>
```

2）SpringMVC的配置文件中，配置上传解析器

```
<bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
       <property name="maxUploadSize" value="10485760"/>
       <property name="defaultEncoding" value="UTF-8"/>
</bean>
```

3）编写文件上传界面

```
<form action="upload" method="post" enctype="multipart/form-data">
  <input type="file" name="f1" />
  <input type="submit"/>
</form>
```

4）编写请求处理方法

```
@RequestMapping(value="/upload",method=RequestMethod.POST)
public String upload(@RequestParam("f1") MultipartFile file) throws Exception {
   if(!file.isEmpty()) {
       String fileName=file.getOriginalFilename();
       File target=new File("c://temp//"+fileName);
       file.transferTo(target);
       System.out.println("success");
       return "success.jsp";
    }
    return "error.jsp";
}
```

2 文件下载

1）下载页面中先导入List、ArrayList、File、JSTL

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ page import="java.util.List" %>
<%@ page import="java.util.ArrayList" %>
<%@ page import="java.io.File" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
```

2）编写下载页面内容

```
<%
  List<String> fileNames = new ArrayList();
  File f = new File("c://temp");
  String[] names = f.list();
  for(int i = 0; i < names.length; i ++) {
    fileNames.add(names[i]);
  }
  request.setAttribute("fileNames", fileNames);
%>

<c:forEach items="${fileNames}" var="x">
  <a href="download?fileName=${x}">${x}</a><br>
</c:forEach>
```

3）修改web.xml，即把前3行：

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >
```

替换为：

```
<?xml version="1.0" encoding="UTF-8"?>
```

4）编写请求处理方法

```
@RequestMapping(value="/download")
```

```
protected void download(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
  String fileName = request.getParameter("fileName");
  fileName = new String(fileName.getBytes("iso-8859-1"),"utf-8");
  File f = new File("c://temp//"+fileName);
  if(f.exists()) {
    response.setHeader("content-disposition", "attachment;filename="+fileName);
    FileInputStream in = new FileInputStream(f);
    OutputStream out = response.getOutputStream();
    byte[] buffer = new byte[1024];
    int len=0;
    while((len=(in.read(buffer)))>0) {
      out.write(buffer, 0, len);
    }
    out.flush();
    out.close();
    in.close();
  }else {
    response.sendRedirect("download.jsp");
  }
}
```

---

3 拦截器

SpringMVC提供了Interceptor拦截器，用于拦截用户请求并进行相应的处理，
如权限认证、判断用户是否登录等，类似于JavaWeb的过滤器Filter

步骤：
1）定义拦截器，实现HandlerInterceptor接口

```
public class MyInterceptor1 implements HandlerInterceptor {
//  拦截器预处理
  public boolean preHandle(HttpServletRequest req, HttpServletResponse res, Object
obj) throws Exception {
    System.out.println("MyInterceptor1-预处理");
    return true;
  }
//  拦截器后处理
  public void postHandle(HttpServletRequest req, HttpServletResponse res, Object obj,
ModelAndView mv)
      throws Exception {
    System.out.println("MyInterceptor1-后处理");
  }
//  拦截结束
  public void afterCompletion(HttpServletRequest req, HttpServletResponse res, Object
obj, Exception ex)
      throws Exception {
    System.out.println("MyInterceptor1-拦截结束");
  }
}
```

2）配置全局拦截器

```
<mvc:interceptors>
    <mvc:interceptor>
        <mvc:mapping path="/**"/>
        <bean class="com.qst.inter.MyInterceptor1"/>
    </mvc:interceptor>
</mvc:interceptors>
```

注意："/**"表示所有url路径（包括子url路径）

3）访问下载页面，拦截 @RequestMapping(value = "/upload", method = RequestMethod.POST)

4）结论：拦截器是SpringMVC的技术，用来拦截 SpringMVC的@RequestMapping注解
