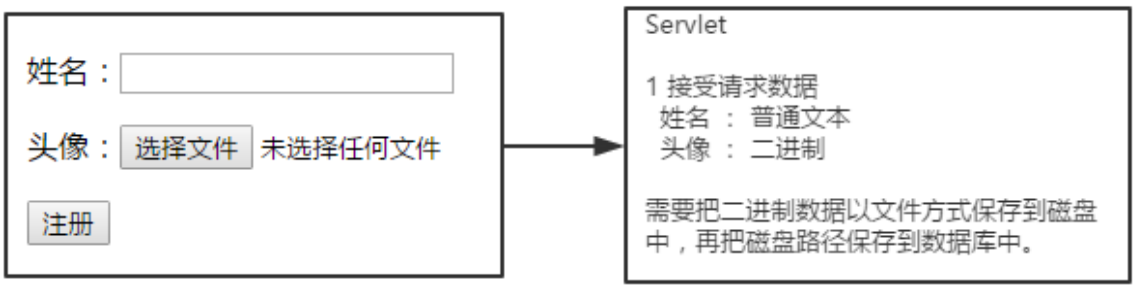
文件上传下载和三层架构

课程目标

- 掌握文件上传的基本操作，文件名和路径的设置，文件类型的约束。
- 掌握文件下载的基本操作，了解文件下载的文件名设置问题。
- 掌握登录功能的业务逻辑实现。
- 理解三层架构思想，掌握三层架构思想的应用。

一、文件上传

文件上传: 将用户本地磁盘中的文件提交保存到服务器中的磁盘上。



1、项目准备

搭建一个新 Web 项目，名为 fileupload-download，并添加 jstl jar。

2、编写 register.jsp

编写一个上传的表单，但要求如下：

- method 需要使用 post 提交，get 限制了数据大小。
- enctype 需使用 multipart/form-data，不然直接报错（需要二进制数据）。
- 需要提供 file 控件。

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>注册</title>
</head>
<body>
    <form action="/fileUpload" method="post" enctype="multipart/form-data">
        <p><input type="text" name="username"/></p>
        <p><input type="file" name="headImg"/></p>
        <input type="submit" value="注册">
    </form>
</body>
</html>
```

3、编写 FileUploadServlet.java

在此 Servlet 的 service 中查看上传文件请求的表单数据。

```
package cn.wolfcode.web._01_upload;

@WebServlet("/fileUpload")
public class FileUploadServlet extends HttpServlet {
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        // 二进制提交的数据，getParameter 方法无法获取到
        System.out.println(req.getParameter("username"));
        // 以流方式获取
        Scanner sc = new Scanner(req.getInputStream());
        while (sc.hasNextLine()){
            System.out.println(sc.nextLine());
        }
    }
}
```

注意：`enctype="multipart/form-data"` 提交的数据，getParameter() 无法获取到。

4、查看请求数据及问题

浏览器查看提交的数据格式：

如何解析出上传文件的内容？可以自己解析流来获取上传文件的内容，但这样太麻烦了，而文件上传又是项目中基本都需要的功能，所以会有人帮咱们解决。

二、Servlet 3.0 文件上传

此前，对于处理上传文件的操作一直是开发者比较头疼的问题，因为 Servlet 本身没有提供直接的支持，需要使用第三方框架来实现（Apache 的 FileUpload 组件），第三方框架使用起来也不简单。如今这些都成为了历史，Servlet 3.0 提供了文件上传操作功能，而且使用也非常简单。

1、API

HttpServletRequest 提供了两个方法用于从请求中解析上传的文件

返回值	方法	作用
Part	getPart(String name)	用于获取请求中指定 name 的文件
Collection	getParts()	获取请求中全部的文件

Part 中常用方法：

返回值	方法	作用
void	**write(String fileName)**	直接把接收到的文件保存到磁盘中
void	getContentType()	获取文件的类型 MIME
String	getHeader(String name)	获取请求头信息
long	getSize()	获取文件的大小

只需要给 Servlet 贴一个标签 `@MultipartConfig` 然后使用 `getPart()` 获取请求中指定 name 的文件到 Part 对象中,再使用 write 方法把文件保存到指定目录就 ok 了

2、代码示例

修改 FileUploadServlet 的 service 方法，使用 Servlet 3.0 上传 API 完成文件上传需求。只需要给 Servlet 贴一个标签 `@MultipartConfig` 然后使用请求对象的 `getPart()` 方法，指定上传控件的 name 的值，就可以获取到 Part 对象，再使用其的 write 方法把文件保存到指定目录就 ok 了。

```
package cn.wolfcode.web._01_upload;

@WebServlet("/fileUpload")
@MultipartConfig
public class FileUploadServlet extends HttpServlet {
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        // 普通控件数据还是使用 getParameter 方法来获取
        System.out.println("username:" + req.getParameter("username"));
        // 文件控件数据获取
        Part part = req.getPart("headImg");
        // 保存到磁盘上
        part.write("D:/headImg.png");
    }
}
```

三、文件上传细节

1、获取上传文件名

以前是拷贝文件（自传自存），知道文件类型；现在是用户传，程序接收，接收到文件时，涉及保存到磁盘使用什么文件名以及文件类型的问题，所有需要先获取到文件名及文件类型。可使用 Part API 获取：

返回值	方法	作用
String	getHeader("content-disposition")	Tocmat 8.0 之前使用通过请求头获取文件名，需截取字符串
String	getSubmittedFileName()	Tomcat8.0 之后提供的直接获取文件名方式

代码示例：

```
package cn.wolfcode.web._01_upload;

@WebServlet("/fileUpload")
@MultipartConfig
public class FileUploadServlet extends HttpServlet {
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        // 文件控件数据获取
        Part part = req.getPart("headImg");
        // 保存到磁盘上
        part.write("D:/" + part.getSubmittedFileName());
    }
}
```

2、文件名相同覆盖现有文件

若上传得文件名相同会导致覆盖服务器之前已上传的的文件，咱们的解决方法就是自己给文件起一个唯一的名称，确保不被覆盖，这里我们使用的是 UUID。

代码示例：

```
package cn.wolfcode.web._01_upload;

@WebServlet("/fileUpload")
@MultipartConfig
public class FileUploadServlet extends HttpServlet {
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        // 文件控件数据获取
        Part part = req.getPart("headImg");
        // 获取上传文件名
        String realFileName = part.getSubmittedFileName();
        // 获取上传文件扩展名
        String ext = realFileName.substring(realFileName.lastIndexOf("."));
        // 生成唯一字符串拼接文件名
        String fileName = UUID.randomUUID().toString() + ext;
        // 保存到磁盘上
        part.write("D:/" + fileName);
    }
}
```

3、文件保存位置问题

文件在磁盘某个位置，不在项目下，无法使用 HTTP 协议访问，所以要把用户上传的文件存放到项目中才可通过 HTTP 协议来访问，且保存的位置路径不可以写绝对路径，那么怎么办？可以通过 ServletContext 对象的 getRealPath("项目中保存上传文件的文件夹的相对路径") 来获取其的绝对路径。

在项目的 web 目录下新建一个名为 upload 文件夹，修改 UploadServlet.java 的代码如下：

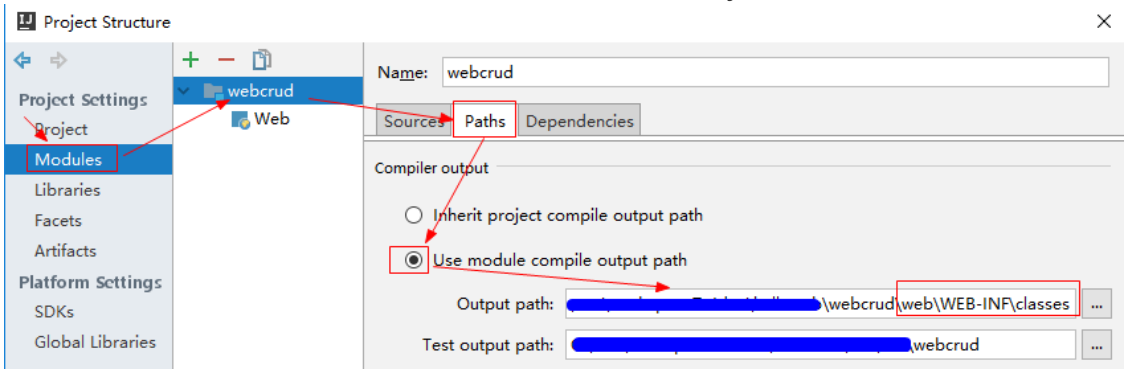
```
package cn.wolfcode.web._01_upload;

@WebServlet("/fileUpload")
@MultipartConfig
public class FileUploadServlet extends HttpServlet {
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        // 文件控件数据获取
        Part part = req.getPart("headImg");
        // 获取上传文件名
        String realFileName = part.getSubmittedFileName();
        // 获取上传文件扩展名
        String ext = realFileName.substring(realFileName.lastIndexOf("."));
        // 生成唯一字符串拼接文件名
        String fileName = UUID.randomUUID().toString() + ext;
        // 获取项目下的 upload 目录的绝对路径，拼接成文件的保存路径
        String realPath = getServletContext().getRealPath("/upload") +"/"+
fileName;
        // 保存到磁盘上
        part.write(realPath);
    }
}
```

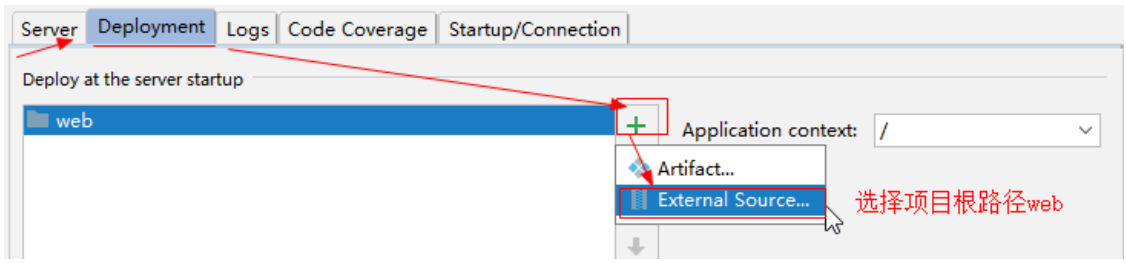
3.1、无法获取项目下的 upload 目录

以上方式没什么效果，原因是 IDEA 工具使用 **打包 web 项目（war）** 的方式来部署，所以位置有偏差，需要还原 Web 项目的原本目录结构，以及调整部署方式。调整步骤如下：

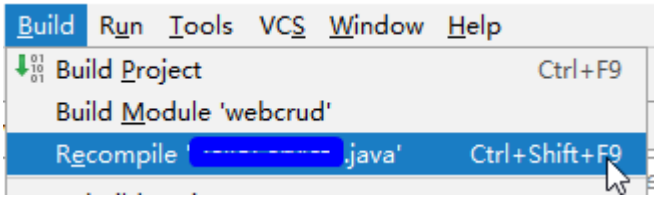
- WEB-INF 下 创建 classes 目录，用于存放 class 文件
- 修改项目 class 文件输出位置到 /WEB-INF/class 中（File -> Project Structure）



- 调整项目部署方式为 External Source



以上操作之后可以获取到项目下的 upload 目录了，但是之前的重新部署无效了，往后可使用编译类的方式来代替重新部署。



4、文件类型约束问题

限制用户恶意上传文件，比如要让用户上传头像，而用户却上传一个非图片文件，比如 JSP 文件。

4.1、修改 UploadServlet.java

加入判断上传文件类型的代码。

```
package cn.wolfcode.web._01_upload;

@WebServlet("/fileUpload")
@MultipartConfig
public class FileUploadServlet extends HttpServlet {
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        // 文件控件数据获取
        Part part = req.getPart("headImg");
        // 判断上传的文件类型合法不
        if(!part.getContentType().startsWith("img/")) {
            req.setAttribute("errorMsg", "请上传图片");
            req.getRequestDispatcher("/register.jsp").forward(req, resp);
            return;
        }

        String realFileName = part.getSubmittedFileName();
        // 获取文件扩展名
        String ext = realFileName.substring(realFileName.lastIndexOf("."));
        // 生成唯一字符串拼接文件名
        String fileName = UUID.randomUUID().toString() + ext;
        // 获取项目下的 upload 目录的绝对路径，拼接成文件的保存路径
        String realPath = getServletContext().getRealPath("/upload") +"/"+
fileName;
        // 保存到磁盘上
        part.write(realPath);
    }
}
```

4.2、修改 register.jsp

显示上传文件类型错误的提示给用户。

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>注册</title>
</head>
<body>
```

```
    <span style="color: red">${errorMsg}</span><br/>
    <form action="/fileUpload" method="post" enctype="multipart/form-data">
        <p><input type="text" name="username"/></p>
        <p><input type="file" name="headImg"/></p>
        <input type="submit" value="注册">
    </form>
</body>
</html>
```

5、文件大小约束问题

文件上传限制大小可提高服务器硬盘的使用率，防止用户恶意上传文件造成服务器磁盘资源紧张。可以通过设置 @MutipartConfig 的属性做限制，其属性如下：

- maxFileSize：单个上传文件大小限制，单位：bytes。
- maxRequestSize：显示请求中数据的大小，单位：bytes。

5.1、修改 register.jsp

再提供一个上传文件的 input 元素。

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>注册</title>
</head>
<body>
    <span style="color: red">${errorMsg}</span><br/>
    <form action="/fileUpload" method="post" enctype="multipart/form-data">
        <p><input type="text" name="username"/></p>
        <p><input type="file" name="headImg"/></p>
        <p><input type="file" name="headImg2"/></p>
        <input type="submit" value="注册">
    </form>
</body>
</html>
```

5.2、修改 FileUploadServlet.java

使用 @MutipartConfig 注解限制上传文件的大小。

```
package cn.wolfcode.web._01_upload;

@WebServlet("/fileUpload")
@MultipartConfig(maxFileSize = 80000, maxRequestSize = 140000)
public class FileUploadServlet extends HttpServlet {
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        req.setCharacterEncoding("UTF-8");

        try {
            Collection<Part> parts = req.getParts();
            for (Part part : parts) {
                String contentType = part.getContentType();
                if (contentType != null) { // input 类型是 file
```

```
                    // 判断上传的文件类型合法不
                    if(!contentType.startsWith("img/")){
                        req.setAttribute("errorMsg", "请上传图片");
                        req.getRequestDispatcher("/register.jsp").forward(req,
resp);
                        return;
                    }

                    // 获取上传文件名
                    String realFileName = part.getSubmittedFileName();
                    // 获取文件扩展名
                    String ext =
realFileName.substring(realFileName.lastIndexOf("."));
                    // 生成唯一字符串拼接文件名
                    String fileName = UUID.randomUUID().toString() + ext;
                    // 获取项目下的 upload 目录的绝对路径，拼接成文件的保存路径
                    String realPath = getServletContext().getRealPath("/upload")
+"/"+ fileName;
                    // 保存到磁盘上
                    part.write(realPath);
                } else { // input 类型非 file
                    System.out.println(req.getParameter(part.getName()));
                }
            }
        }catch (Exception e) {
            e.printStackTrace();
            req.setAttribute("errorMsg", "文件太大了");
            req.getRequestDispatcher("/register.jsp").forward(req, resp);
        }
    }
}
```

拓展：抽取文件上传工具方法。

四、文件下载

将服务器中的资源下载保存到用户电脑中。

1、文件下载简单实现

1.1、在项目提供下载资源

在 web 下新建 download 目录，里面提供两个资源 dog.rar 和 猫.rar。

1.2、编写 download.jsp

提供两个下载链接。

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>下载</title>
</head>
<body>
    <h3>文件下载</h3>
    <a href="/download/dog.rar">dog.rar</a><br/>
    <a href="/download/猫.rar">猫.rar</a><br/>
</body>
</html>
```

效果如下：

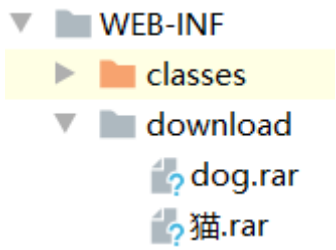
文件下载

dog.rar
猫.rar

2、文件下载限制

下载功能已经实现，但是文件放在 WEB-INF 外面不安全，用户只需要拿到下载的超链接都能够下载，实际开发中，我们的文件通常需要用户有一定的权限才能下载，所以文件应该放在 WEB-INF 下，这样的话，用户就不可以直接访问到了，须请求到交由 Servlet 来处理，我们就可以在其 service 方法中编写下载限制操作。

2.1、移动下载资源到 WEB-INF 下



2.2、修改 download.jsp

修改下载资源的链接地址。

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>下载</title>
</head>
<body>
    <h3>文件下载</h3>
    <a href="/download?fileName=dog.rar">dog.rar</a><br/>
    <a href="/download?fileName=猫.rar">猫.rar</a><br/>
</body>
</html>
```

2.3、编写 DownloadServlet.java

根据请求传递文件名参数获取对应文件响应给浏览器。

```
package cn.wolfcode.web._01_upload;

@WebServlet("/download")
public class DownloadServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        // 获取用户要下载的文件名称
        String fileName = req.getParameter("fileName");
        // 获取文件所在根路径
        String realPath = req.getServletContext().getRealPath("/WEB-
INF/download/");
        // 使用工具类 Files 的 copy 方法获取文件输入流，响应回浏览器
        Files.copy(Paths.get(realPath, fileName), resp.getOutputStream());
    }
}
```

3、下载文件名称问题

默认情况下，Tomcat 服务器未告知浏览器文件的名称，所以需要手动设置响应头来告知浏览器文件名称，方法如下：

```
// 无需记，知道需要设置即可
// 给浏览器一个推荐名称
resp.setHeader("Content-Disposition", "attachment;filename=文件名称");
```

处理中文件名称的问题

- IE 使用 URL 编码方式：URLEncoder.encode(fileName, "UTF-8")
- 非 IE使用 ISO-8859-1 编码：new String (fileName.getBytes("UTF-8"), "ISO-8859-1")

修改 DownloadServlet.java 代码如下：

```
package cn.wolfcode.web._01_upload;

@WebServlet("/download")
public class DownloadServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        // 获取用户要下载的文件名称
        String fileName = req.getParameter("fileName");
        // 获取浏览器类型
        String header = req.getHeader("User-Agent");
        // 根据浏览器类型设置文件下载的名称
        String name = header.contains("MSIE") ?  URLEncoder.encode(fileName,
"UTF-8") : new String(fileName.getBytes("UTF-8"), "ISO-8859-1");
        // 设置下载文件名
        resp.setHeader("Content-Disposition", "attachment;filename=" + name);
        // 获取文件所在根路径
        String realPath = req.getServletContext().getRealPath("/WEB-
INF/download/");
        // 使用工具类 Files 的 copy 方法获取文件输入流，响应回浏览器
        Files.copy(Paths.get(realPath, fileName), resp.getOutputStream());
```

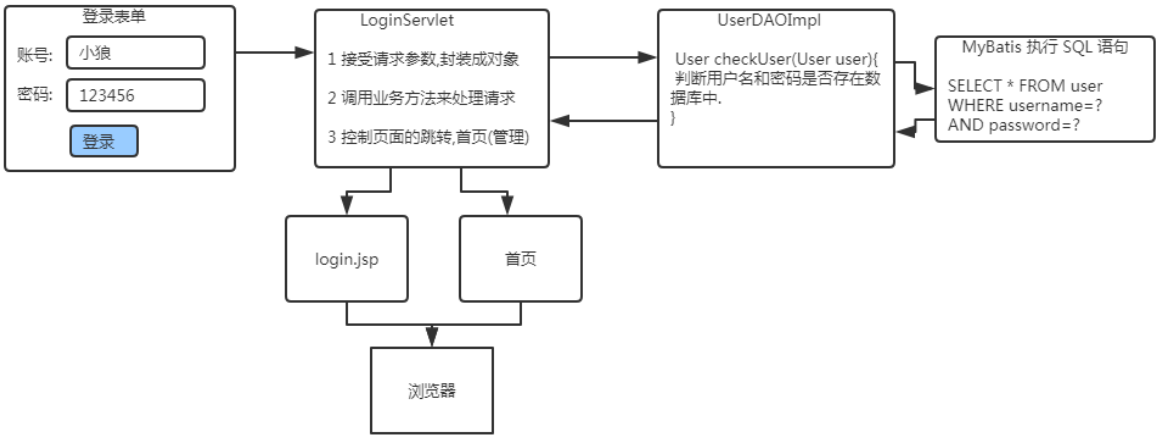
```
    }
}
```

五、登录实现

1、为什么做登录

服务器资源给合理的人访问，在这里只有登录的用户才可以访问，比如只有登录的用户就可以访问产品进行 CRUD。

2、分析登录流程



六、登录实现第一个版本

1、项目准备及新建 user 表

打开之前 Web CRUD 项目，在其对应的库中新建 user 表，并造一条数据，表结构如下：

```
CREATE TABLE `user` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `username` varchar(255) DEFAULT NULL,
  `password` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO user(username, password) VALUES('xiaolang', '123456');
```

2、编写 User.java

```java
package cn.wolfcode.domain;

@Setter
@Getter
public class User {
    private Long id;
    private String username;
    private String password;
}
```

3、编写 UserMapper.xml

在 mapper 包下新建 UserMapper.xml，文件内容如下：

```xml
<select id="checkUser" resultType="cn.wolfcode.domain.User">
    SELECT * FROM user WHERE username = #{username} AND password = #{password}
</select>
```

记得在 mybatis-config.xml 关联。

4、编写 IUserDAO.java 和 UserDAOImpl.java

里面提供根据用户名和密码查询用户的方法。

```java
package cn.wolfcode.dao;

public interface IUserDAO {
    User checkUser(User user);
}
```

```java
package cn.wolfcode.dao.impl;

public class UserDAOImpl implements IUserDAO {
    @Override
    public User checkUser(User user) {
        SqlSession session = MyBatisUtil.getSession();
        User u = session.selectOne("cn.wolfcode.mapper.UserMapper.checkUser",
user);
        session.close();
        return u;
    }
}
```

5、编写 LoginServlet.java

处理登录请求，对登录成功或失败进行响应。

```java
package cn.wolfcode.web.servlet;

@WebServlet("/login")
public class LoginServlet extends HttpServlet {

    private IUserDAO userDAO = new UserDAOImpl();
```

```
    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        req.setCharacterEncoding("UTF-8");
        // 接受请求参数，封装成对象
        User user = new User();
        user.setUsername(req.getParameter("username"));
        user.setPassword(req.getParameter("password"));
        // 调用业务方法来处理登录请求
        User u = userDAO.checkUser(user);
        if(u == null){
            // 通知用户账号或密码错误
            req.setAttribute("errorMsg", "账号或密码错误");
            req.getRequestDispatcher("/login.jsp").forward(req, resp);
            return;
        }
        // 登录成功，重定向产品列表页面
        resp.sendRedirect("/product");
    }
}
```

6、编写 login.jsp

在 web 目录新建 login.jsp，文件内容如下：

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>登录</title>
</head>
<body>
    <h3>用户登录</h3>
    ${errorMsg}
    <form action="/login" method="post">
        <p>账号: <input type="text" name="username"></p>
        <p>密码: <input type="text" name="password"></p>
        <input type="submit" value="登录">
    </form>
</body>
</html>
```

7、访问限制

因为上面虽然把登录需求做完，但大家发现，若在浏览器地址中直接输入 localhost/product 访问依然是可以的。那么怎么解决这个问题呢?

7.1、修改 LoginServlet.java

当登录成功时往 Session 中存入一个标识，比如 User 对象，给其它需要登录才可以操作的资源做判断使用。

```
package cn.wolfcode.web.servlet;

@WebServlet("/login")
```

```
public class LoginServlet extends HttpServlet {

    private IUserDAO userDAO = new UserDAOImpl();

    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        req.setCharacterEncoding("UTF-8");
        // 接受请求参数，封装成对象
        User user = new User();
        user.setUsername(req.getParameter("username"));
        user.setPassword(req.getParameter("password"));
        // 调用业务方法来处理登录请求
        User u = userDAO.checkUser(user);
        if(u == null){
            // 通知用户账号或密码错误
            req.setAttribute("errorMsg", "账号或密码错误");
            req.getRequestDispatcher("/login.jsp").forward(req, resp);
            return;
        }

        // 登录成功时往 Session 中加入一个登录成功的标识
        req.getSession().setAttribute("USER_IN_SESSION", u);

        // 重定向产品列表页面
        resp.sendRedirect("/product");
    }
}
```

7.2、修改 ProductServlet 的 service 方法

对产品 CRUD 的资源或者操作是需要登录才可以访问或操作，那么从 Session 中来获取登录的标识，获取到则有登录可以访问或操作；获取不到则没登录，没登录则跳转去登录页面。

```
protected void service(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
    req.setCharacterEncoding("UTF-8");

    // 判断 Session 中是否有登录的标示，有则说明登录成功了，没有则说明没有登录
    Object user = req.getSession().getAttribute("USER_IN_SESSION");
    if(user == null){
        // 没有登录滚去登录
        resp.sendRedirect("/login.jsp");
        return;
    }

    String cmd = req.getParameter("cmd");
    // 针对参数 cmd 值不同，分发到不同的方法去处理
    if("delete".equals(cmd)){
        delete(req, resp);
    }else if("input".equals(cmd)){
        input(req, resp);
    }else if("saveOrUpdate".equals(cmd)){
        saveOrUpdate(req, resp);
    }else{
        list(req, resp);
    }
```

```
}
```

七、登录实现第二个版本

若现在新增一个需求，登录时若用户名写错了，提示用户其输入的用户名错了；若是密码写错了，提示用户其收入的密码错了。

1、修改 UserMapper.xml

```
<select id="checkByUsername" resultType="cn.wolfcode.domain.User">
    SELECT * FROM user WHERE username = #{username}
</select>
```

2、修改 IUserDAO.java 和 UserDAOImpl.java

```
package cn.wolfcode.dao;

public interface IUserDAO {
    User checkUser(String username);
}
```

```
package cn.wolfcode.dao.impl;

public class UserDAOImpl implements IUserDAO {
    @Override
    public User checkUser(String username) {
        SqlSession session = MyBatisUtil.getSession();
        User u =
session.selectOne("cn.wolfcode.mapper.UserMapper.checkByUsername", username);
        session.close();
        return u;
    }
}
```

3、修改 LoginServlet.java

登录时若用户名写错了，提示用户其输入的用户名错了；若是密码写错了，提示用户其收入的密码错了。

```
package cn.wolfcode.web.servlet;

@WebServlet("/login")
public class LoginServlet extends HttpServlet {

    private IUserDAO userDAO = new UserDAOImpl();

    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        req.setCharacterEncoding("UTF-8");
        // 接受请求参数
```

```
        User user = new User();
        String username = req.getParameter("username");
        String password = req.getParameter("password");

        // 调用业务方法来处理登录请求
        User u = userDAO.checkUser(username);
        // 判断用户名是否正确
        if(u == null) {
            req.setAttribute("errorMsg", "账号不存在");
            req.getRequestDispatcher("/login.jsp").forward(req, resp);
            return;
        }
        // 判断密码是否正确
        if(!u.getPassword().equals(password)) {
            req.setAttribute("errorMsg", "密码错误");
            req.getRequestDispatcher("/login.jsp").forward(req, resp);
            return;
        }

        // 登录成功时往 Session 中加入一个登录成功的标识，给其它需要登录才可以操作的资源做判断使用
        req.getSession().setAttribute("USER_IN_SESSION", u);

        // 重定向产品列表页面
        resp.sendRedirect("/product");
    }
}
```

4、存在的问题

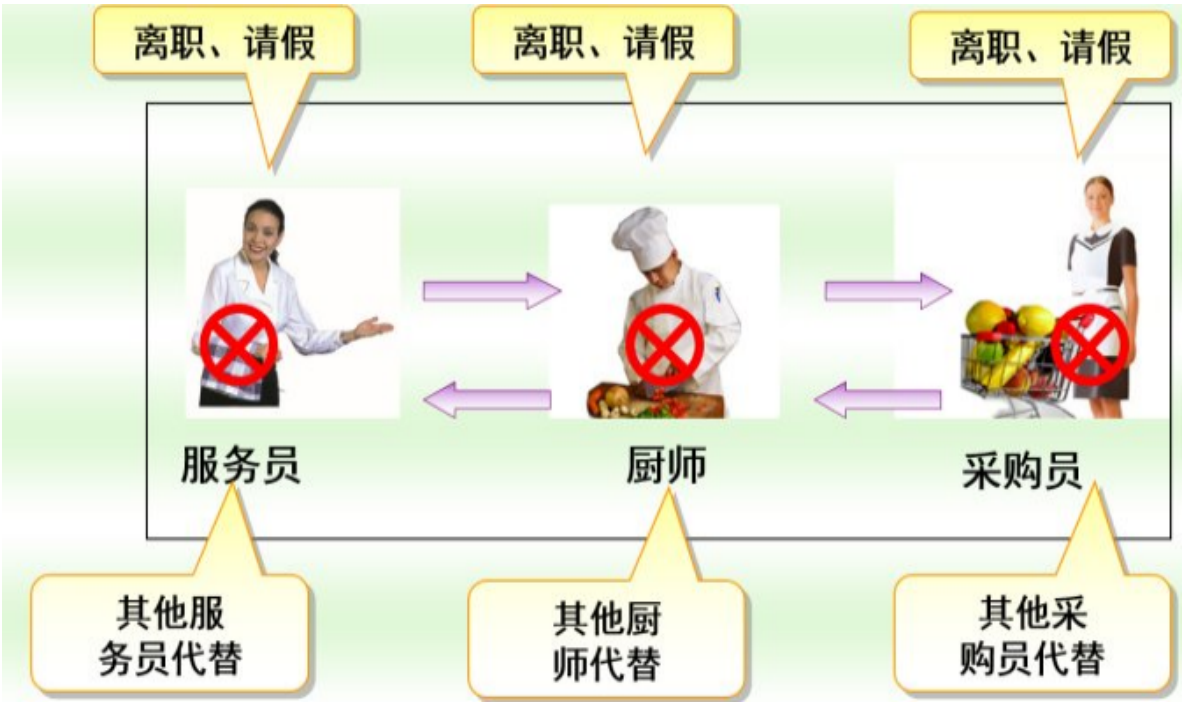
目前在 Servlet 中书写了逻辑的操作代码，这不属于 Servlet 的责职的，怎么办？解决方案：抽取 Servlet 的逻辑代码。

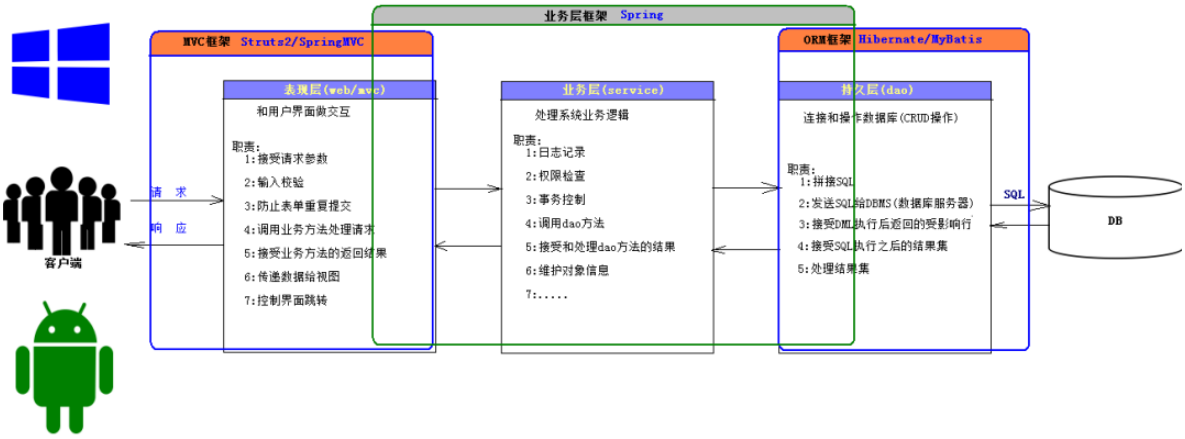
八、三层架构

1、三层架构介绍

Web 开发中的最佳实践：分层开发模式（技术层面的"分而治之"）。三层架构(3-tier architecture)：通常意义上的三层架构就是将整个业务应用划分为：**表现层、业务逻辑层、数据访问层。**区分层次的目的即为了“高内聚低耦合”的思想。在软件体系架构设计中，分层式结构是最常见，也是最重要的一种结构。

- **表现层**（Predentation Layer）：MVC，负责处理与界面交互的相关操作。
- **业务层**（Business Layer）：Service，负责复杂的业务逻辑计算和判断。
- **持久层**（Persistent Layer）：DAO，负责将业务逻辑数据进行持久化存储。





分层好处：责任分离，各司其职，体现高内聚低耦合。

2、业务层命名规范

- 包名：
  - 公司域名倒写.service：存放业务接口代码。
  - 公司域名倒写.service.impl：存放业务层接口的实现类。
- 类名：

- IXxxService：业务层接口，Xxx 表示对应模型，比如操作 User 的就起名为 IUserService。
- XxxServiceImpl：业务层接口对应的实现类，比如操作 User 的就起名为 UserServiceImpl。
- XxxServiceTest：业务层实现的测试类。

九、登录加入业务层

1、编写 IUserService.java 和 UserServiceImpl.java

调用 DAO 对象的查询方法，若登录失败，通过抛出异常的方式，返回错误信息。

```
package cn.wolfcode.service;

public interface IUserService {
    User login(String username, String password);
}
``

```java
package cn.wolfcode.service.impl;

public class UserServiceImpl implements IUserService {

    private IUserDAO userDAO = new UserDAOImpl();

    @Override
    public User login(String username, String password) {
        User user = userDAO.checkUser(username);
        // 用户名错误
        if(user == null) {
            throw new RuntimeException("账号不存在");
        }
        // 密码错误
        if(!user.getPassword().equals(password)) {
            throw new RuntimeException("密码错误");
        }
        return user;
    }
}
```

2、修改 LoginServlet.java

调用业务对象的业务方法。

```
package cn.wolfcode.web.servlet;

@WebServlet("/login")
public class LoginServlet extends HttpServlet {

    private IUserService userService = new UserServiceImpl();

    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        req.setCharacterEncoding("UTF-8");
```
```

```
        // 接受请求参数
        String username = req.getParameter("username");
        String password = req.getParameter("password");
        try {
            // 调用业务方法来处理登录请求
            User user = userService.login(username, password);
            // 登录成功时往 Session 中加入一个登录成功的标识，给其它需要登录才可以操作的资源做判断使用
            req.getSession().setAttribute("USER_IN_SESSION", user);
            // 控制跳转
            resp.sendRedirect("/product");
        }catch (Exception e){ // 有异常时表示登录失败，
            e.printStackTrace();
            req.setAttribute("errorMsg", e.getMessage());
            req.getRequestDispatcher("/login.jsp").forward(req, resp);
        }
    }
}
```

十、加入用户头像

1、显示用户头像

1.1、新建存放头像文件夹

在项目中的 web 目录新建一个 headImg 文件夹，把一些头像图片放进去。

1.2、修改 user 表

给 user 表新增一个 headImg 字段，类型 varchar，用于记录用户头像图片路径。给现有用户数据的 headImg 字段设置好现有图片的路径。

1.3、修改 User.java

```
package cn.wolfcode.domain;

@Setter
@Getter
public class User {
    private Long id;
    private String username;
    private String password;
    private String headImg; // 增加头像字段
}
```

1.4、修 list.jsp

在产品的 list.jsp 的添加 a 标签之前添加如下代码：

```
<img src="${USER_IN_SESSION.headImg}"/><br/>
```

2、修改用户头像

2.1、修改 list.jsp

点击用户头像跳转修改头像的页面。

```
<a href="/replaceImg.jsp"><img src="${USER_IN_SESSION.headImg}" title="更换头像"/></a><br/>
```

2.2、编写 replaceImg.jsp

在项目的 web 目录新建 replaceImg.jsp，编写一个上传头像的表单。

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>头像修改</title>
</head>
<body>
    <form action="/replaceImg" method="post" enctype="multipart/form-data">
        更换头像：<input type="file" name="headImg">
        <input type="submit" value="提交">
    </form>
</body>
</html>
```

2.3、编写 HeadImgServlet.java

接受用户修改头像的请求并调用业务方法修改用户头像。

```
package cn.wolfcode.web.servlet;

@WebServlet("/replaceImg")
@MultipartConfig
public class HeadImgServlet extends HttpServlet {

    private IUserService userService = new UserServiceImpl();

    protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        // 判断是否有登录，有则可以修改头像
        Object obj = req.getSession().getAttribute("USER_IN_SESSION");
        if(obj == null){
            resp.sendRedirect("/login.jsp");
            return;
        }
        // 有登录，获取新上传的头像图片
        Part part = req.getPart("headImg");
        // 获取上传文件名称
        String fileName = part.getSubmittedFileName();
        // 获取上传文件后缀名
        String ext = fileName.substring(fileName.lastIndexOf("."));
        // 创建的文件名，需要存到数据库中的
        String newFileName =  UUID.randomUUID().toString() + ext;
        // 获取图片存放的根路径
        String baseDir = req.getServletContext().getRealPath("/headImg");
        // 上传的头像保存倒磁盘中
```

```
            part.write(baseDir + "/" + newFileName);

            User user = (User) obj;
            user.setHeadImg("/headImg/" + newFileName);
            // 调用业务方法来修改用户头像
            userService.updateHeadImg(user);
            // 控制跳转
            resp.sendRedirect("/product");
        }
    }
```

2.4、修改 IUserService.java 和 UserServiceImpl.java

提供修改用户头像的业务方法。

```
void updateHeadImg(User user);
```

```
@Override
public void updateHeadImg(User user) {
    userDAO.updateHeadImg(user);
}
```

2.5、修改 IUserDAO.java 和 UserDAOImpl.java

提供修改用户头像的方法。

```
void updateHeadImg(User user);
```

```
@Override
public void updateHeadImg(User user) {
    SqlSession session = MyBatisUtil.getSession();
    session.update("cn.wolfcode.mapper.UserMapper.updateHeadImg",user);
    session.commit();
    session.close();
}
```

2.6、修改 UserMapper.xml

```
<update id="updateHeadImg">
    UPDATE user SET headImg = #{headImg} WHERE id = #{id}
</update>
```

练习

- 在之前 Web CRUD 项目基础上基于三层架构的完成登录和登录拦截功能（只有登录才操作产品资源）。
- 在之前 Web CRUD 项目基础上提供文件上传功能。
- 在之前 Web CRUD 项目基础上提供文件下载功能。