J2EE

J2EE: 分布式开发平台的标准

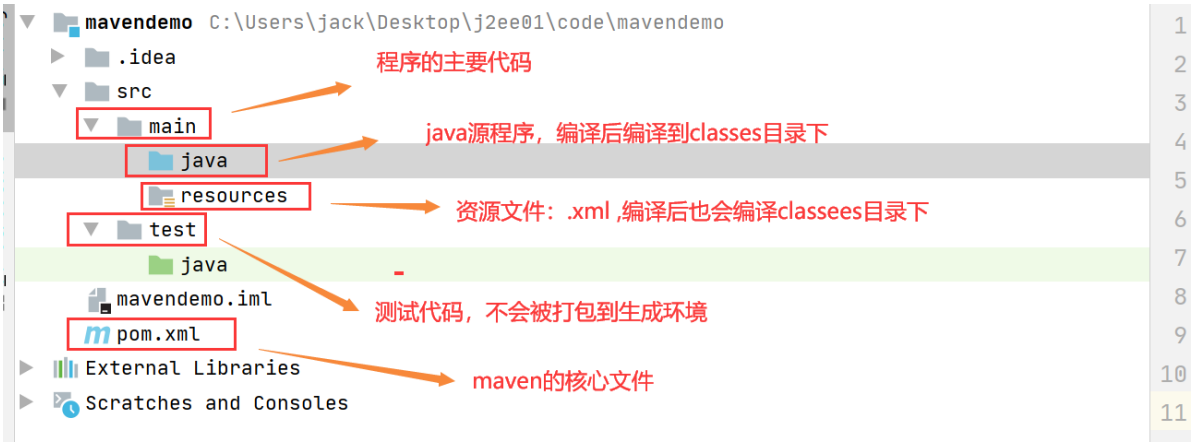
servlet：基于 J2EE标准的API

目标：

1. 使用maven部署servlet环境
2. 使用 servlet 实现简单的CURD

maven

工程结构



pom.xml: 配置简介

```
<properties>
    <!--编码-->
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <!--编译的源版本：环境版本-->
    <maven.compiler.source>1.8</maven.compiler.source>
    <!--编译的目标版本-->
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>

<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.11</version>
        <!--只可以在 test 目录下引用-->
        <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
```

```
        <version>8.0.17</version>
    </dependency>

</dependencies>

<!--工具配置-->
<build>
    <plugins>
        <!--配置编译工具-->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>
        <!--配置打包工具-->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-jar-plugin</artifactId>
            <configuration>
                <archive>
                    <manifest>
                        <addClasspath>true</addClasspath>
                        <!--   MANIFEST.MF 中 Class-Path -->
                        <classpathPrefix>lib/</classpathPrefix>
                        <mainClass>com.iweb.test.User</mainClass>
                    </manifest>
                </archive>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Servlet入门程序（环境部署）

1. 构建maven工程选择： `maven-archetype-webapp`;
2. 配置pom.xml

   修改JDK版本：1.8

   ```
   <properties>
       <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
       <maven.compiler.source>1.8</maven.compiler.source>
       <maven.compiler.target>1.8</maven.compiler.target>
   </properties>
   ```

   引入servlet的依赖

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
</dependency>
```

3. 编写服务端代码： com.iweb.servlet.UserServlet 继承 HttpServlet，并重写service方法

```
@Override
protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
    // 业务方法
    System.out.println("service");
    // 获取用户请求的 url
    String url = req.getRequestURI();
    System.out.println("url:" + url);

    // 获取用户请求的数据 : http://localhost:8080/login?
username=jack&password=111
    String username = req.getParameter("username");
    String password = req.getParameter("password");
    System.out.println(username + "\t" + password);


    // 转发视图
    req.getRequestDispatcher("/main.html").forward(req, resp);
}
```

4. 配置web.xml

```
<servlet>
    <servlet-name>user</servlet-name>
    <!--com.iweb.servlet.UserServlet和上面的业务类对应-->
    <servlet-class>com.iweb.servlet.UserServlet</servlet-class>
</servlet>
<!--servlet映射-->
<servlet-mapping>
    <servlet-name>user</servlet-name>
    <url-pattern>/login</url-pattern>
</servlet-mapping>
```

5. 编写请求页面index.html和返回页面main.html

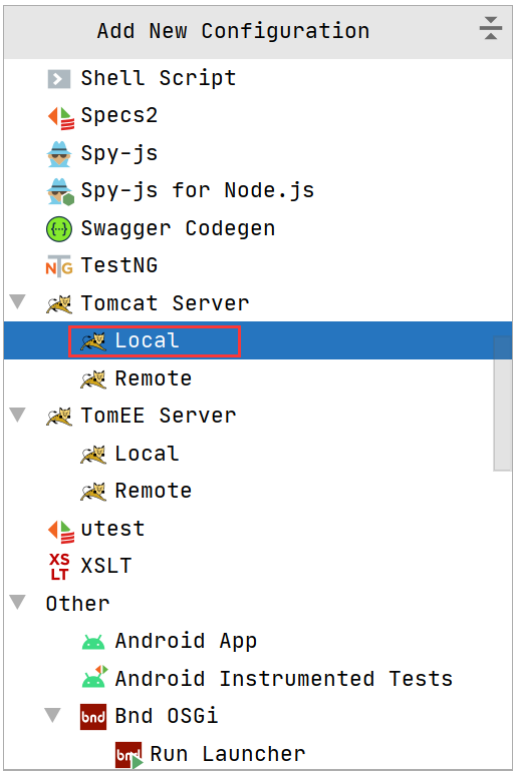
index.html

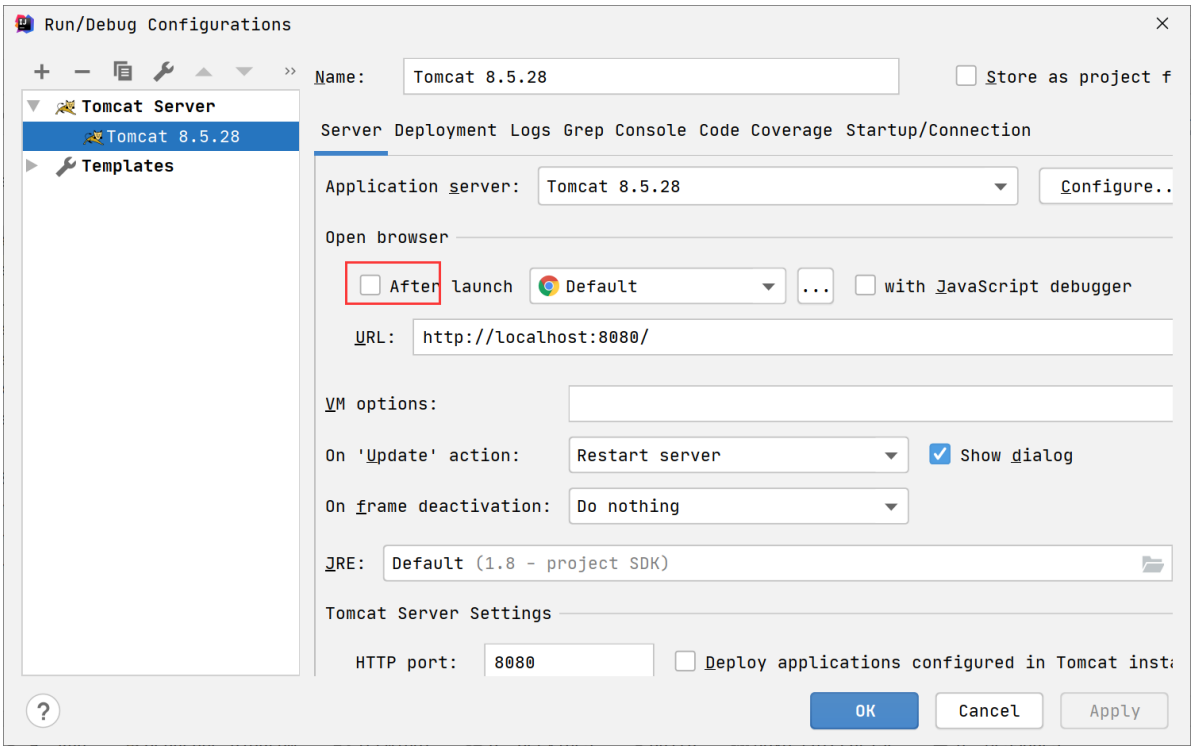
```
<form action="/login" method="get">
    <label for="username">username:</label>
    <input type="text" name="username" id="username">
    <label for="password">username:</label>
    <input type="text" name="password" id="password">
    <input type="submit" value="登录">
</form>
```

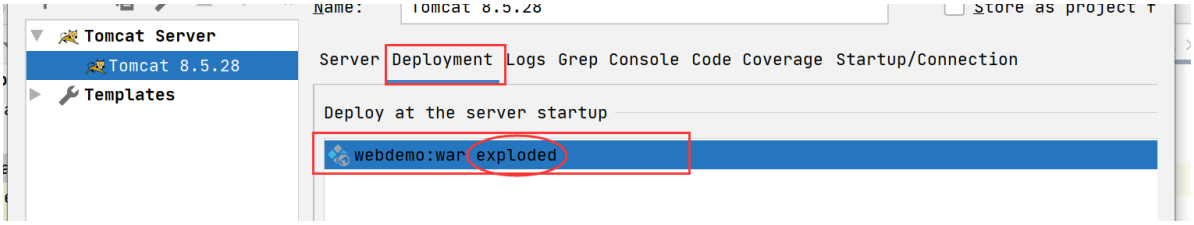
main.html

```
<h1>main</h1>
```

6. 通过idea部署







修改 应用的根路径为 /

请求访问的原理：

1. 用户从页面发起一个 http request请求
2. 通过请求url找到对应的 servlet 业务类
3. 执行业务类中的 service 方法