

Servlet简介和环境设置

[Servlet 简介](#)

[Servlet 环境设置](#)

[导入jar包](#)

[web.xml文件配置](#)

[@WebServlet注解配置](#)

[web.xml文件的方式和@WebServlet区别](#)

Servlet 简介

Servlet 是运行在 Web 服务器或应用服务器上的程序，它是作为来自 Web 浏览器或其他 HTTP 客户端的请求和 HTTP 服务器上的数据库或应用程序之间的中间层。

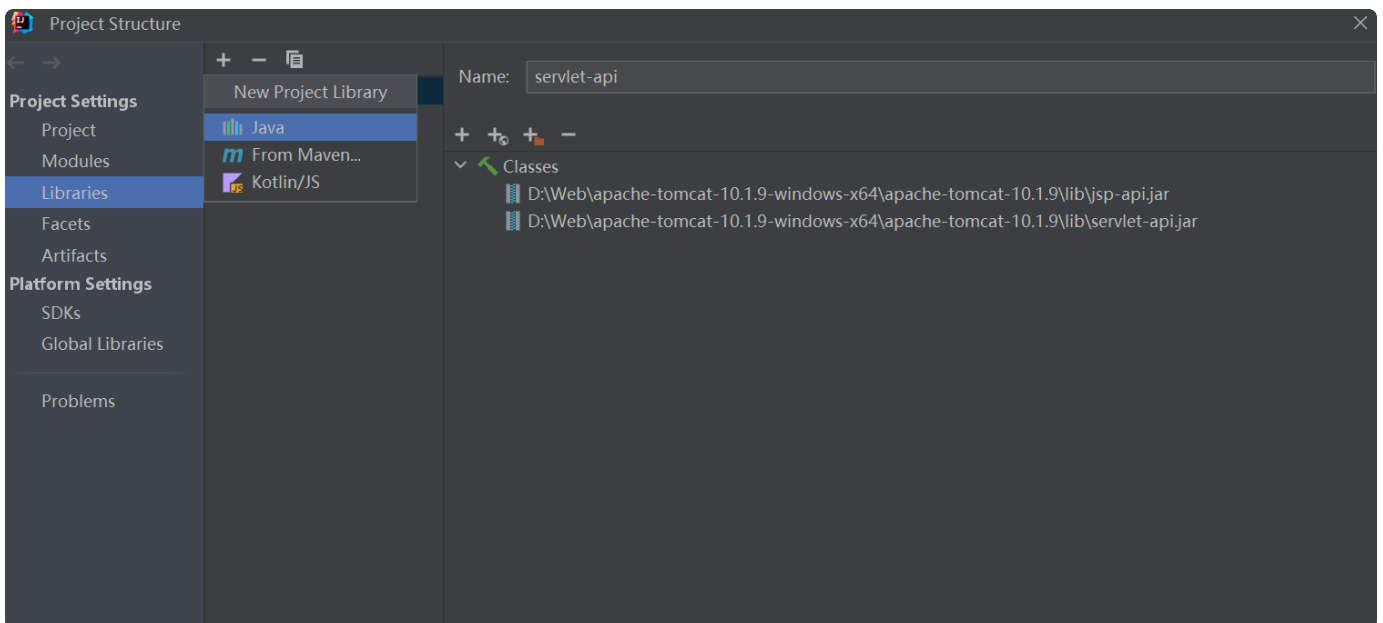
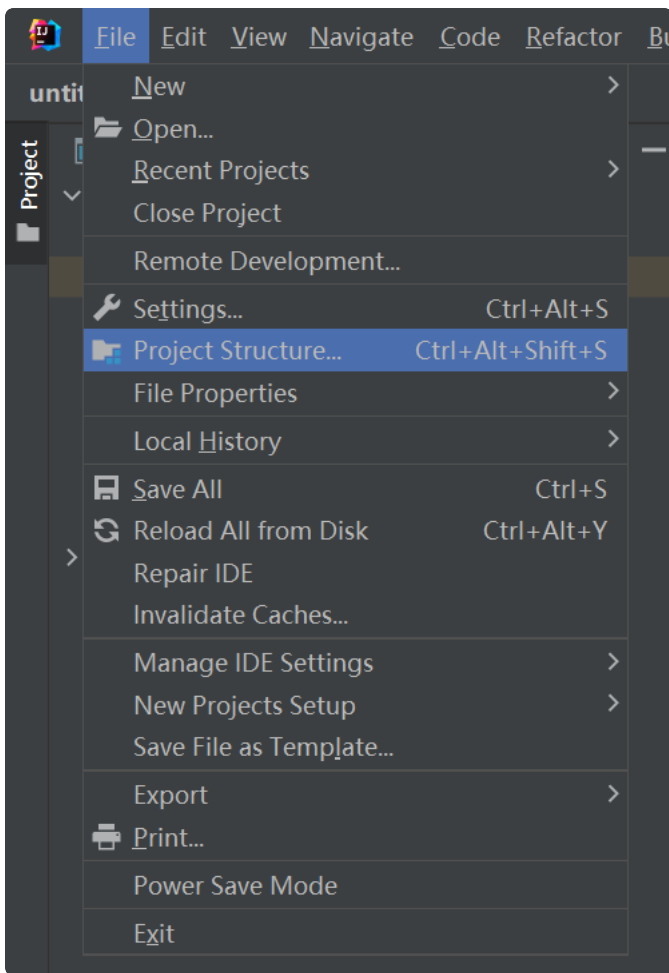
使用 Servlet，您可以收集来自网页表单的用户输入，呈现来自数据库或者其他源的记录，还可以动态创建网页。

总而言之，Servlet 是用于处理 Web 请求和响应的标准 Java 技术，是 Web 应用程序开发不可或缺的组成部分。

Servlet 环境设置

导入jar包

File-> Project Structure-> libraries 选择“+”，选择java-> 找到tomcat路径下的lib文件夹里选择servlet-api与 jsp-api；

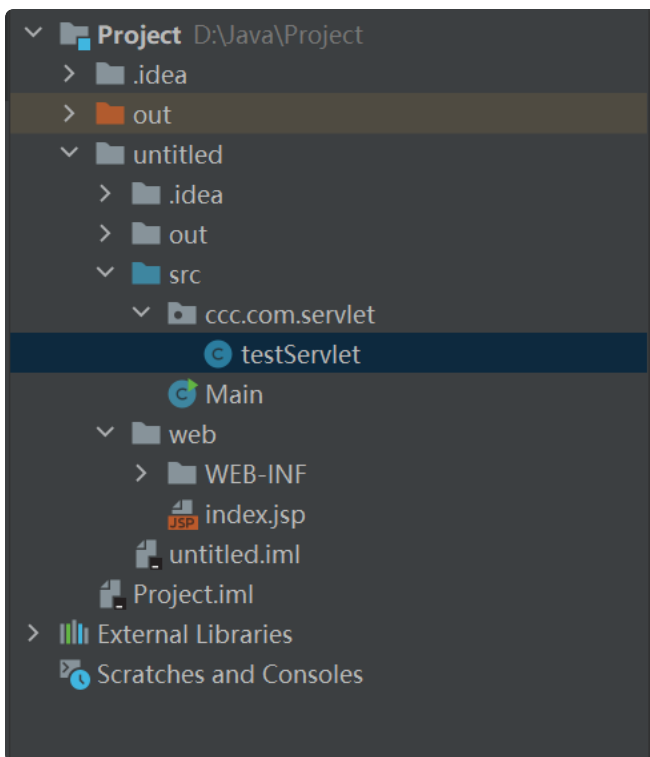


web.xml文件配置

在 WEB-INF 目录下创建 web.xml 文件；

```
1
2 <servlet>
3     <!-- Servlet别名 自定义-->
4     <servlet-name>TestServlet</servlet-name>
5     <!--servlet的类全名-->
6     <servlet-class>ccc.com.servlet.testServlet</servlet-class>
7 </servlet>
8
9 <servlet-mapping>
10    <!-- 将Servlet和URL绑定 -->
11    <!--给Servlet提供（映射）一个可供客户端访问的URI-->
12    <servlet-name>TestServlet</servlet-name>
13    <!--必须和servlet中的name相同-->
14    <url-pattern>/test</url-pattern>
15    <!-- servlet的映射路径（访问servlet的名称） -->
16 </servlet-mapping>
17
18
```

在src文件下新建包 `ccc.com.servlet`，在里面新建一个类 `testServlet`；



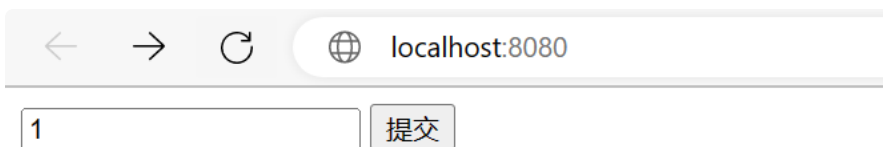
`testServlet`类里继承`HttpServlet`，代码如下：

```
1 public class testServlet extends HttpServlet {  
2  
3     protected void doGet(HttpServletRequest request, HttpServletResponse response) {  
4         System.out.println(request.getParameter("id"));  
5     }  
6 }
```

在index.jsp里面写入以下代码，注意name属性要对应：

```
1 <form action="/test" method="get">  
2     <input type="text" name="id">  
3     <input type="submit" value="提交">  
4 </form>
```

然后我们启动服务器进入浏览器页面

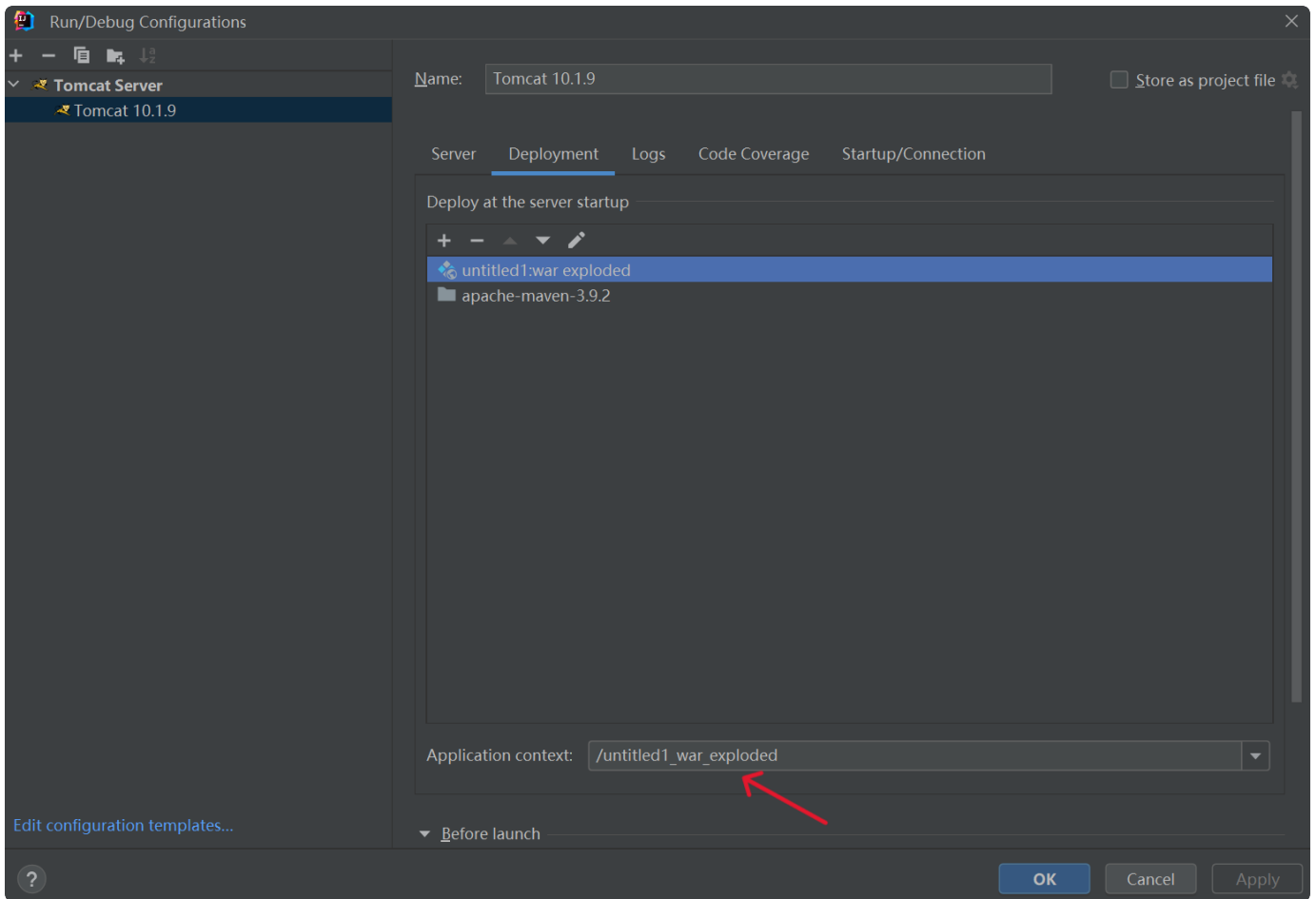


← → ↻ 🌐 localhost:8080

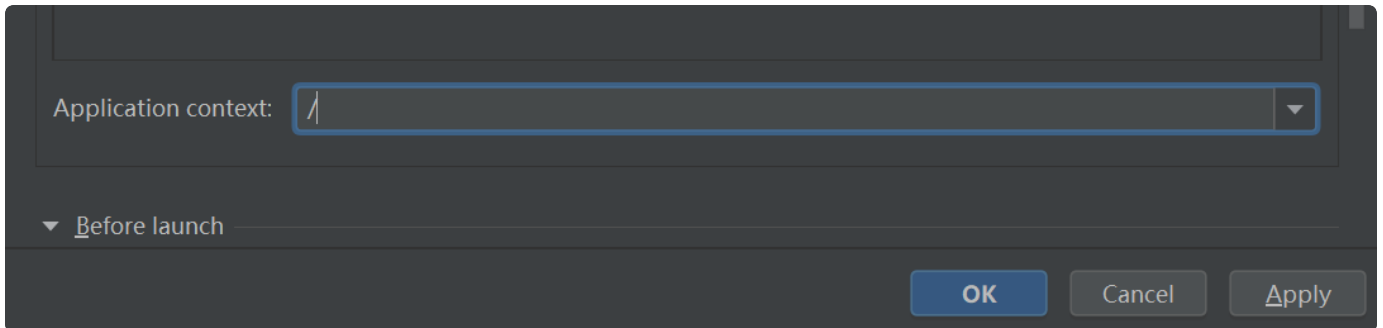
1



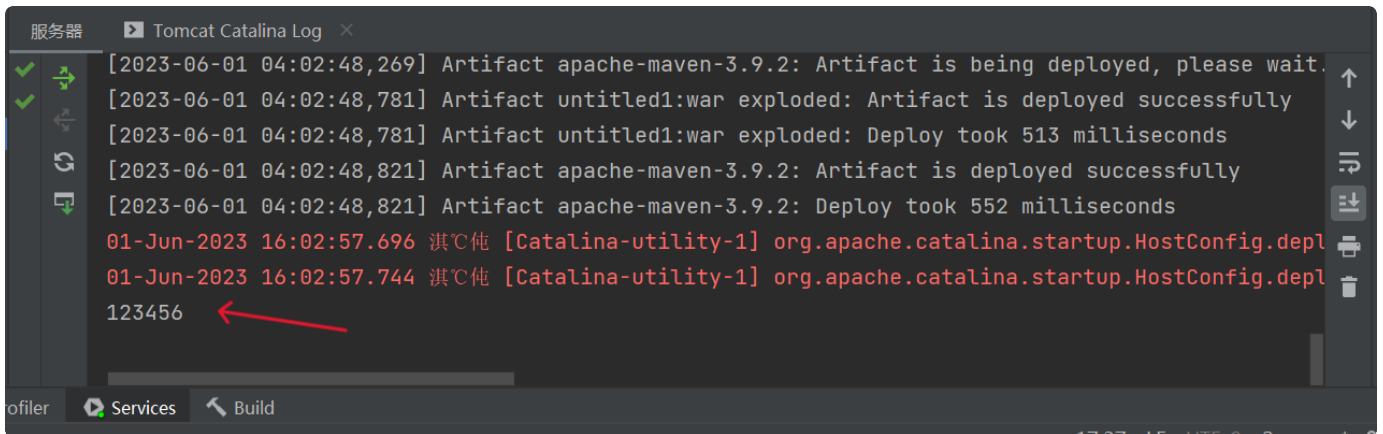
找到Run-> Edit Configurations ->右边找到Deployment



把他改成/



重新启动服务器然后输入框里随便输入字符串点击提交，就可以发现已经在控制台打印出来了；



要设置 servlet 返回数据，可以调用 HttpServletResponse 对象的方法，例如：

```
Java | 复制代码

1 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
2     // 设置响应内容类型
3     response.setContentType("text/html;charset=UTF-8");
4
5     // 获取输出流对象
6     PrintWriter out = response.getWriter();
7     try {
8         // 将响应信息输出到客户端浏览器
9         out.println("<html>");
10        out.println("<head>");
11        out.println("<title>成功</title>");
12        out.println("</head>");
13        out.println("<body>");
14        out.println("<h1>成功</h1>");
15        out.println("</body>");
16        out.println("</html>");
17    } finally {
18        // 关闭输出流对象
19        out.close();
20    }
21 }
```

结果如下：

```
← ↻ ⓘ localhost:8080/test?id=1
```

成功

@WebServlet注解配置

这种方式的话比较简单，只需要添加一个@WebServlet；

```
1  @WebServlet(name = "TestServlet",urlPatterns = "/test")
2
3  public class testServlet extends HttpServlet {
4
5      protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
6          System.out.println(request.getParameter("id"));
7      }
8  }
```

web.xml文件的方式和@WebServlet区别

相比于 web.xml 文件的方式，在使用 @WebServlet 注解时，不再需要手动编写 XML 文件来描述 Servlet 的配置信息，而是通过直接在 Servlet 对应的类上添加注解来完成。这样做的好处是：

1. 更加方便：不用像 web.xml 那样繁琐地编写 XML 文件；
2. 更加简洁：去除了 XML 中大量的模板式代码和冗余的信息；
3. 更加高效：在应用启动时只需扫描注解，而不必解析整个 XML 文件；
4. 更加灵活：注解更容易被理解和调整。

虽然使用 @WebServlet 注解能够带来很多好处，但也有局限性，不能用于配置其它组件，并且无法适应更复杂的场景。在实际开发中，要根据需要来选择合适的方式来进行配置。