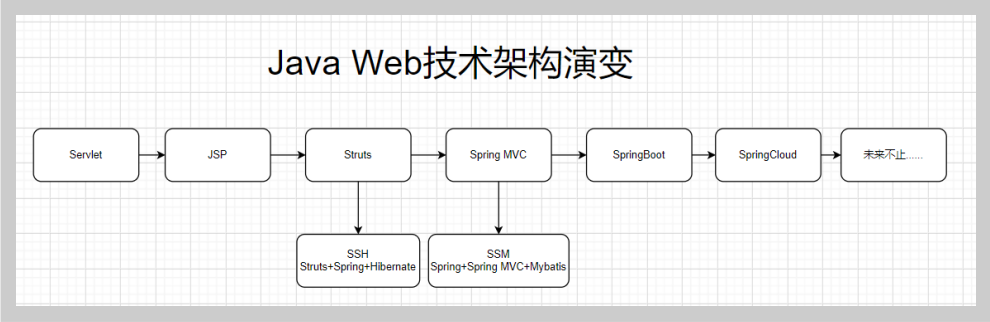
Java Web，是用Java技术来解决相关web互联网领域的技术栈。

web包括：web服务端和web客户端两部分。Java在客户端的应用有Java Applet，不过使用得很少。

Java在服务器端的应用非常的丰富，比如Servlet，JSP、SpringBoot等等。

JavaWeb架构演变过程大致分为以下几个阶段：



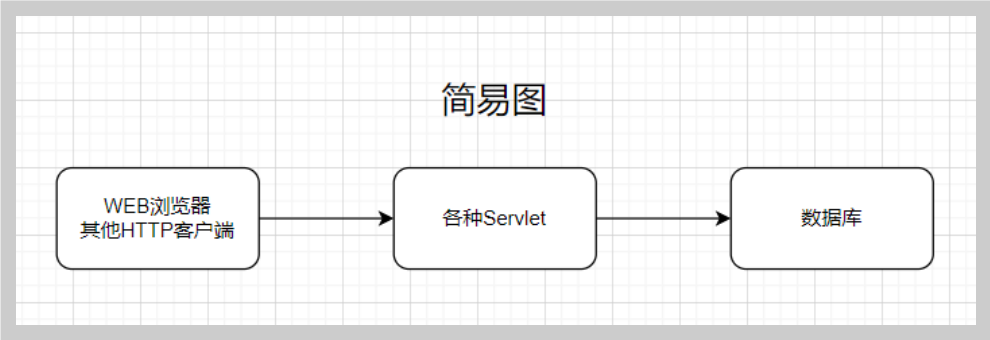
下面，我们通过案例学习这些技术点。

一、Servlet

1.1、什么是Servlet

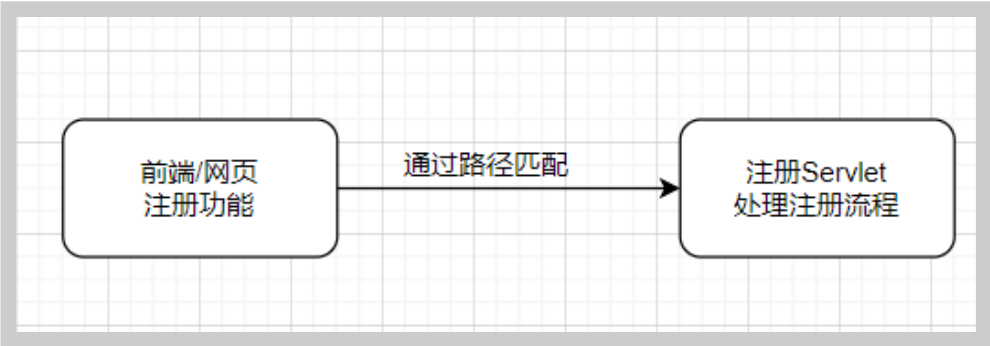
Servlet 是运行在 Web 服务器或应用服务器上的程序，它是作为来自 Web 浏览器或其他 HTTP 客户端的请求和 HTTP 服务器上的数据库或应用程序之间的中间层。

画个简易图，便于理解。



上面是个简易图，用语言描述下流程。

比如，我们打开一个网站有个注册功能，在填写完信息后，点击提交，所填写的信息传输到后端，根据所指向的路径，来匹配对应的servlet，专门用于处理注册流程。



当然，我们还可以有登录Servlet，个人信息Servlet等等。

从代码上来说，Servlets 是 Java 类，服务于 HTTP 请求并实现了 javax.servlet.Servlet 接口。

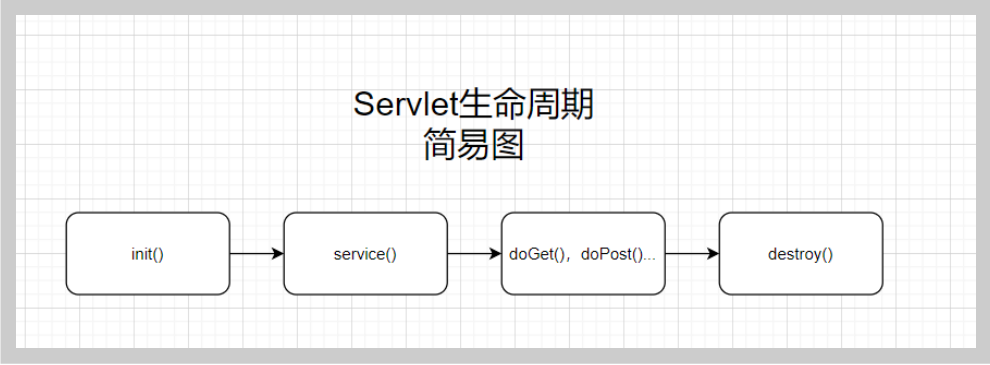
1.2、Servlet生命周期

Servlet生命周期就是从创建到毁灭的过程。

大致是四个阶段。

1. init()：初始化阶段，只被调用一次，也就是在第一次创建Servlet时被调用。
2. service()：服务阶段，主要处理来自客户端的请求，并可以根据HTTP请求类型来调用对应的方法，比如 `doGet()，doPost()，doPut()` 等等。
3. doGet()，doPost()：处理阶段，将主要代码逻辑写在此处。根据不同HTTP请求对应不同方法。
4. destroy()：毁灭阶段，该方法只会被调用一次，即在Servlet生命期结束时被调用。

画个简易图，大致流程如下图所示：



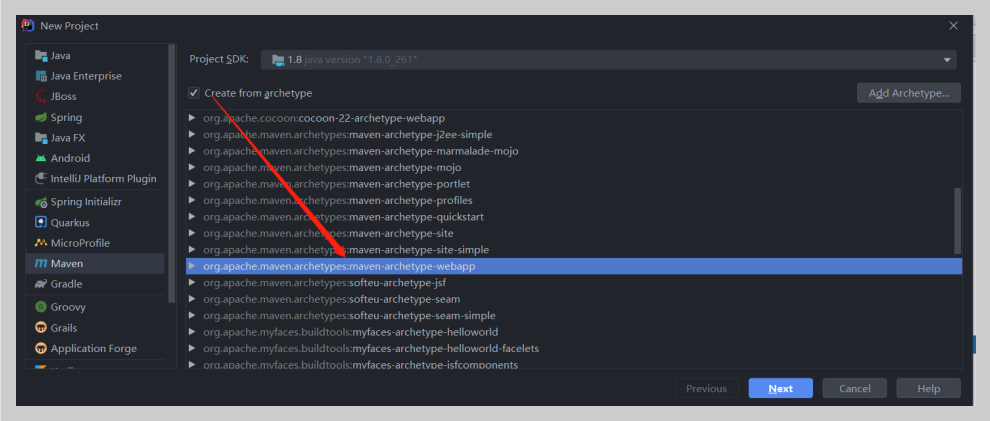
1.3、你的第一个Servlet

光说不练假把式，我们动手写个Servlet案例，进一步理解它。

我们使用Maven创建一个Servlet案例，Maven在 `1.1 Java基础环境搭建之Windows（上）` 中有讲到，可以回顾一下。

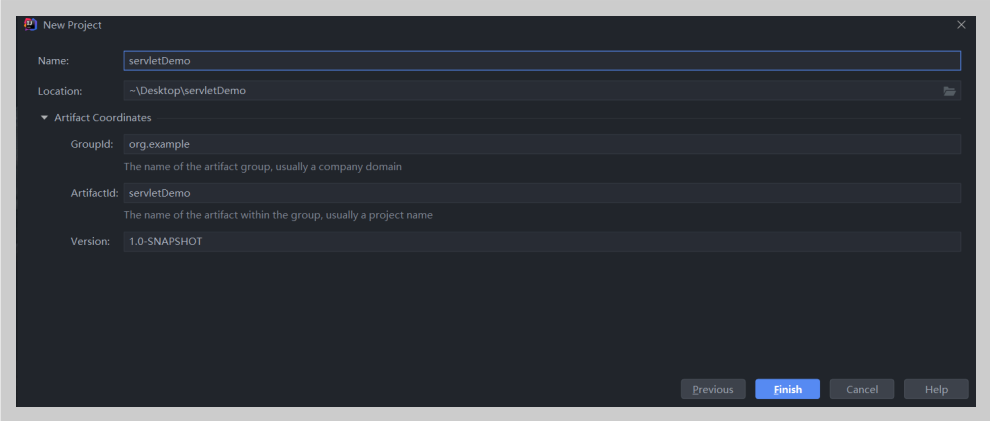
1.3.1、创建项目

①、双击启动IDEA，点击 `Create New Project` （如果默认进入某个项目，需要退出，点击左上角 `File -> Close Project` ），左侧选择 `Maven` 项目，选中 `Create from archetype` 后选择 `maven-archetype-webapp` ，点击Next，如下图所示：

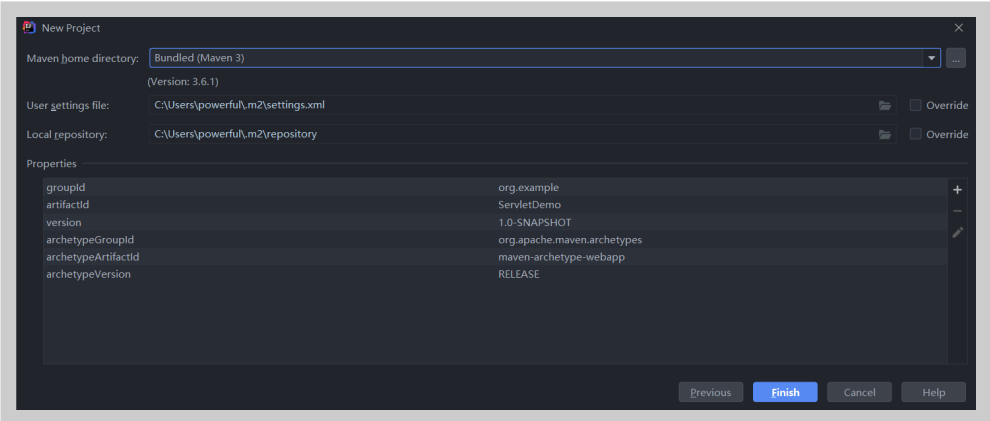


选择 `archetype` 可以简单理解为选择模板，选择不同的模板会有各自的规范。

②、在该页面中起个项目文件名字，其他配置信息默认即可，点击Next，如下图所示：

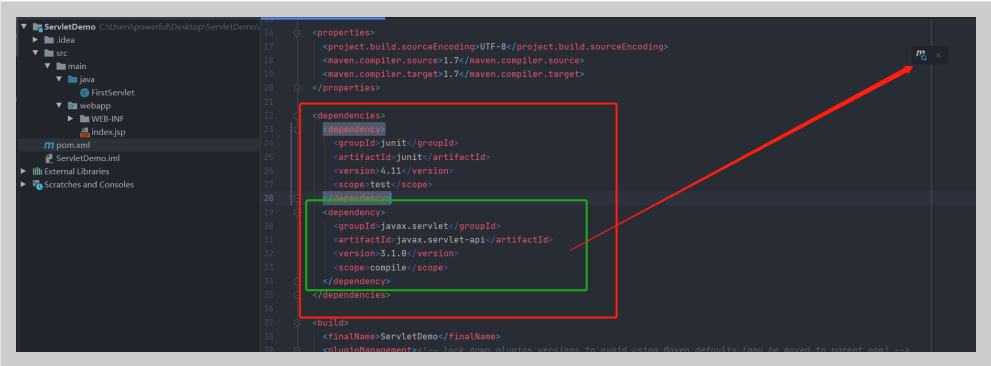


③、这个页面是选择Maven，默认即可，最后点击Finish进入项目，如下图所示：



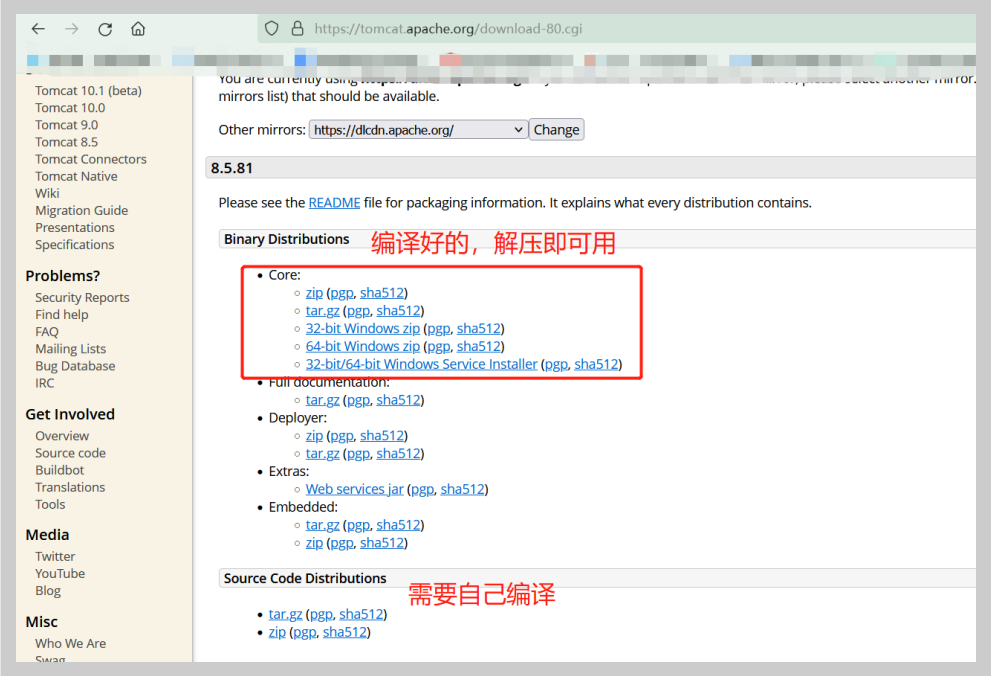
④、在 `pom.xml` 中引入servlet的依赖，写在 `<dependencies>` 标签内，然后可以点击右上角 `加载文件改动` 的按钮（也就是重新加载安装依赖），如下图所示：

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



1.3.2、环境配置Tomcat

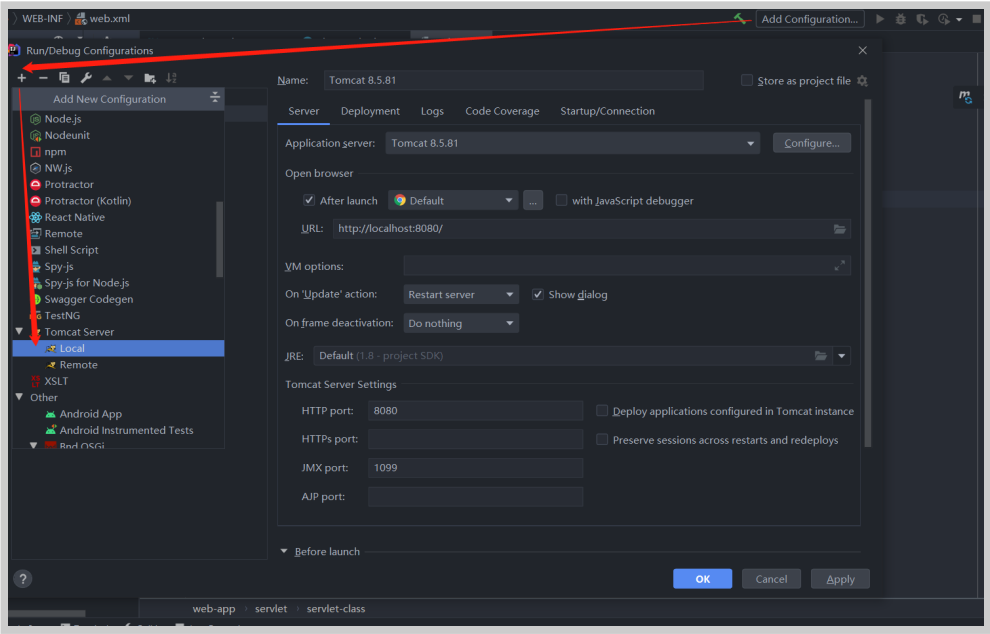
①、访问Tomcat官网 `https://tomcat.apache.org/download-80.cgi` ，下载任意版本Tomcat，我选择的版本是 `8.5.81` ，如下图所示：



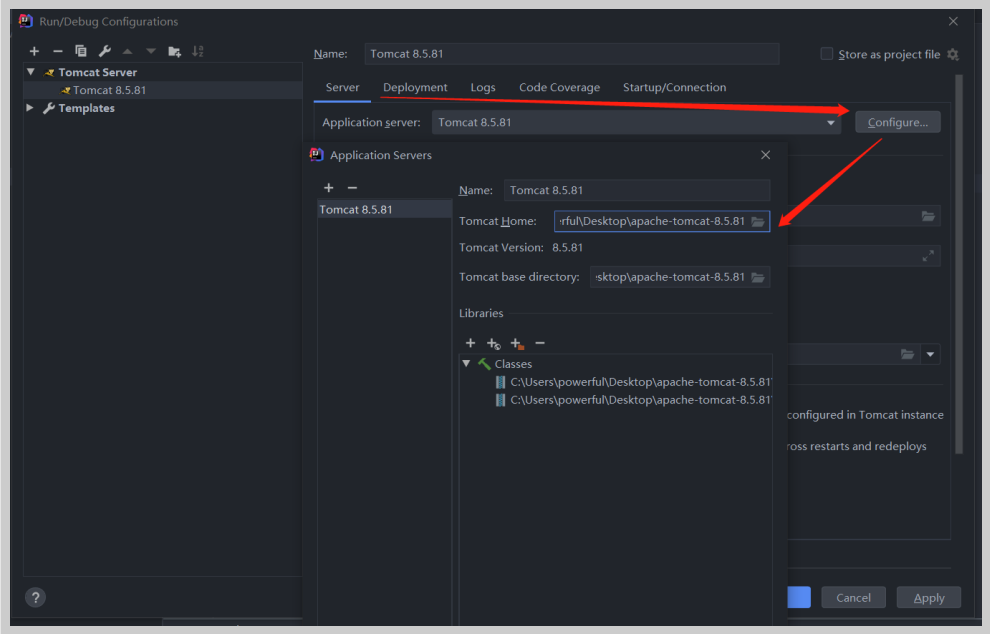
选择 `Binary Distributions` 下的 `Core` 分类，这是Tomcat正式的二进制发布版本，一般学习和开发都用此版本。根据自己计算机系统选择下载项。

②、下载完成后，先解压到桌面备用。

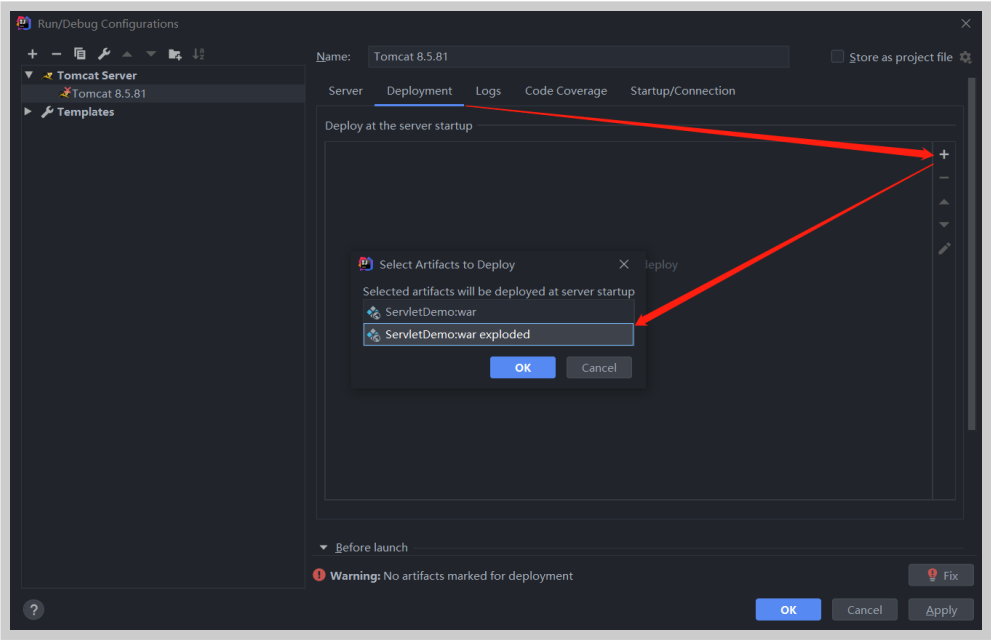
③、进入IDEA，配置Tomcat。在右上侧选择 `Add Configuration...` ，在新的界面点击左上角的 `+` 按钮，滑到下面找到 `Tomcat Server` 点击选择 `Local` ，如下图所示：



④、在Server标签栏下点击 `Configure...` ，进入新的界面，在 `Tomcat Home` 处添加Tomcat，最后点击OK。如下图所示：

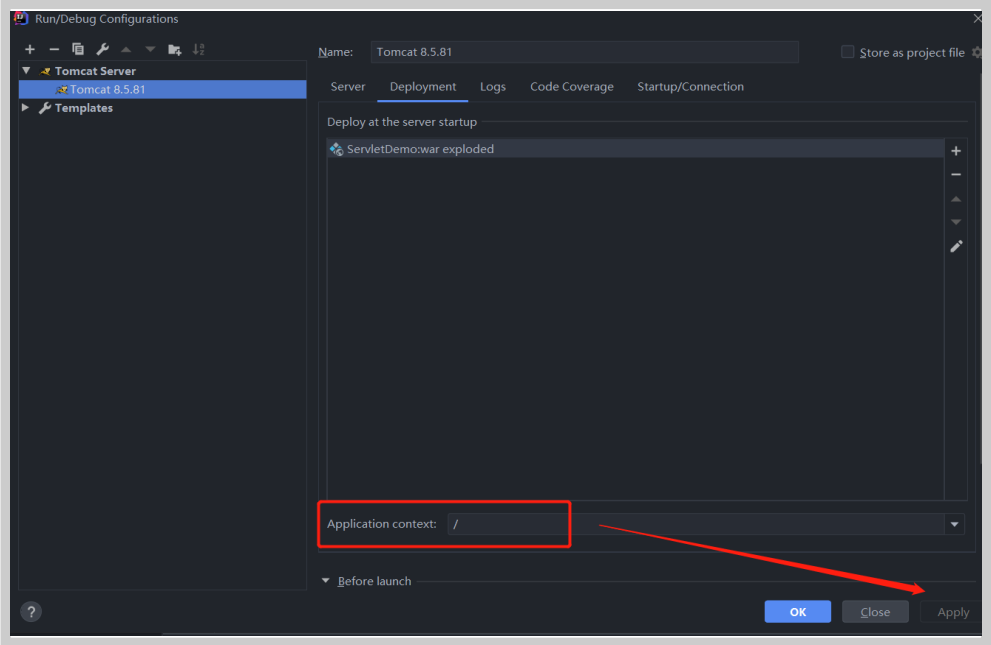


⑤、配置部署方式，选择 `Deployment` 标签栏，在右侧点击 `+` 按钮，选择 `war exploded` 方式，如下图所示：



`war方式`：是发布模式，先打包成war包，再发布。`war exploded方式`：常在开发的时候使用这种方式，它可以支持热部署，但需要设置。跟本次案例调试无关，仅是给大家简单讲讲。

⑥、设置URL根路径后，点击`Apply`，左后点击`OK`。如下图所示：



至此，完成了在IDEA中配置Tomcat。

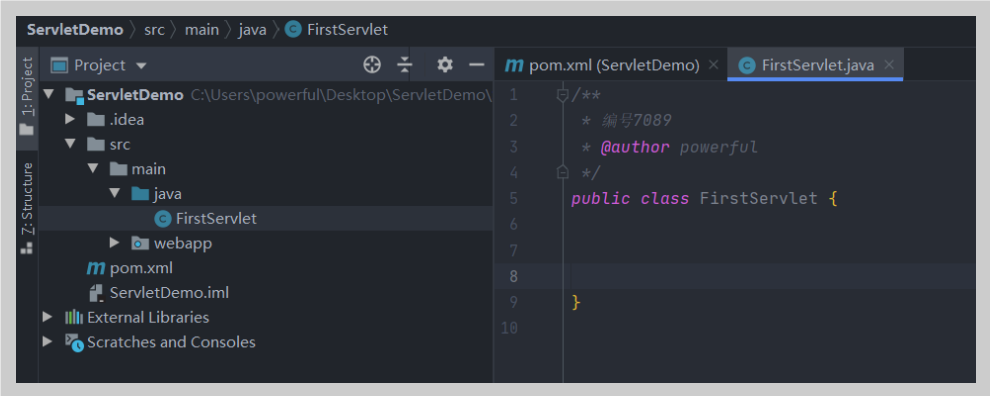
下面我们开始编写代码案例。

1.3.3、编写Servlet并运行

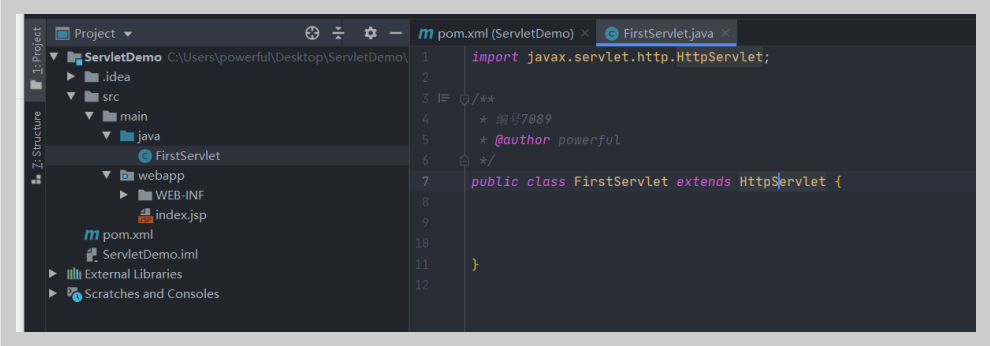
接`1.3.1、创建项目`，在创建好后项目进行下面步骤。

①、鼠标右键点击`main`目录，选择`New - Directory`，创建名为`java`的主目录，后面我们的代码都写在此处。

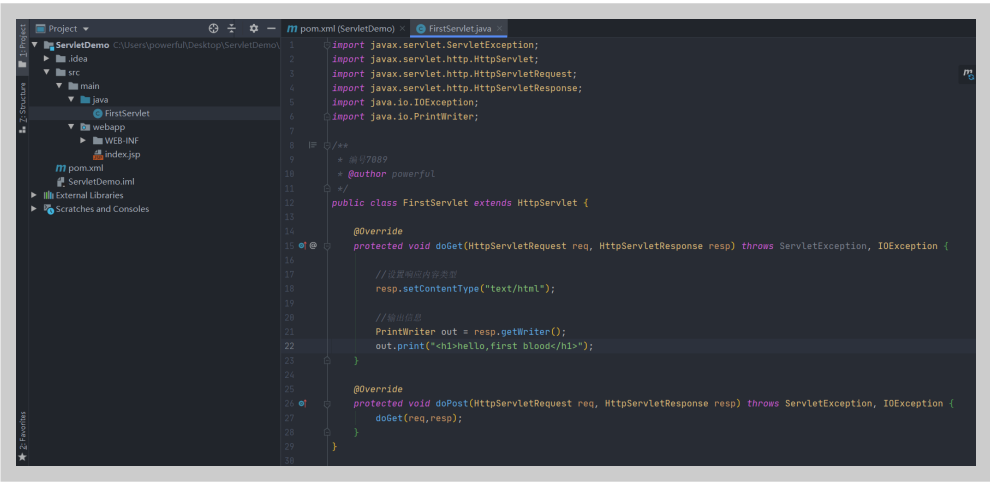
②、鼠标右键点击 `java` 目录，选择 `New - Java Class` ，创建名为 `FirstServlet` 的class文件，在这里面编写我们的代码，最终如下图所示：



③、想要真正运行起来Servlet程序，我们需要继承 `HTTPServlet` ，重写部分方法，如下图所示：



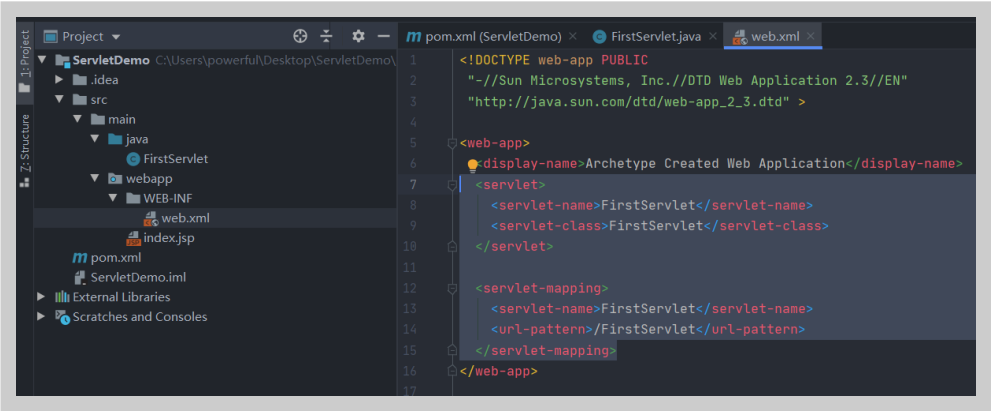
④、编写代码，响应Get和Post请求内容，最终如下图所示：



⑤、在servlet中，需要根据URL路径匹配映射到对应的servlet。即在 `web.xml` 中注册servlet，如下图所示：

```
<servlet>
  <servlet-name>FirstServlet</servlet-name>
  <servlet-class>FirstServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>FirstServlet</servlet-name>
  <url-pattern>/FirstServlet</url-pattern>
</servlet-mapping>
```

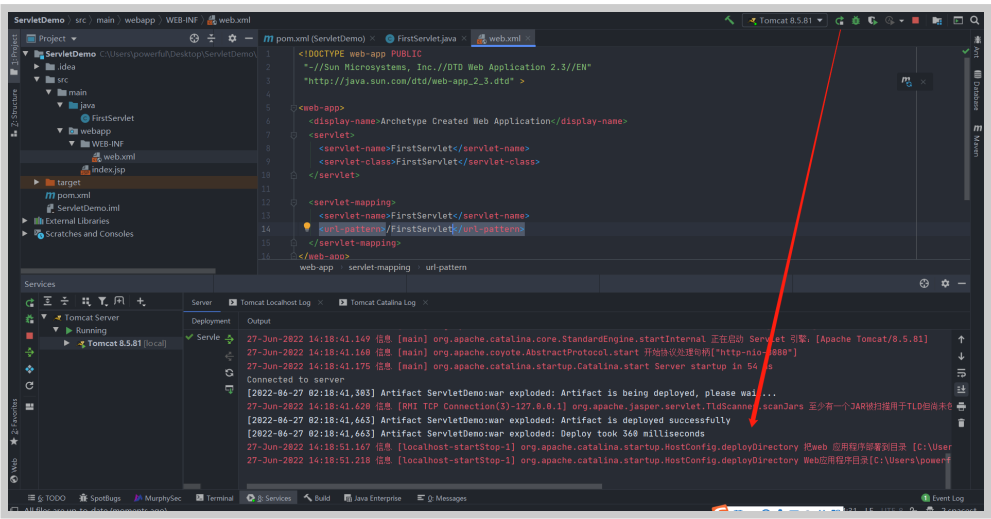


映射匹配流程：`/FirstServlet`路径绑定的`Servlet-name为FirstServlet`，而`FirstServlet绑定的class是FirstServlet`，最终访问`/FirstServlet`，调用的类也就是`FirstServlet.class`。

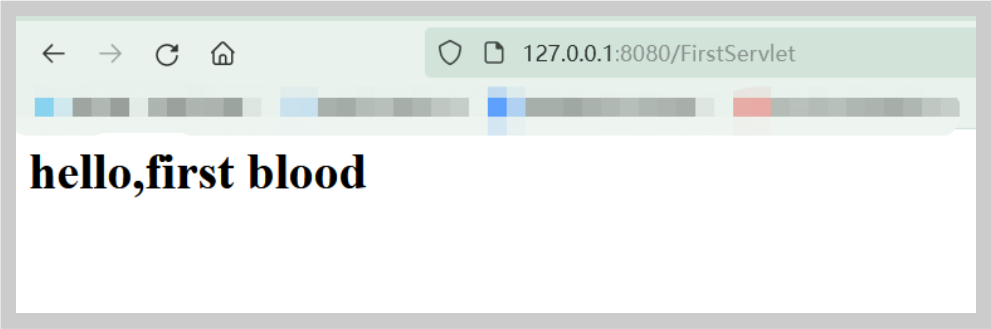
多看两遍就明白了了。

servlet注册有两种方式：一是通过上述`web.xml`进行注册，二是在servlet3以后可以通过`@WebServlet()`注解方式进行注册。

⑥、启动项目，可以点击右上侧运行按钮，稍等片刻，看到控制台输出以下信息即表明启动成功，如下图所示：



⑦、浏览器访问`127.0.0.1:8080/FirstServlet`，即可看到代码中输出的信息，如下图所示：



至此，你的第一个Servlet程序成功运行启动起来了。

现阶段，你只需要关注知道Servlet是什么就好，我相信这个流程走下来你能明白不少。

如果想进一步了解，我从网上找了几个JSP+Servlet的系统，可以进一步研究下。

```
https://github.com/Hui4401/StudentManager
https://github.com/czwbig/Tmall_JavaEE
```

二、过滤器、监听器和拦截器

过滤器：在servlet中，过滤器也就是`Filter`，它主要用于过滤字符编码，做一些统一的业务等等。是使用`javax.servlet.Filter`接口进行实现的。在代码安全中，他常被用于防止XSS，防SQL注入，防任意文件上传等。再配置了Filter之后，它可以统一过滤危险字符，省时省力。

监听器：在servlet中，监听器也就是`Listener`，它主要用于做一些初始化的内容。是使用`javax.servlet.ServletContextListener`接口进行实现的。如果同时有监听器和过滤器，监听器是在过滤器之前启动。

拦截器：依赖WEB框架，在SrpingMvc中就依赖SpringMVC框架。是属于面向切面变成的一种运用。

过滤器和拦截器的区别，分为以下五种：

- 拦截器是基于Java的反射机制的，而过滤器是基于函数回调
- 过滤器依赖与servlet容器，而拦截器不依赖与servlet容器
- 拦截器只能对action请求起作用，而过滤器则可以对几乎所有的请求起作用
- 拦截器可以访问action上下文、值栈里的对象，而过滤器不能
- 在action的生命周期中，拦截器可以多次被调用，而过滤器只能在容器初始化时被调用一次

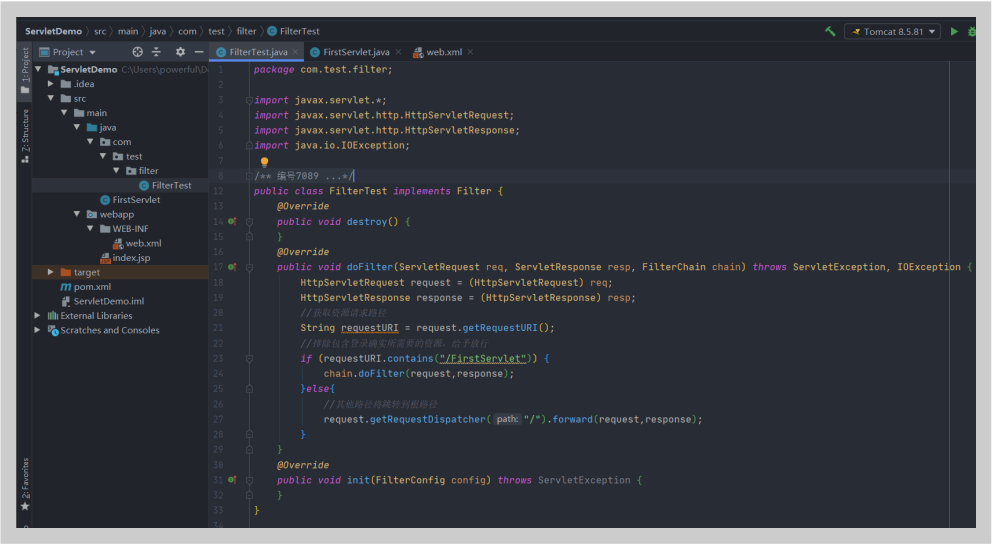
我们现在主要关注一下过滤器。

2.1、过滤器代码

过滤器是使用`javax.servlet.Filter`接口进行实现的。需要使用`doFilter()`方法实现拦截。

以`1.3.3、编写Servlet并运行`代码为基础，继续编写Filter代码。

①、在`main/java`右键点击后选择`New - package`，键入`com.test.filter`新建个filter，编写过滤层代码，如下图所示：

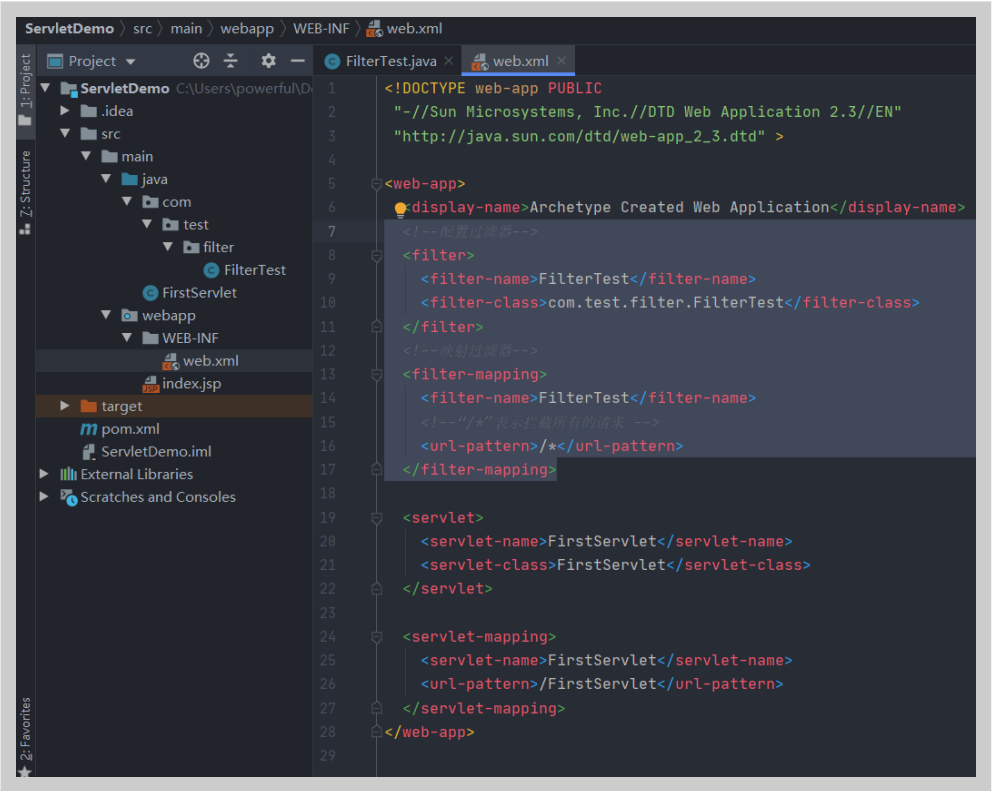


我们重写了 `doFilter()` 方法，代码逻辑如下：

首先通过 `String requestURI = request.getRequestURI();` 获取URL路径。然后对路径进行判断，如果路径中包含 `/FirstServlet` ，则放行。否则就跳转到根路径下。

②、然后在web.xml中配置注册过滤器，如下图所示：

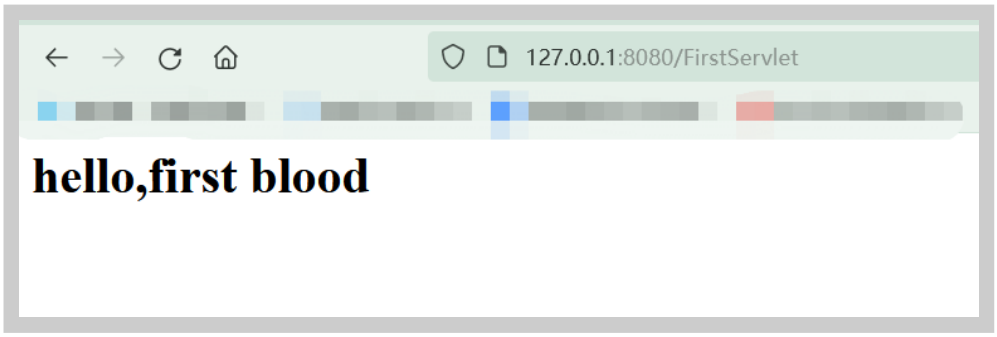
```
<!--配置过滤器-->
<filter>
  <filter-name>FilterTest</filter-name>
  <filter-class>com.test.filter.FilterTest</filter-class>
</filter>
<!--映射过滤器-->
<filter-mapping>
  <filter-name>FilterTest</filter-name>
  <!--“/*”表示拦截所有的请求 -->
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

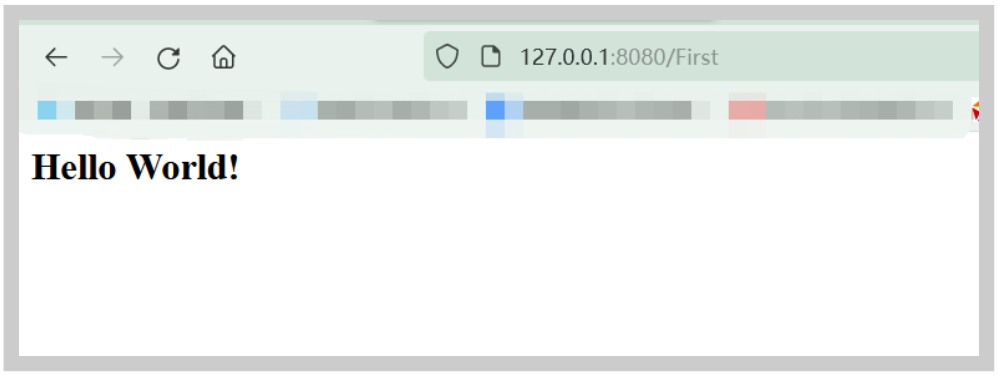


过滤器标签需要要在servlet标签上面，程序会按照注册顺序进行执行。如果涉及多个过滤器，也是按照注册顺序进行执行的。

注册过滤器有两种方式：一是上面通过 `web.xml` 进行注册，另一种是通过 `@WebFilter()` 注解的方式进行注册。

③、运行项目，访问不通路径，观察不同效果。如下图所示：





至此，本节讲解完毕。请大家务必动手操作，有问题积极交流探讨。