

## 04 | 实战：纯手工打造和运行一个Servlet

2019-05-18 李号双

深入拆解Tomcat & Jetty

[进入课程 >](#)



讲述：李号双

时长 08:54 大小 8.16M



作为 Java 程序员，我们可能已经习惯了使用 IDE 和 Web 框架进行开发，IDE 帮我们做了编译、打包的工作，而 Spring 框架在背后帮我们实现了 Servlet 接口，并把 Servlet 注册到了 Web 容器，这样我们可能很少有机会接触到一些底层本质的东西，比如怎么开发一个 Servlet？如何编译 Servlet？如何在 Web 容器中跑起来？

今天我们就抛弃 IDE、拒绝框架，自己纯手工编写一个 Servlet，并在 Tomcat 中运行起来。一方面进一步加深对 Servlet 的理解；另一方面，还可以熟悉一下 Tomcat 的基本功能使用。

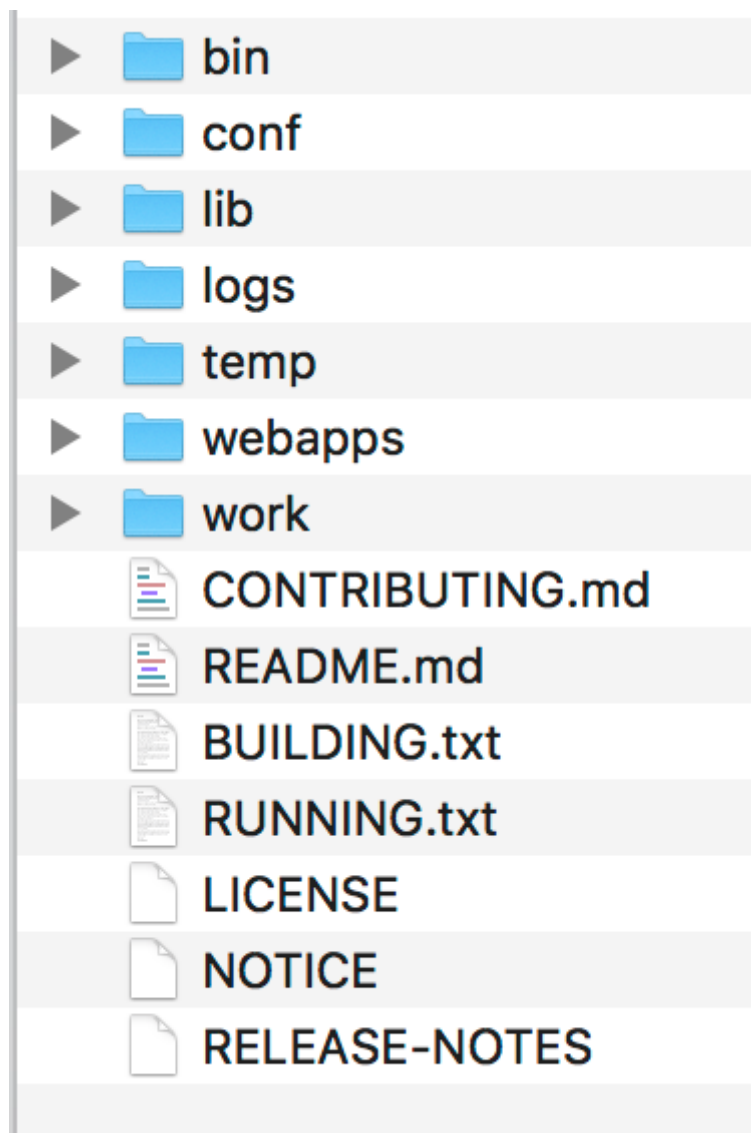
主要的步骤有：

1. 下载并安装 Tomcat。
2. 编写一个继承 HttpServlet 的 Java 类。
3. 将 Java 类文件编译成 Class 文件。
4. 建立 Web 应用的目录结构，并配置 web.xml。
5. 部署 Web 应用。
6. 启动 Tomcat。
7. 浏览器访问验证结果。
8. 查看 Tomcat 日志。

下面你可以跟我一起一步步操作来完成整个过程。Servlet 3.0 规范支持用注解的方式来部署 Servlet，不需要在 web.xml 里配置，最后我会演示怎么用注解的方式来部署 Servlet。

## 1. 下载并安装 Tomcat

最新版本的 Tomcat 可以直接在[官网](#)上下载，根据你的操作系统下载相应的版本，这里我使用的是 Mac 系统，下载完成后直接解压，解压后的目录结构如下。



下面简单介绍一下这些目录：

/bin：存放 Windows 或 Linux 平台上启动和关闭 Tomcat 的脚本文件。

/conf：存放 Tomcat 的各种全局配置文件，其中最重要的是 server.xml。

/lib：存放 Tomcat 以及所有 Web 应用都可以访问的 JAR 文件。

/logs：存放 Tomcat 执行时产生的日志文件。

/work：存放 JSP 编译后产生的 Class 文件。


/webapps：Tomcat 的 Web 应用目录，默认情况下把 Web 应用放在这个目录下。

## 2. 编写一个继承 HttpServlet 的 Java 类

我在专栏上一期提到，javax.servlet 包提供了实现 Servlet 接口的 GenericServlet 抽象类。这是一个比较方便的类，可以通过扩展它来创建 Servlet。但是大多数的 Servlet 都在 HTTP 环境中处理请求，因此 Servlet 规范还提供了 HttpServlet 来扩展 GenericServlet 并

且加入了 HTTP 特性。我们通过继承 `HttpServlet` 类来实现自己的 `Servlet` 只需要重写两个方法：`doGet` 和 `doPost`。

因此今天我们创建一个 Java 类去继承 `HttpServlet` 类，并重写 `doGet` 和 `doPost` 方法。首先新建一个名为 `MyServlet.java` 的文件，敲入下面这些代码：


 复制代码

```
1 import java.io.IOException;
2 import java.io.PrintWriter;
3
4 import javax.servlet.ServletException;
5 import javax.servlet.http.HttpServlet;
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.http.HttpServletResponse;
8
9
10 public class MyServlet extends HttpServlet {
11
12     @Override
13     protected void doGet(HttpServletRequest request, HttpServletResponse response)
14         throws ServletException, IOException {
15
16         System.out.println("MyServlet 在处理 get () 请求...");
17         PrintWriter out = response.getWriter();
18         response.setContentType("text/html;charset=utf-8");
19         out.println("<strong>My Servlet!</strong><br>");
20     }
21
22     @Override
23     protected void doPost(HttpServletRequest request, HttpServletResponse response)
24         throws ServletException, IOException {
25
26         System.out.println("MyServlet 在处理 post () 请求...");
27         PrintWriter out = response.getWriter();
28         response.setContentType("text/html;charset=utf-8");
29         out.println("<strong>My Servlet!</strong><br>");
30     }
31
32 }
```

这个 `Servlet` 完成的功能很简单，分别在 `doGet` 和 `doPost` 方法体里返回一段简单的 HTML。

### 3. 将 Java 文件编译成 Class 文件

下一步我们需要把 MyServlet.java 文件编译成 Class 文件。你需要先安装 JDK，这里我使用的是 JDK 10。接着你需要把 Tomcat lib 目录下的 servlet-api.jar 拷贝到当前目录下，这是因为 servlet-api.jar 中定义了 Servlet 接口，而我们的 Servlet 类实现了 Servlet 接口，因此编译 Servlet 类需要这个 JAR 包。接着我们执行编译命令：


 复制代码

```
1 javac -cp ./servlet-api.jar MyServlet.java
```

编译成功后，你会在当前目录下找到一个叫 MyServlet.class 的文件。

## 4. 建立 Web 应用的目录结构

我们在上一期学到，Servlet 是放到 Web 应用部署到 Tomcat 的，而 Web 应用具有一定的目录结构，所有我们按照要求建立 Web 应用文件夹，名字叫 MyWebApp，然后在这个目录下建立子文件夹，像下面这样：

 复制代码

```
1 MyWebApp/WEB-INF/web.xml
2
3 MyWebApp/WEB-INF/classes/MyServlet.class
```

然后在 web.xml 中配置 Servlet，内容如下：

 复制代码

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5   http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
6   version="4.0"
7   metadata-complete="true">
8
9   <description> Servlet Example. </description>
10  <display-name> MyServlet Example </display-name>
11  <request-character-encoding>UTF-8</request-character-encoding>
12
13  <servlet>
```

```
14     <servlet-name>myServlet</servlet-name>
15     <servlet-class>MyServlet</servlet-class>
16 </servlet>
17
18 <servlet-mapping>
19     <servlet-name>myServlet</servlet-name>
20     <url-pattern>/myservlet</url-pattern>
21 </servlet-mapping>
22
23 </web-app>
```

你可以看到在 web.xml 配置了 Servlet 的名字和具体的类，以及这个 Servlet 对应的 URL 路径。请你注意，**servlet** 和 **servlet-mapping** 这两个标签里的 **servlet-name** 要保持一致。

## 5. 部署 Web 应用

Tomcat 应用的部署非常简单，将这个目录 MyWebApp 拷贝到 Tomcat 的安装目录下的 webapps 目录即可。

## 6. 启动 Tomcat

找到 Tomcat 安装目录下的 bin 目录，根据操作系统的不同，执行相应的启动脚本。如果是 Windows 系统，执行 startup.bat.; 如果是 Linux 系统，则执行 startup.sh。

## 7. 浏览访问验证结果

在浏览器里访问这个 URL: `http://localhost:8080/MyWebApp/myservlet`，你会看到：

 复制代码

```
1 My Servlet!
```

这里需要注意，访问 URL 路径中的 MyWebApp 是 Web 应用的名字，myservlet 是在 web.xml 里配置的 Servlet 的路径。



## 8. 查看 Tomcat 日志

打开 Tomcat 的日志目录，也就是 Tomcat 安装目录下的 logs 目录。Tomcat 的日志信息分为两类：一是运行日志，它主要记录运行过程中的一些信息，尤其是一些异常错误日志信息；二是访问日志，它记录访问的时间、IP 地址、访问的路径等相关信息。

这里简要介绍各个文件的含义。

```
catalina.***.log
```

主要是记录 Tomcat 启动过程的信息，在这个文件可以看到启动的 JVM 参数以及操作系统等日志信息。

```
catalina.out
```

catalina.out 是 Tomcat 的标准输出 (stdout) 和标准错误 (stderr)，这是在 Tomcat 的启动脚本里指定的，如果没有修改的话 stdout 和 stderr 会重定向到这里。所以在这个文件里可以看到我们在 MyServlet.java 程序里打印出来的信息：

```
MyServlet 在处理 get() 请求...
```

```
localhost.***.log
```

主要记录 Web 应用在初始化过程中遇到的未处理的异常，会被 Tomcat 捕获而输出这个日志文件。

```
localhost_access_log.***.txt
```


存放访问 Tomcat 的请求日志，包括 IP 地址以及请求的路径、时间、请求协议以及状态码等信息。

```
manager.***.log/host-manager.***.log
```

存放 Tomcat 自带的 manager 项目的日志信息。

## 用注解的方式部署 Servlet

为了演示用注解的方式来部署 Servlet，我们首先修改 Java 代码，给 Servlet 类加上 **@WebServlet** 注解，修改后的代码如下。

 复制代码

```
1 import java.io.IOException;
2 import java.io.PrintWriter;
3
4 import javax.servlet.ServletException;
5 import javax.servlet.annotation.WebServlet;
6 import javax.servlet.http.HttpServlet;
7 import javax.servlet.http.HttpServletRequest;
8 import javax.servlet.http.HttpServletResponse;
9
10 @WebServlet("/myAnnotationServlet")
11 public class AnnotationServlet extends HttpServlet {
12
13     @Override
14     protected void doGet(HttpServletRequest request, HttpServletResponse response)
15         throws ServletException, IOException {
16
17         System.out.println("AnnotationServlet 在处理 get () 请求...");
18         PrintWriter out = response.getWriter();
19         response.setContentType("text/html; charset=utf-8");
20         out.println("<strong>Annotation Servlet!</strong><br>");
21
22     }
23
24     @Override
25     protected void doPost(HttpServletRequest request, HttpServletResponse response)
26         throws ServletException, IOException {
27
28         System.out.println("AnnotationServlet 在处理 post () 请求...");
29         PrintWriter out = response.getWriter();
30         response.setContentType("text/html; charset=utf-8");
31         out.println("<strong>Annotation Servlet!</strong><br>");
32
33     }
34
35 }
```

这段代码里最关键的就是这个注解，它表明两层意思：第一层意思是 AnnotationServlet 这个 Java 类是一个 Servlet，第二层意思是这个 Servlet 对应的 URL 路径是 myAnnotationServlet。



```
1 @WebServlet("/myAnnotationServlet")
```

创建好 Java 类以后，同样经过编译，并放到 MyWebApp 的 class 目录下。这里要注意的是，**你需要删除原来的 web.xml**，因为我们不需要 web.xml 来配置 Servlet 了。然后重启 Tomcat，接下来我们验证一下这个新的 AnnotationServlet 有没有部署成功。在浏览器里输入：`http://localhost:8080/MyWebApp/myAnnotationServlet`，得到结果：

```
1 Annotation Servlet!
```

这说明我们的 AnnotationServlet 部署成功了。可以通过注解完成 web.xml 所有的配置功能，包括 Servlet 初始化参数以及配置 Filter 和 Listener 等。

## 本期精华

通过今天的学习和实践，相信你掌握了如何通过扩展 HttpServlet 来实现自己的 Servlet，知道了如何编译 Servlet、如何通过 web.xml 来部署 Servlet，同时还练习了如何启动 Tomcat、如何查看 Tomcat 的各种日志，并且还掌握了如何通过注解的方式来部署 Servlet。我相信通过专栏前面文章的学习加上今天的练习实践，一定会加深你对 Servlet 工作原理的理解。之所以我设置今天的实战练习，是希望你知道 IDE 和 Web 框架在背后为我们做了哪些事情，这对于我们排查问题非常重要，因为只有我们明白了 IDE 和框架在背后做的事情，一旦出现问题的时候，我们才能判断它们做得对不对，否则可能开发环境里的一个小问题就会折腾我们半天。

## 课后思考

我在 Servlet 类里同时实现了 doGet 方法和 doPost 方法，从浏览器的网址访问默认访问的是 doGet 方法，今天的课后思考题是如何访问这个 doPost 方法。

不知道今天的内容你消化得如何？如果还有疑问，请大胆的在留言区提问，也欢迎你把你的课后思考和心得记录下来，与我和其他同学一起讨论。如果你觉得今天有所收获，欢迎你把它分享给你的朋友。

# 深入拆解 Tomcat & Jetty

从源码角度深度探索 Java 中间件

李号双

eBay 技术主管



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 03 | 你应该知道的Servlet规范和Servlet容器

下一篇 05 | Tomcat系统架构（上）：连接器是如何设计的？

## 精选留言 (33)

写留言



feitian

2019-05-20

46

既然是纯手工，就应该把servlet那套完整的写出来，不应该再用tomcat容器而应该手写实现tomcat的核心代码。最核心的应该是类似HttpServlet功能的实现，把这个从最初的servlet接口实现了才算讲透，不然还是有点和稀泥的感觉。我觉得应该按照这种思路讲会更好，请参考我写的mytomcat, <https://github.com/feifa168/mytomcat>

展开



Monday

2019-05-18

6

- 1、postman
- 2、curl 命令发送post

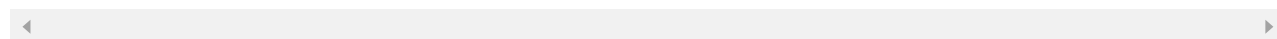
### 3、用HttpClient发送

周六早上坚持打卡，本章节绝大多数知识以前有接触过，只有@WebServlet注解是新知...  
展开 ▾

作者回复: 1. 你可以向Servlet容器注册多个Servlet。

2. 你看这个有没有帮助

<https://github.com/spring-projects/spring-framework/issues/14296>



allea

2019-05-18

👍 4

IDE和框架诞生之初是为了让程序员从繁琐的底层配置中抽离出来专注于业务开发，然而大多数人在享受IDE和框架带来的便捷时，也成了温水里的青蛙，对于实现原理认识模糊，渐渐沦落为一个CRUD，这不是一件好事，啊～幡然醒悟



今夜秋风和

2019-05-18

👍 4

老师，验证的时候默认增加了 `super.doGet(req, resp);`在http1.1写一下不能工作，查看 `HttpServlet` 源码里面 对协议做了限制，http 1.1 协议默认不支持。这个为什么是这样设计的呢？

源代码:

```
String protocol = req.getProtocol();...
```

展开 ▾

作者回复: `super.doGet(req, resp);` 调的是 `HttpServlet` 的 `doGet` 方法，但是这个 `doGet` 需要你去实现的。

`HttpServlet` 的 `service` 方法会调 `doXXX` 方法，并且 `HttpServlet` 里的各种 `doXXX` 方法的默认实现都是直接返回错误。

为什么 `HttpServlet` 要这样设计呢？这是因为它需要做一个限制：程序员要么重写 `HttpServlet` 的 `service` 方法，要么重写 `HttpServlet` 的 `doXXX` 方法。

`web.xml` 和注解可以同时工作，你需要把 `web.xml` 中的 `metadata-complete="true"` 设置成 `false`。

Tomcat启动时会扫描注解，同时记下Servlet的名字和映射路径，如果你设置了延迟记载Servlet，通过浏览器访问时Tomcat才会加载和实例化Servlet。



风翺

2019-05-18

👍 4

可以利用工具，例如postman。也可以编写代码，利用http的post方法去调用。或者像楼上所说的不管通过get还是post都通知转发到doPost中。



youknowmj...

2019-05-19

👍 3

@Amanda 不知道怎么直接回复你，我跟你一样也是遇到乱码的问题。我也设置了response.setCharacterEncoding(utf8)，在getWriter之前，但依然是一样的乱码。原因在与javac编译生成的class文件是用的gbk的编码。换句话说，你生成的class源文件就已经是中文乱码了。

所以在javac的时候加上 -encoding UTF-8就好了。...

展开 ▾



Amanda

2019-05-18

👍 2

老师，实践中发现个问题：虽然response.setContentType("text/html;charset=utf-8")，但是out.println中有输出中文还是乱码的

作者回复：调下顺序，像下面这样：

```
response.setContentType("text/html; charset=utf-8");
PrintWriter out = response.getWriter();
```

getWrite的源码如下：

-----

```
public PrintWriter getWriter()
    throws IOException {

    if (usingOutputStream) {
        throw new IllegalStateException
            (sm.getString("coyoteResponse.getWriter.ise"));
    }
}
```

```

if (ENFORCE_ENCODING_IN_GET_WRITER) {
    /*
     * If the response's character encoding has not been specified as
     * described in <code>getCharacterEncoding</code> (i.e., the method
     * just returns the default value <code>ISO-8859-1</code>),
     * <code>getWriter</code> updates it to <code>ISO-8859-1</code>
     * (with the effect that a subsequent call to <code>getContentType()</code> will
     * include a charset=ISO-8859-1 component which will also be
     * reflected in the Content-Type response header, thereby satisfying
     * the Servlet spec requirement that containers must communicate the
     * character encoding used for the servlet response's writer to the
     * client).
     */
    setCharacterEncoding(getCharacterEncoding());
}

usingWriter = true;
outputBuffer.checkConverter();
if (writer == null) {
    writer = new CoyoteWriter(outputBuffer);
}
return writer;
}
-----

```

你看注释里它说：如果调这个方法之前没有指定Response的字符编码，就用默认的ISO-8859-1，ISO-8859-1不包括中文字符。



**Geek\_Odb34...**

2019-05-18

👍 2

表单提交method=post 就可以啦

展开 ▾



**KL3**

2019-05-18

👍 2

把业务逻辑写在dopost里，然后doget方法调用dopost方法

展开 ▾



**Geek\_ebda9...**

2019-05-23

👍 1

李老师，请教一个问题，你这里所说的servlet和spring mvc里面的controller是什么关系，servlet里面可以直接接收请求，处理请求业务，controller只是通过dispatch servlet再接入进来的？

展开 ▾

作者回复: 你说的没错，具体是这样的 Tomcat的Wrapper组件-Filter-DispatcherServlet-Controller



**Shmily**

2019-05-23

👍 1

这个demo 大学期间一直这样写啊 最后才用Spring的

展开 ▾



**Howard**

2019-05-22

👍 1

代码放在github上需要注意开源协议，像apache这种协议明确说明受美国出口管制的



**张济增**

2019-05-20

👍 1

看着前几章跟 head first servlet jsp差不多，先来个基础

展开 ▾



**Geek\_bde3d...**

2019-05-18

👍 1

最简单的办法就是在doGet里调用doPost 😊

展开 ▾





**darren**  
2019-05-18

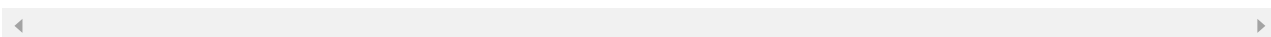


发现xml与注解不能同时起作用，那在用xml方式的老项目中就没办法使用注解的方式了吗？

作者回复: web.xml 和注解可以同时工作的。

例子中的web.xml和注解不能同时工作的原因是web.xml中的配置metadata-complete="true"，你需要把它设置成metadata-complete="false"。

metadata-complete为true的意思是，告诉Tomcat不要去扫描Servlet注解了。



**Tomato**  
2019-06-01



老师，请问一下，在webapps中，把一个项目的文件夹名称改成aaa#bbb时，url路径要用.../aaa/bbb/index.html. 这个后面的原理是什么啊？

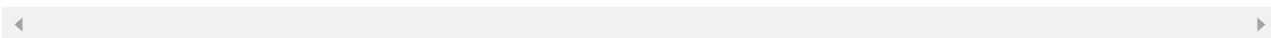


**802.11**  
2019-05-31



老师，我在windows启动Tomcat，点击bin下的startup.bat，一个命令行黑框一闪而过，一直无法成功启动。求助

作者回复: 看看启动日志，或者在命令窗口执行启动脚本，看错误日志是什么

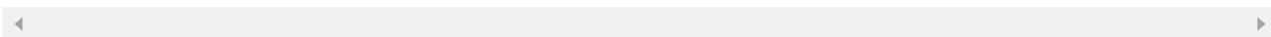


**桔子**  
2019-05-31



李老师，doGet方法的request和response的初始化代码在哪里呢，只知道是servlet容器创建的，但是去哪里可以看到容器初始化response的源码呢。

作者回复: 在Tomcat中CoyoteAdapter类的service方法里







**Geek 1eaf1...**

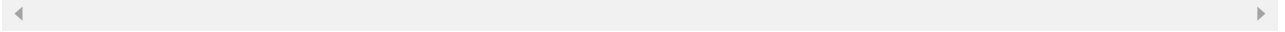
2019-05-27



`javac -cp ./servlet-api.jar MyServlet.java`

这个不是很理解老师

作者回复: 就是编译一个java类



**吃饺子不吐...**

2019-05-27



记录下tomcat工程目录结构:

webapps

|

|

myWebApp...

展开 ▾