

## 简单粗暴，详细得不要不要的 JavaWeb快速入门实例（1）

额，有些标题党的嫌疑，小细节不用在意哈。。。

前端时间我在写一个系列，是关于JavaWeb的一个入门级项目实战，我的初衷就是打算写给初学者的，希望能对他们有所帮助。

这段时间博主也接触了一些事情，感觉有必要专门把JSP的基础拿出来说明一下，因为还是很多人和我说看不懂。

其实写博文真的是挺有挑战性的一件事情，如何把知识点用通俗易懂的语言来表述出来呢?这真的是不太容易的。

首先，写教程需要大量的时间成本，我之前学习JavaWeb的时候，网上搜到的大部分博客，真心话我看不懂。因为那时候的水平太菜了，看视频又嫌太慢。

当时我还喜欢抱怨，说网上那些大牛明明水平很高，可为什么都不肯用大白话把技术点给讲出来呢？当时我记得，遇到问题上网随便一搜，看到的大部分文章，都是代码一贴就完了。最多就是在文章最后写一行字说明一下。

这对当时小白的我真的是心累，而且，我往往看了很多博客，发现里面的代码都是一样的。这个时候就挺郁闷了。

博客精简一些自然有精简的好处，可是那主要是对于有好几年工作经验的程序员而言的，对新手来说，无疑增加了阅读的难度。

那时候我总想着，要是有一个真正意义上的，完全面向初学者的博客系列就好了，而不是代码一贴，你们自己去意会吧。

好在博主终于度过了那个难熬的阶段，然后某一天发现了简书这么好的互联网产品，凭着一腔激情和冲动，还有一丝天真，开始写我的第一篇博客。

写了之后才发现，写教程不难，可是要写出那种通俗易懂的教程是非常耗时间的，有的时候，我感觉知识点就应该是这样的，可是为了让新人容易理解，我不得不花时间去组织语言和编写案例。

好几次我都想退出简书不写了，因为写教程真的很累。可是看到自己写的文章阅读量越来越多，而且大部分看我文章的人都是处于迷茫期的朋友。

我仿佛看到了当年，刚走出校门的我自己。

于是，我决定继续写下去，只是更新的话，不可能像之前有段时间那样，仿佛打了鸡血似地一天一更了。不过我还是尽量保证每周一到两更，当然，断更也是有可能的。（额，我感觉自己有点轻微的洁癖，我竟然总是忍不住要区分 **的，地，得**）

当然，不管怎样，那个文章发布系统我肯定会坚持写完的，毕竟做出了承诺。

---

我是萌萌的分割线

---

好的，本系列对JSP，以及如何建立JavaWeb工程做一个讲解，如果你在收看《文章发布系统》系列，而且基础相对有些薄弱的话，也许这个系列可以帮到你。

这也算是《文章发布系统》系列的一个小分支，我会尽快写完，为什么要写呢？因为我发觉看我文章的大部分人还是初学者，有的甚至连web项目是啥都不知道，于是，我感觉有必要把基础的东西做一个总结。

### 1. 实验环境准备

（假设你已经装好了jdk，如果不会安装jdk，请参考[这篇文章](#)）

**1.1** 安装一个eclipse或者MyEclipse（本文以eclipse为例），其实所谓的安装就是网上去下载一个eclipse，然后解压一下就好了。

**1.2** 安装tomcat，网上下载一个，然后解压一下。我这次使用的tomcat容器配的是8080端口。

**1.3** 将tomcat配置到eclipse中。具体步骤为：Window -- Preferences -- Server -- Runtime Environment -- Add 将tomcat解压后的路径配置上去就行了。

本文假设读者已经对tomcat，eclipse有一定的了解。

### 2. 新建web项目

首先，一言不合就打开eclipse



我们新建一个项目，File -- new -- Dynamic Web Project。

项目名称是web，点击Next

New Dynamic Web Project

**Dynamic Web Project**

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: web

Project location

☒ Use default location

Location: D:\software\work\eclipse\workspace\web Browse...

Target runtime

Apache Tomcat v7.0 New Runtime...

Dynamic web module version

3.0

Configuration

Default Configuration for Apache Tomcat v7.0 Modify...

A good starting point for working with Apache Tomcat v7.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

☐ Add project to an EAR

EAR project name: EAR New Project...

Working sets

☐ Add project to working sets

Working sets: Select...

< Back Next > Finish Cancel

New Dynamic Web Project

**Java**

Configure project for building a Java application.

Source folders on build path:

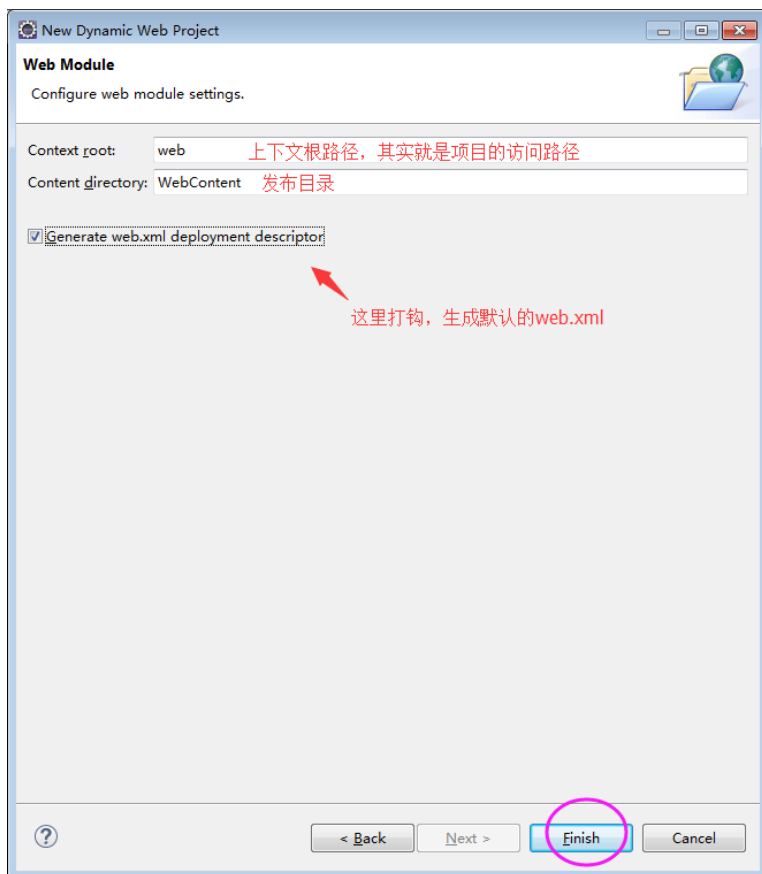
src Add Folder... Edit... Remove

Default output folder:

build\classes

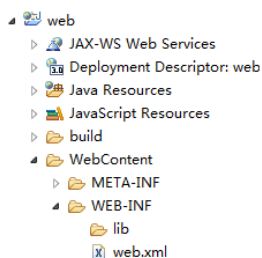
< Back Next > Finish Cancel

build\classes是默认的编译目录，还是点击Next



好的，最后点击Finish，一个web项目就建立好了。

目录结构



## 2.啥叫JavaWeb项目，它能干嘛？

JavaWeb项目就是一个应用程序，你不要以为它有多么神秘。你每天打开的QQ就是一个应用程序，你在手机上打开的微信，百度地图等等，这些都是应用程序。

不同的是，QQ是一个本地客户端程序，它需要你自己在自己的电脑上安装一个客户端，然后你打开QQ，可以登录，聊天。

QQ肯定也分客户端和服务端，我们电脑上装的就是客户端，你要知道的一点是，我们发送消息，发送到哪里去了呢？

没错，肯定是发送到QQ的服务器上了，那里保存着我们的所有数据。

想想也不可能在本地上，因为我们在电脑上发送的消息，在手机QQ上不是也能看到吗？这就说明，肯定是上传到它的服务器然后保存下来了。

而Web项目是什么呢？

它也是一个应用程序，只不过它的客户端是运行在浏览器上的。

我们打开浏览器，访问一个地址，比如 <http://www.jianshu.com/>

这个就是我们的访问路径，我们通过这个URL向简书的服务器提交请求，然后服务器进行处理，给你回应。

接下来，我们就看到出来一个网页。

我们在这个网页上可以写文章，点赞，收藏，等等。

我们的每一步操作，其实都和服务器做了一个交互。

比如一个登录功能，当我们鼠标点击那个登录按钮，浏览器就向服务器提交了一个请求，服务器进行处理，然后操作数据库。最终，它会得出一个结果，就是说，到底能不能让你登录，总会有个说法。

如果你密码写错了，它可能就会得出一个用户名或密码错误的结论。然后，这个信息会从服务器返回到前台，前台就是浏览器。

所谓的前台就是你的浏览器，没什么神秘的。

前台接受到后台传过来的信息后，进行一些处理，然后给你个提示，比如，告诉你用户名或密码错误。

这就是一个完整的交互过程。

现在随着网速越来越好，浏览器的性能越来越强大，我想web肯定是未来的大趋势。毕竟，谁都不希望在电脑上安装一大堆客户端吧。

只用一个浏览器多好。

有了web，我只需要有一个浏览器，然后就能通过互联网获取我想要的资源了。这样不是很美妙吗？

近几年H5非常流行，尤其是移动端，因为手机浏览器基本上都支持css3。一样的道理，我们肯定也不希望在手机上安装一大堆应用，卡都卡死了，如果能直接访问一个网页，就能获得我想要的服务就好了。

而且现在WIFI也普遍了，如果以后流量能没有限制，那么APP的热度很可能会逐渐被web取代。

当然，现在是不太可能的，毕竟访问网页要流量啊，而我在手机上安装了APP，就好像电脑的QQ一样，大部分资源文件就在我本地，我不需要每次都去联网下载。比如图片，app就没事，因为就在本地，可是如果用web，我每次访问那个网页，就需要把图片重新下载一遍。这样不是很浪费流量吗？

### 3. 手工搭建web项目

现在，为了说明服务器和web项目的概念，我们先不用eclipse，来手工搭建一个web项目。

服务器，正常情况下就是一台配置高一点的电脑，除非是那种大型的专用服务器。

一般来说，所谓的服务器，就是电脑。

比如我现在有一台电脑，给他装了一个linux系统或者windos系统，然后我说，好了，从今以后，这就是服务器了。现在服务器一般都是用linux系统的。

那么tomcat又是什么？

tomcat其实也是一个应用程序，你网上下载的tomcat往往是一个压缩包，然后我们解压以后就相当于安装好了。

可以这么理解：

**服务器就是一台电脑，而tomcat是一个容器，专门存放web项目的容器。**

以下我都将tomcat称为tomcat容器。

我们看到在tomcat容器根目录下，有一个webapps文件夹

里面是这样的：

名称	修改日期	类型	大小	
docs	2016/9/30 17:36	文件夹		
examples	2016/9/30 17:36	文件夹		
host-manager	2016/9/30 17:36	文件夹		
manager	2016/9/30 17:36	文件夹		
ROOT	2016/9/30 17:36	文件夹		

好的，现在我要发布一个项目了，我只需要把一个已经做好的web项目往里面一丢就行了。

除了webapps，我们还发现一个bin目录。一般来说，可执行的文件都放在bin目录下。

名称	修改日期	类型	大小	
backup	2016/10/11 19:21	文件夹		
bin	2016/9/30 17:36	文件夹		
conf	2016/9/30 17:36	文件夹		
lib	2016/9/30 17:36	文件夹		
logs	2016/10/12 8:32	文件夹		
temp	2016/10/12 22:30	文件夹		
webapp	2016/10/12 22:31	文件夹		
webapps	2016/9/30 17:36	文件夹		
work	2016/9/30 17:36	文件夹		
LICENSE	2016/2/8 22:28	文件	57 KB	
NOTICE	2016/2/8 22:28	文件	2 KB	
RELEASE-NOTES	2016/2/8 22:28	文件	9 KB	
RUNNING	2016/2/8 22:28	文本文档	17 KB	

打开bin，找到一个startup.bat文件。这就是启动tomcat的东西，双击它，tomcat就被启动了。


然后，浏览器可以访问tomcat里面的项目。

现在我们来手工搭建一个web项目，首先，在webapps目录下新建一个文件夹，是的，就是文件夹，不管你项目是什么，肯定还是放在文件夹里面的。

名称	修改日期	类型	大小	
docs	2016/9/30 17:36	文件夹		
examples	2016/9/30 17:36	文件夹		
host-manager	2016/9/30 17:36	文件夹		
manager	2016/9/30 17:36	文件夹		
myapp	2016/10/15 16:31	文件夹		
ROOT	2016/9/30 17:36	文件夹		

项目名称就叫做webapp。

打开webapp，根据web项目的规范，我们需要有一个WEB-INF文件夹。

名称	修改日期	类型	大小	
 WEB-INF	2016/10/15 16:39	文件夹		

然后，在WEB-INF文件夹里面，必须要有一个web.xml文件。

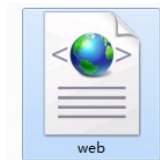
xml文件，就是一个描述性的文件，我现在的观点如下：

XML = JavaBean = Json = HashMap

它无非就是描述一些东西，保存一些数据而已。

好的，我们在里面新建一个web.xml。这个文件非常重要，正因为它的存在，tomcat容器才会知道这个文件夹里面竟然是一个web项目。

否则，tomcat容器是不知道这个web项目的，它只会将myapp文件夹看做是一个文件夹而已。



我们用记事本打开web.xml，将以下代码拷贝进去。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <display-name>web</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

XML的一个作用就是配置文件，web.xml本身就是一个配置文件。在web项目中，我们应用xml最多的也就是配置一些参数。

配置参数，就是给属性赋值嘛，没什么神秘的。

包括我们学习JavaSE，归根到底，一直在做的一件事就是new对象，然后调用方法，调用方法的目的一方面是做一些事情，另一方面不还是给属性赋值嘛。

你可以把web.xml看做是一个java类，类名叫做 webApp。它里面有两个属性，分别是display-name和welcome-file-list。

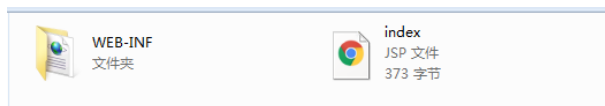
display-name是发布名称，也就是项目的名字。

welcome-file-list是欢迎页面，就是说，当你在浏览器直接访问这个myapp项目，默认跳转的页面。

想象一下，应该会变得非常好理解。

XML就是一个数据描述语言，我们通过web.xml描述这个项目的构成和配置。

好的，接下来，我们是不是要给他一个欢迎页啊。嗯，我们在webapp目录下添加一个简单的欢迎页，里面就打印一个HelloWorld。



```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>Hello World!</h1>
</body>
</html>
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
```

这是一条JSP的page指令，如果你用面向对象的思维来看待这个玩意，就是new了一个page对象，并且给它里面的language，contentType，charset，pageEncoding属性分别赋了值。

language表示JSP页面所用的语言，默认是java，其实你写不写都没有关系，因为目前来说JSP它只支持Java。

我们来试一下，现在我们把language属性去掉。就变成了这样：

```
<%@ page contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
```

contentType="text/html; charset=UTF-8": 设置页面的内容是文本或者html页面，字符设置为UTF-8。

pageEncoding="UTF-8": 页面编码设置为UTF-8。

好的，现在我们在bin目录，双击运行startup.bat

名称	修改日期	类型	大小
bootstrap.jar	2016/2/8 22:28	Executable Jar File	27 KB
catalina.bat	2016/2/8 22:28	Windows 批处理...	13 KB
catalina.sh	2016/2/8 22:28	Shell Script	21 KB
catalina-tasks.xml	2016/2/8 22:28	XML 文档	2 KB
commons-daemon.jar	2016/2/8 22:28	Executable Jar File	24 KB
commons-daemon-native.tar.gz	2016/2/8 22:28	360压缩	201 KB
configtest.bat	2016/2/8 22:28	Windows 批处理...	2 KB
configtest.sh	2016/2/8 22:28	Shell Script	2 KB
daemon.sh	2016/2/8 22:28	Shell Script	8 KB
digest.bat	2016/2/8 22:28	Windows 批处理...	3 KB
digest.sh	2016/2/8 22:28	Shell Script	2 KB
service.bat	2016/2/8 22:28	Windows 批处理...	7 KB
setclasspath.bat	2016/2/8 22:28	Windows 批处理...	4 KB
setclasspath.sh	2016/2/8 22:28	Shell Script	4 KB
shutdown.bat	2016/2/8 22:28	Windows 批处理...	2 KB
shutdown.sh	2016/2/8 22:28	Shell Script	2 KB
startup.bat	2016/2/8 22:28	Windows 批处理...	2 KB
startup.sh	2016/2/8 22:28	Shell Script	2 KB
tcnative-1.dll	2016/2/8 22:28	DLL 文件	1,122 KB
tomcat7.exe	2016/2/8 22:28	应用程序	85 KB
tomcat7w.exe	2016/2/8 22:28	应用程序	108 KB
tomcat-juli.jar	2016/2/8 22:28	Executable Jar File	38 KB
tomcat-native.tar.gz	2016/2/8 22:28	360压缩	380 KB
tool-wrapper.bat	2016/2/8 22:28	Windows 批处理...	4 KB

```

Tomcat
cat-7.0.68\webapps\manager has finished in 94 ms
十月 16, 2016 10:09:44 上午 org.apache.catalina.startup.HostConfig deployDirecto
ry
INFO: Deploying web application directory D:\software\work\tomcat\apache-toncat-
7.0.68\webapps\myapp
十月 16, 2016 10:09:44 上午 org.apache.catalina.startup.HostConfig deployDirecto
ry
INFO: Deployment of web application directory D:\software\work\tomcat\apache-ton
cat-7.0.68\webapps\myapp has finished in 93 ms
十月 16, 2016 10:09:44 上午 org.apache.catalina.startup.HostConfig deployDirecto
ry
INFO: Deploying web application directory D:\software\work\tomcat\apache-toncat-
7.0.68\webapps\ROOT
十月 16, 2016 10:09:44 上午 org.apache.catalina.startup.HostConfig deployDirecto
ry
INFO: Deployment of web application directory D:\software\work\tomcat\apache-ton
cat-7.0.68\webapps\ROOT has finished in 109 ms
十月 16, 2016 10:09:44 上午 org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-apr-8080"]
十月 16, 2016 10:09:45 上午 org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-apr-8009"]
十月 16, 2016 10:09:45 上午 org.apache.catalina.startup.Catalina start
INFO: Server startup in 2099 ms

```

启动完毕。

打开浏览器，在地址栏输入http://localhost:8080/myapp/

回车

# Hello World!

哇，是不是出来了。

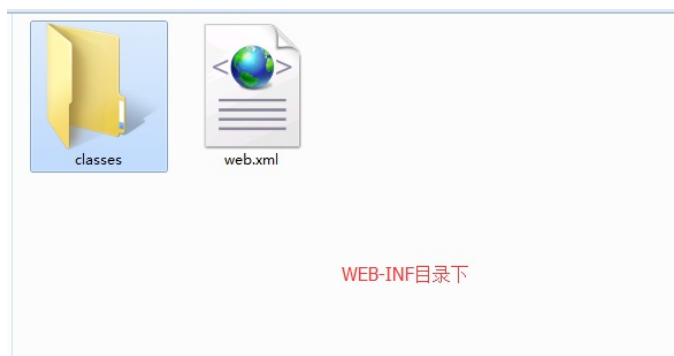
太棒了。

这就是手工搭建一个web项目的过程。

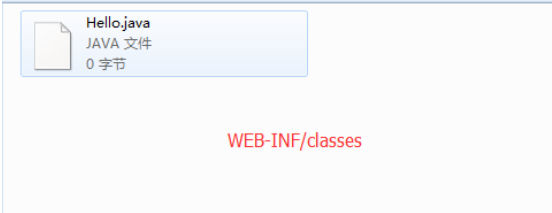
只要你符合web项目的规范，包括文件夹的名字，文件的名字，就会被tomcat容器识别为一个web项目。

接下来，我们来写服务器代码。

在WEB-INF下面新建一个文件夹，名字叫做classes，这个也是规范，就叫这个名字，否则tomcat容器识别不了。



里面在创建一个java文件，名字就叫Hello吧



用记事本打开，将下面的代码拷贝进去。

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Hello extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doPost(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        System.out.println("已经进入服务器...");
    }
}
```

这是一个比较简单的HttpServlet 程序，说到servlet，他的意思就是服务器小程序。

原来，在英文中，但凡是let结尾的单词，都有微小的意思。比如servlet，server是服务器，let结尾，那么就是服务器小程序。

servlet是Server Applet的缩写，我们再来看Applet，app是应用程序，又是let结尾，所以应该就是小的应用程序。

Applet

编辑

Applet是采用Java编程语言编写的小应用程序，该程序可以包含在 [HTML](#)（[标准通用标记语言](#)的一个应用）页中，与在页中包含图像的方式大致相同。<sup>[1]</sup>

中文名	小应用程序	编写语言	Java
外文名	Applet	适用范围	通常用于网页程序

这个就是所谓的英文词根，也是学英语的一个窍门。类似的例子还有很多，比如d开头的单词，大部分都有往下，分开，分散的意思。反正就是有一种往下，或者分发出去的韵味。

为了验证这个事情，我们打开有道词典，随便找几个看看。

distribution

纠错

英 [dɪstrɪˈbjʊːf(ə)n]

美 [ˈdɪstrəˈbjʊʃən]

n. 分布；分配

暂无图片 欢迎上传

想要图(124)

网络释义

专业释义

英英释义

一 分销

...产品与服务直接销售给消费者的销售行为，一般适合于大件的、销售周期短、销售点距生产企业较近的商品；而 **分销**（**Distribution**）是借助外部资源来完成商品的销售服务过程，一般适用于销售量较大、售后服务小、销售点与生产企业较远的商品。

基于1078个网页 [相关网页](#)

+ 分配

+ 配送

+ 分发

短语

**distribution center** 配送中心；配送中心；分发中心；物流中心

**frequency distribution** 频率分布；频数分布；次数分配；频次分布

**joint distribution** 共同配送；联合分布；配合配送；配送

**distribution**  
哦，对的，分配，配送，分发。这 不就是有分散，分出去的韵味吗？

**determine**

determine

纠错

英

[dɪˈtɜːmɪn]

美

[dɪˈtɜːmɪn]

v.

〈使〉下决心，〈使〉做出决定

vt.

决定，确定；判定，判决；限定

vi.

确定；决定；判决，终止；[主用于法律]了结，终止，结束

No

hurry

to

make

a

decision

Yes

129

84

网络释义

专业释义

英英释义

— 确定

我们虽然不能成为贵族的后代，但是透过我们的努力，我们可以成为贵族的祖先以**确定(Determine)**是否一个赌场是有信誉的，或不。幸福始终充满着缺陷。

基于667个网页-[相关网页](#)

+ 决定

+ 决心

+ 判定

确定，下决心。这不就是把什么什么东西**定下来**，定下来，不也是往下吗？

所以，得出一个结论，d开头的单词，而且，第二个字母必须是元音字母，比如de，di等。那么，这一类单词的情感色彩就有一种往下，或者分发出去的韵味。

再来说说servlet，let结尾都代表这个单词是一个小型的东西，比如

**piglet** 小猪  
**starlet** 小星星

再比如，psy开头的单词，情感色彩就是偏向于精神和心理学方面的，

psy

psy

psy**chology**

psy**chologist**

psy**chiatrist**

psy**cho**

**psychology** 心理学

还有

可达鸭

编辑

收藏

1250

14

可达鸭，日本任天堂公司发行的掌机游戏系列《**精灵宝可梦**》（国内常称“**口袋妖怪**”）中登场精灵的一种，首次出现于第一世代游戏《精灵宝可梦：红/绿》。

虽然头痛剧烈时就能使出不可思议的力量，不过当时的记忆却不会留下来。由于怎样也想不起来因此总歪着脖子。

中文名	可达鸭	体 重	19.6kg
外文名	コダック、Psyduck	生蛋分组	水中1组、陆上组
全国图鉴	054	孵化步数	5120步
属 性	水	性别比率	♂1：♀1
分 类	鸭子宝可梦	捕获度	190
特 性	潮湿/无天气	初始亲密度	70

可达鸭图册

中文名	可达鸭
外文名	コダック、Psyduck
全国图鉴	054
属 性	水
分 类	鸭子宝可梦

这下子应该比较好理解了吧，所以它为什么叫servlet，也是有原因的。

继续，我们用命令行的方式将java文件编译成class文件。

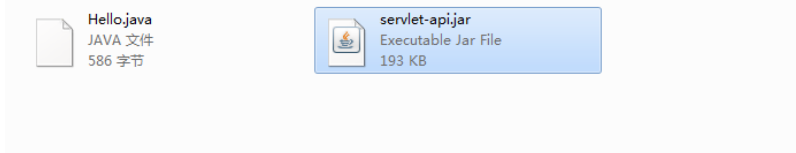
在编译之前，我们先去tomcat容器的lib目录找一个jar：



backup	2016/10/11 19:21	文件夹	
bin	2016/9/30 17:36	文件夹	
conf	2016/9/30 17:36	文件夹	
lib	2016/9/30 17:36	文件夹	
logs	2016/10/16 10:09	文件夹	
temp	2016/10/12 22:30	文件夹	
webapp	2016/10/12 22:31	文件夹	
webapps	2016/10/15 16:31	文件夹	
work	2016/9/30 17:36	文件夹	
apache-tomcat-7.0.68	2016/10/16 11:17	快捷方式	2 KB
LICENSE	2016/2/8 22:28	文件	57 KB
NOTICE	2016/2/8 22:28	文件	2 KB
RELEASE-NOTES	2016/2/8 22:28	文件	9 KB
RUNNING.txt	2016/2/8 22:28	文本文档	17 KB

名称	修改日期	类型	大小
annotations-api.jar	2016/2/8 22:28	Executable Jar File	16 KB
catalina.jar	2016/2/8 22:28	Executable Jar File	1,620 KB
catalina-ant.jar	2016/2/8 22:28	Executable Jar File	54 KB
catalina-ha.jar	2016/2/8 22:28	Executable Jar File	128 KB
catalina-tribes.jar	2016/2/8 22:28	Executable Jar File	257 KB
ecj-4.4.2.jar	2016/2/8 22:28	Executable Jar File	2,257 KB
el-api.jar	2016/2/8 22:28	Executable Jar File	55 KB
jasper.jar	2016/2/8 22:28	Executable Jar File	588 KB
jasper-el.jar	2016/2/8 22:28	Executable Jar File	122 KB
jsp-api.jar	2016/2/8 22:28	Executable Jar File	86 KB
servlet-api.jar	2016/2/8 22:28	Executable Jar File	194 KB
tomcat7-websocket.jar	2016/2/8 22:28	Executable Jar File	212 KB
tomcat-api.jar	2016/2/8 22:28	Executable Jar File	7 KB
tomcat-coyote.jar	2016/2/8 22:28	Executable Jar File	774 KB
tomcat-dbcp.jar	2016/2/8 22:28	Executable Jar File	229 KB
tomcat-i18n-es.jar	2016/2/8 22:28	Executable Jar File	71 KB
tomcat-i18n-fr.jar	2016/2/8 22:28	Executable Jar File	43 KB
tomcat-i18n-jar.jar	2016/2/8 22:28	Executable Jar File	46 KB
tomcat-jdbc.jar	2016/2/8 22:28	Executable Jar File	125 KB
tomcat-util.jar	2016/2/8 22:28	Executable Jar File	33 KB
websocket-api.jar	2016/2/8 22:28	Executable Jar File	36 KB

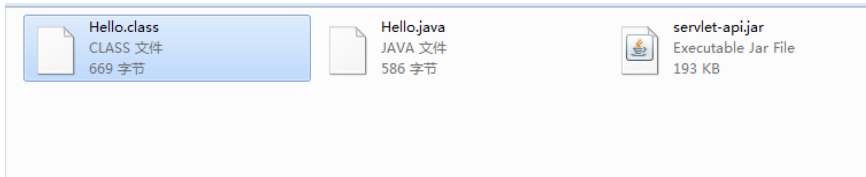
找到servlet-api.jar,复制一份，拷贝到classes目录下。



然后，我们在该classes目录下，按住shift，鼠标右键，选择在此处打开命令行窗口。

输入javac -classpath servlet-api.jar Hello.java

class文件就出来了



再次打开web.xml，我们还需要把这个servlet配上去，不然tomcat怎么知道这个servlet需要加入我们的web项目呢？

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <display-name>web</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

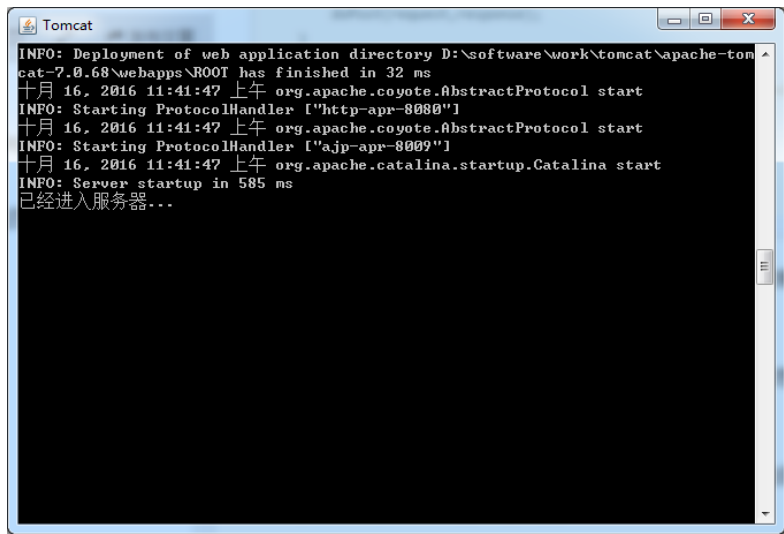
  <servlet>
    <servlet-name>Hello</servlet-name>
    <servlet-class>Hello</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Hello</servlet-name>
    <url-pattern>/hello</url-pattern>
  </servlet-mapping>
</web-app>
```

双击startup.bat，启动tomcat容器

然后，在浏览器的地址栏输入http://localhost:8080/myapp/hello  
发现网页上一片空白，回顾我们的servlet程序，按理说控制台会打印一句话的

```
public class Hello extends HttpServlet {  
  
    protected void doGet(HttpServletRequest request, HttpServletResponse response){  
        doPost(request,response);  
    }  
  
    protected void doPost(HttpServletRequest request, HttpServletResponse response){  
        System.out.println("已经进入服务器...");  
    }  
  
}
```

看看控制台：



果然如此。  
你应该也已经发现了，我们手工搭建web项目的话，是不是很麻烦呀？所以，这也是为什么我们现在都用eclipse，或者MyEclipse来开发项目了。  
这些IDE工具就是为了解决手工编译的麻烦而出现的。  
本文先介绍到这里，下一节重点来聊聊JSP。  
未完待续。。。