LightHouseIT

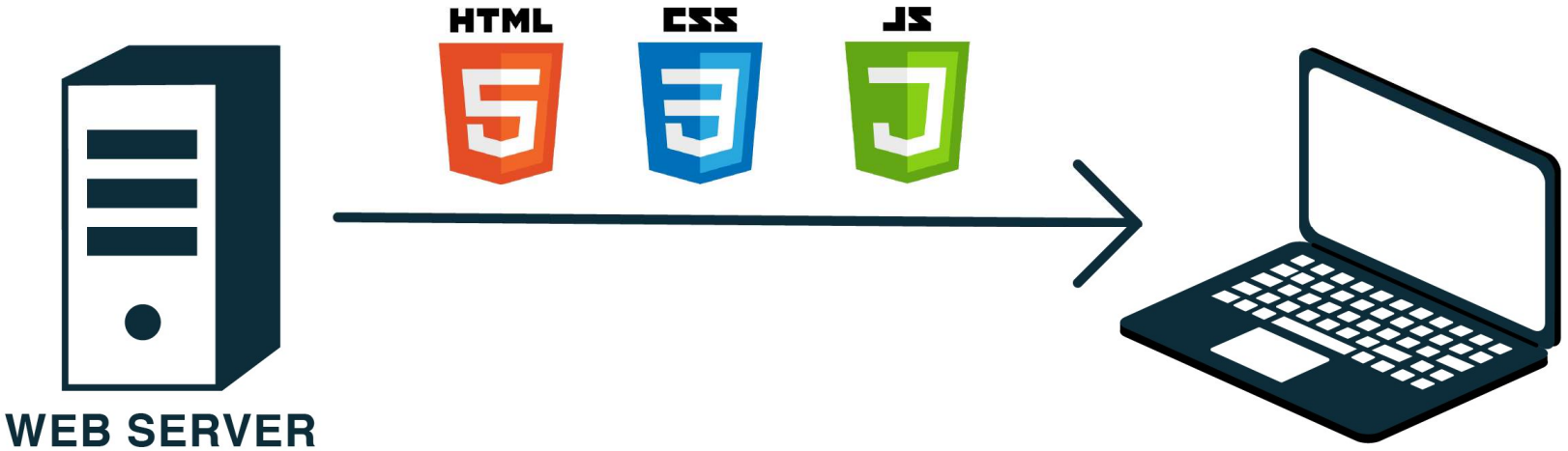
Java Web开发基础 Javaウェブ開発基本

- Web应用开发概述　ウェブアプリ開発概要
- Tomcat简介　Tomcat概要
- Servlet入门　Servlet入門

LightHouseIT

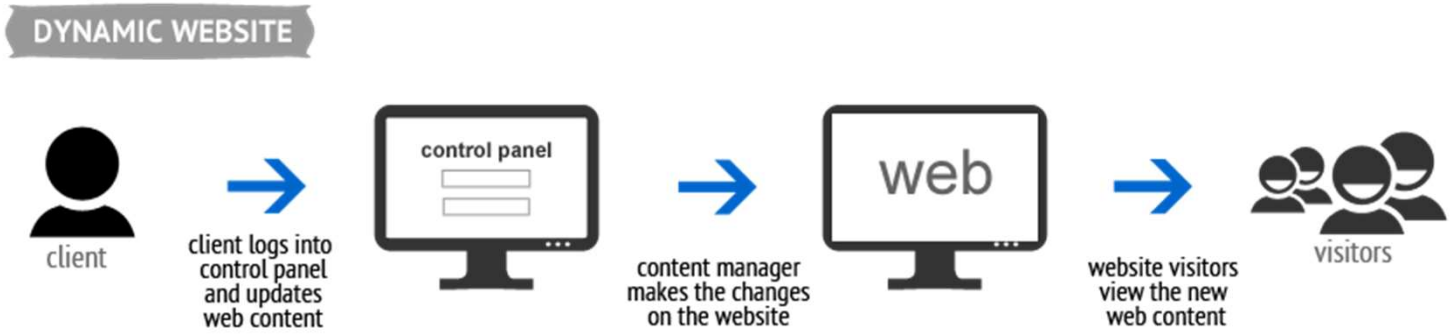
Web应用程序的工作原理

- Web应用程序大体上可以分为两种，即静态网站[静的ウェブページ]和动态网站[動的ウェブページ]。早期的Web应用主要是静态页面的浏览，即静态网站[静的ウェブページ]。这些网站使用HTML语言来编写，放在Web服务器上，用户使用浏览器通过HTTP协议请求服务器上的Web页面，服务器上的Web服务器将接收到的用户请求处理后，再发送给客户端浏览器，显示给用户。



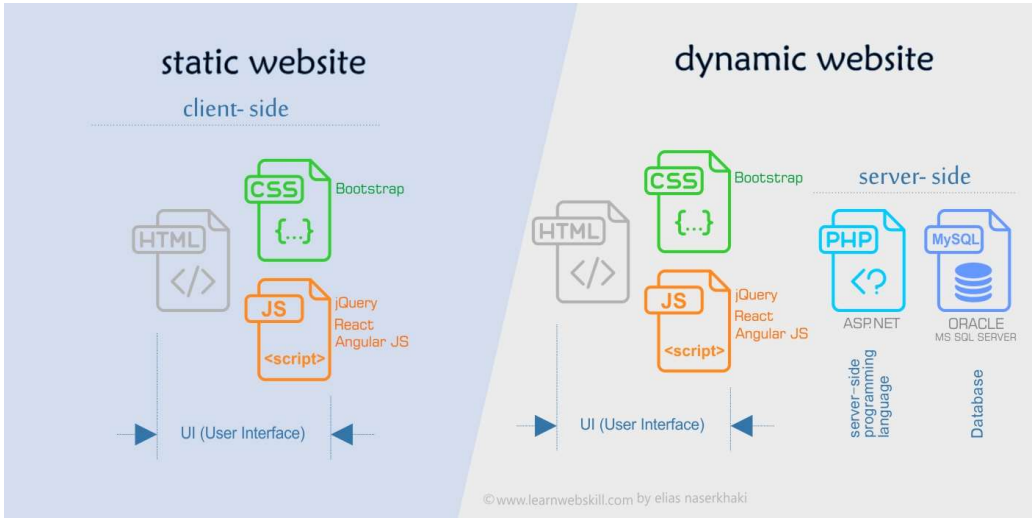
Web应用程序的工作原理

- 随着网络的发展，很多线下业务开始向网上发展，基于Internet的Web应用也变得越来越复杂，
- 用户所访问的资源已不再是只局限于服务器上保存的静态网页，更多的内容需要根据用户的请求动态生成页面信息，即动态网站[動的ウェブページ]。这些网站通常使用HTML语言和动态脚本语言(如JSP、ASP或是PHP等)编写，并将编写后的程序部署到Web服务器上，由Web服务器对动态脚本代码进行处理，并转化为浏览器可以解析的HTML代码，返回给客户端浏览器，显示给用户。



Web应用程序的工作原理

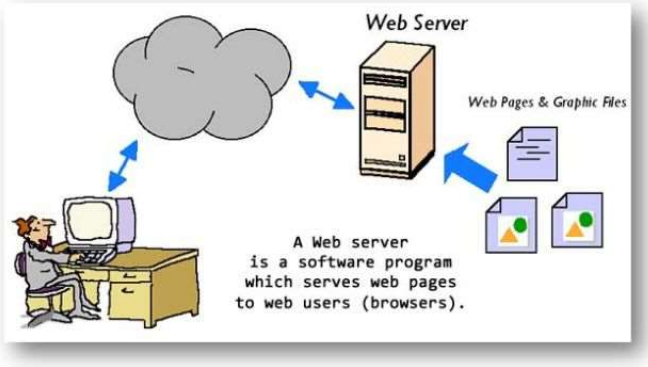
- 初学者经常会错误地认为带有动画效果的网页就是动态网页，其实不然。动态网页是指具有交互性、内容可以自动更新的网页，并且内容会根据访问的时间和访问者而改变。这里所说的交互性，是指网页可以根据用户的要求动态地改变或响应。
- 由此可见，静态网站[静的ウェブページ]类似于十几年前研制的手机，这种手机只能使用出厂时设置的功能和铃声，用户自己并不能对其铃声进行添加和删除等；而动态网站[動的ウェブページ]则类似于现在研制的手机，用户在使用这些手机时，不再是只能使用机器中默认的铃声，而是可以根据自己的喜好任意设置。



什么是网络服务器[ウェブサーバー]

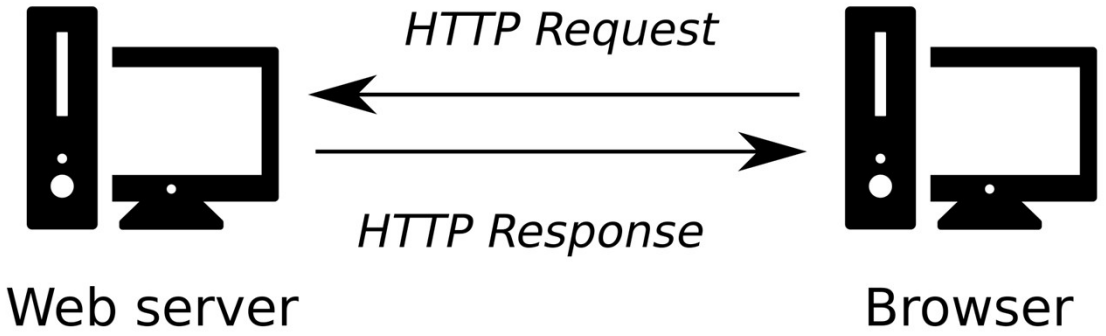
“网络服务器[ウェブサーバー]（Web server）”可以代指硬件或软件，或者是它们协同工作的整体。

- 硬件部分，一个网络服务器[ウェブサーバー]是一台存储了网络服务软件以及网站的组成文件（比如，HTML文档、图片、CSS样式表和JavaScript文件）的计算机。它接入到互联网并且支持与其他连接到互联网的设备进行物理数据的交互。
- 软件部分，网络服务器[ウェブサーバー]包括控制网络用户如何访问托管文件的几个部分，至少他要是一台HTTP服务器。一台HTTP服务器是一种能够理解URL（网络地址）和HTTP（浏览器用来查看网页的协议）的软件。通过服务器上存储的网站的域名（比如mozilla.org）可以访问这个服务器，并且他还可以将他的内容分发给最终用户的设备。



什么是网络服务器[ウェブサーバー]

- 基本上，当浏览器需要一个托管在网络服务器[ウェブサーバー]上的文件的时候，浏览器通过HTTP请求[HTTPリクエスト]这个文件。当这个请求到达正确的网络服务器[ウェブサーバー]（硬件）时，HTTP服务器（软件）收到这个请求，找到这个被请求的文档（如果这个文档不存在，那么将返回一个404响应），并把这个文档通过HTTP发送给浏览器。



什么是网络服务器[ウェブサーバー]

- 静态网络服务器（static web server），或者堆栈，由一个计算机（硬件）和一个 HTTP 服务器（软件）组成。我们称它为 “静态” 是因为这个服务器把它托管文件的 “保持原样” 地传送到你的浏览器。
- 动态网络服务器（dynamic web server） 由一个静态的网络服务器加上额外的软件组成，最普遍的是一个应用服务器和一个数据库[データベース]。我们称它为 “动态” 是因为这个应用服务器会在通过 HTTP 服务器把托管文件传送到你的浏览器之前会对这些托管文件进行更新。
- 举个例子，要生成你在浏览器中看到的最终网页，应用服务器或许会用一个数据库[データベース]中的内容填充一个 HTML 模板。网站像维基百科 [Wikipedia] 有成千上万的网页，但是它们不是真正的 HTML 文档，它们只是少数的 HTML 模板以及一个巨大的数据库[データベース]。这样的设置让它更快更简单地维护以及分发内容。

什么是网络服务器[ウェブサーバー]

- 一个网络服务器[ウェブサーバー]首先需要存储这个网站的文件，也就是说所有的 HTML 文档和它们的相关资源 （related assets），包括图片，CSS 样式表，JavaScript 文件，字形 （fonts） 以及影像。
- 严格来说，你可以在你自己的计算机上托管所有的这些文件，但是在一个专用的网络服务器[ウェブサーバー]上存储它们会方便得多，因为它
  - 会一直启动和运行
  - 会一直与互联网连接
  - 会一直拥有一样的 IP 地址（不是所有的 ISPs 都会为家庭线提供一个固定的 IP 地址）
  - 由一个第三方提供者维护
- 因为所有的这些原因，寻找一个优秀的托管提供者是建立你的网站的一个重要部分。比较不同公司提供的服务并选择一个适合你的需求和预算的服务（服务的价格从免费到每月上万美金不等）。
- 一旦你设置好一个网络托管解决方案，你只需要去上传你的文件到你的网络服务器[ウェブサーバー]

什么是HTTP

- HTTP (The HyperText Transfer Protocol，超文本传输协议) 是用于在 Web 上传输超媒体文件(如HTML)的底层协议，最典型场景的是在浏览器和服务器之间传递数据，以供人们浏览。现行的 HTTP 标准的版本是 HTTP/2。
- “http://” 称为 “schema”，是 URI 的组成部分，一般位于网络地址的开头。以 “https://google.com”为例，该地址说明请求文档时使用 HTTP 协议；这里的 https 代指 HTTP 协议的安全版本，即 SSL (en-US) （或称 TLS）
- HTTP 是基于文本的(所有的通信都以纯文本的形式进行) 以及无状态的 (当前通信状态不会发现以前的通信状态)，该特性极大方便了在www上浏览网页的人。

什么是HTTP

- 一个协议[プロトコル]（Protocol）是一套为了在两台计算机间交流而制定的规则。 HTTP 是一个文本化的 （textual），无状态的 （stateless）协议。
- 文本化
  - 所有的命令都是纯文本的 （plain-text） 和人类可读的 （human-readable）。
- 无状态
  - 无论是服务器还是客户都不会记住之前的交流。举个例子，仅依靠 HTTP，一个服务器不能记住你输入的密码或者你正处于业务中的哪一步。
- HTTP 为客户和服务器间的如何沟通提供清晰的规则。就目前而言，只需要知道这几点：
  - 只有用户可以制定 HTTP 请求，然后只会发送到服务器。服务器只能响应客户端的HTTP请求[HTTPリクエスト]。
  - 当通过 HTTP 请求一个文件时，客户必须提供这个文件的URL。
  - 网络服务器[ウェブサーバー]必须应答每一个 HTTP 请求，至少也要回复一个错误信息。

HTTP请求[HTTPリクエスト]（http request）

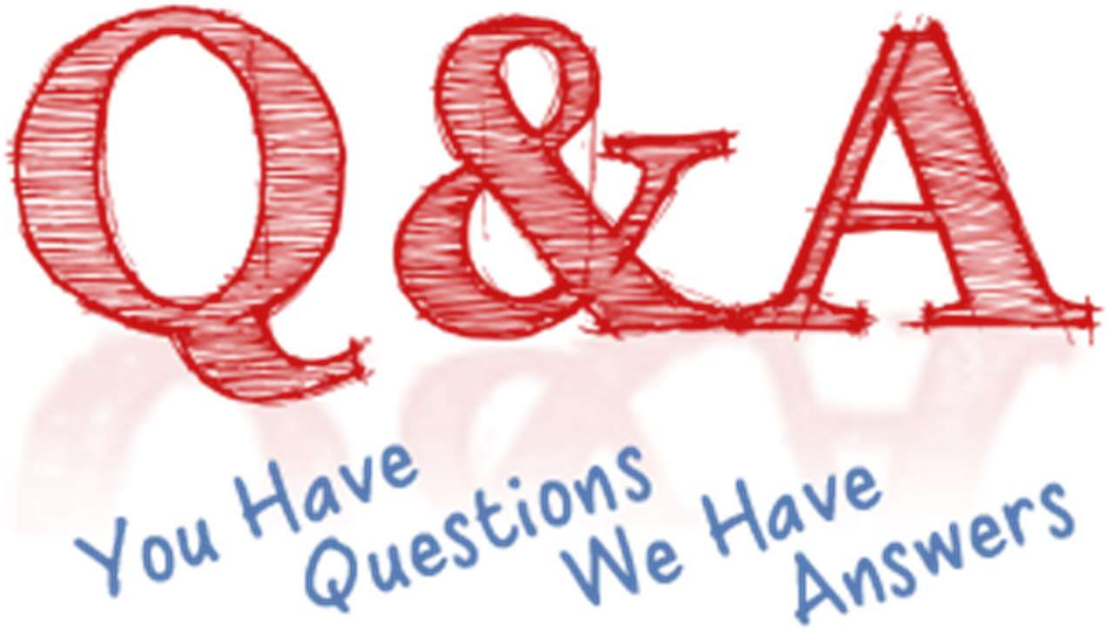
- 回忆一下之前HTML学过的form表单标签，这里的Submit就是一种HTTP请求[HTTPリクエスト]，比如下图是“填完姓名提交之后，请求页面跳转到/HelloWorld”

```
<form action="/HelloWorld">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
```

- 用户点击submit，填写的信息会被传到后端Control部（java代码），对用户的信息进行某些处理（比如判断是否为合法用户），然后针对用户的请求作出相应的回应（http response）这里的回应则是跳转到/HelloWorld界面。

Web应用技术

- 在开发Web应用程序时，通常需要应用客户端和服务器两方面的技术。其中，客户端应用的技术主要用于展现信息内容，而服务器端应用的技术则主要用于进行业务逻辑的处理和与数据库[データベース]的交互等。
- 客户端应用的技术
  - HTML
  - CSS
  - 客户端脚本技术：JavaScript
- 服务器端应用的技术
  - CGI：Common Gateway Interface，通用网关接口，最早用来创建动态网页的一种技术，它可以使浏览器与服务器之间产生互动关系。
  - PHP：PHP来自于Personal Home Page 一词，但现在的PHP已经不再表示名词的缩写，而是一种开发动态网页技术的名称。
  - JSP：JSP(Java Server Page)是以Java为基础开发的，所以它沿用Java强大的API功能。JSP页面中的HTML代码用来显示静态内容部分，嵌入到页面中的Java代码与JSP标记用来生成动态的内容部分。



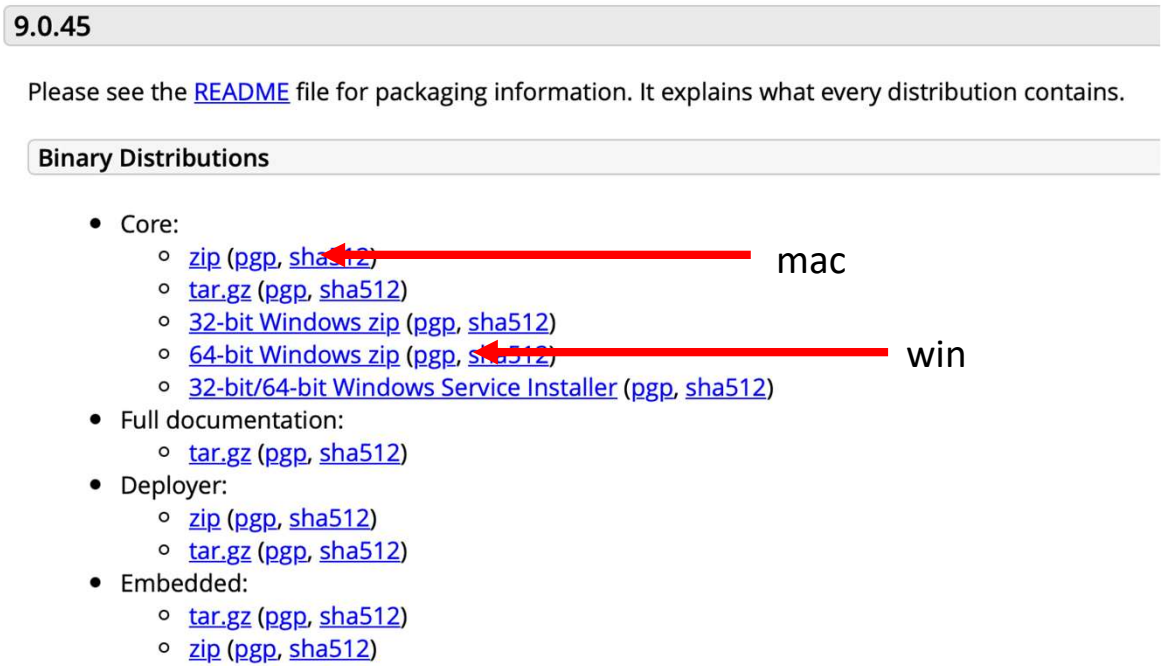
LightHouseIT

Tomcat简介

- Tomcat是一个Web服务器（同时也是Servlet容器），通过它我们可以很方便地接收和返回到请求（如果不用Tomcat，那我们需要自己写Socket来接收和返回请求）。
- Tomcat其实我们并不需要学太多的知识，只要学会安装和启动以及了解一下各个目录的含义就差不多了。

Tomcat下载安装

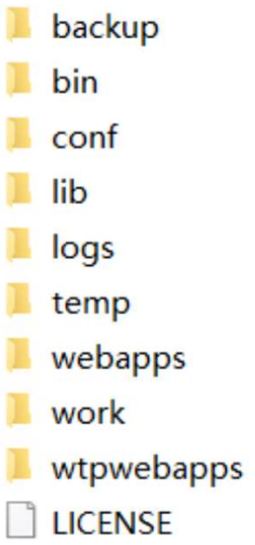
- 我们统一使用tomcat 9, 已下载的同学不必重复下载
- 下载链接： https://tomcat.apache.org/download-90.cgi



- 下载之后解压，请务必记住解压包的地址，之后会用到。

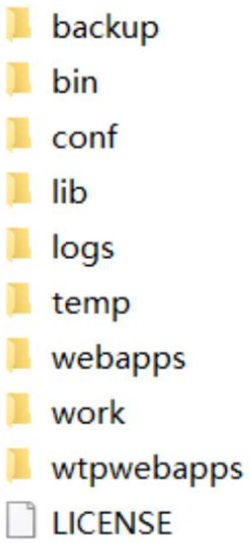
Tomcat目录层次结构

- bin：该目录下存放的是二进制可执行文件
- conf：这个目录下有四个最为重要的文件：
  - server.xml：配置整个服务器信息。例如修改端口号，设置编码，添加虚拟主机等。
  - tomcat-users.xml：存储tomcat用户的文件，这里保存的是tomcat的用户名及密码，以及用户的角色信息。可以按着该文件中的注释信息添加tomcat用户，然后就可以在Tomcat主页中进入Tomcat Manager页面。
  - web.xml：部署描述符文件，这个文件中注册了很多MIME类型，即文档类型。这些MIME类型是客户端与服务器之间说明文档类型的，如用户请求一个html网页，那么服务器还会告诉客户端浏览器响应的文档是text/html类型。客户端浏览器通过这个MIME类型就知道如何处理它了。
  - context.xml：对所有应用的统一配置，通常我们不会去配置它。



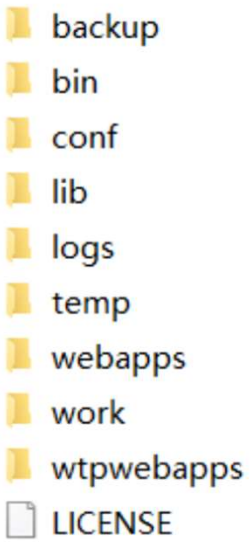
Tomcat目录层次结构

- lib：Tomcat的类库里面是一大堆jar文件。如果需要添加Tomcat依赖的jar文件，可以把它放到这个目录中，当然也可以把应用依赖的jar文件放到这个目录中，这个目录中的jar可以被所有项目共享，但这样你的应用放到其他Tomcat下时就不能再共享这个目录下的Jar包了，所以建议只把Tomcat需要的Jar包放到这个目录下。
- logs：这个目录中都是日志文件，记录了Tomcat启动和关闭的信息，如果启动Tomcat时有错误，那么异常也会记录在日志文件中。
- temp：存放Tomcat的临时文件，这个目录下的东西可以在停止Tomcat后删除。
- webapps：存放web项目的目录，其中每个文件夹都是一个项目。如果这个目录下已经存在了目录，那么都是tomcat自带的项目。其中ROOT是一个特殊的项目，在地址栏中没有给出项目目录时，对应的就是ROOT项目。 http://localhost:8080/examples 进入示例项目。其中examples就是项目名，即文件夹的名字。



Tomcat目录层次结构

- work：运行时生成的文件，最终运行的文件都在这里。它是通过webapps中的项目生成的。可以把这个目录下的内容删除，再次运行时会生再次生成work目录。当客户端用户访问一个JSP文件时，Tomcat会通过JSP生成Java文件，然后再编译Java文件生成class文件，生成的java和class文件都会存放到这个目录下。logs：这个目录中都是日志文件，记录了Tomcat启动和关闭的信息，如果启动Tomcat时有错误，那么异常也会记录在日志文件中。
- LICENSE：许可证。



LightHouseIT

Java Web应用程序

- 一个 Java Web 应用程序是由一组 Servlet，JSP页面，Java类，以及其它的资源组成的运行在 web 服务器（Tomcat）上的完整的应用程序，以一种结构化的有层次的目录形式存在。

什么是Servlet？

- Servlet 是 SUN 推出的一套规范，规定了如何用 Java 来开发动态网站[動的ウェブページ]。也就是说，Java 可以用来开发网站后台，但是要遵循一定的标准。
- Servlet 可以使用所有的 Java API，类库丰富，功能强大。
- 通过Servlet，你可以：
  - 接收用户通过 <form> 表单提交的信息；
  - 查询数据库[データベース]，包括用户信息、文章内容、页面点击次数等；
  - 生成验证码，防止机器恶意注册。

什么是Servlet？

- 例如，要在网页上显示IP地址，那它的HTML源码是：

```
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>演示页面</title>
</head>
<body>
    <p>你的IP地址是：127.0.0.1</p>
</body>
</html>
```

什么是Servlet？

- 那么服务器上的Java代码就应该这样写：

```java
// 导入必需的类
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

// 继承 HttpServlet 类
public class HelloWorld extends HttpServlet {
    public void init() throws ServletException{
        // TODO
    }

    public void doGet(
        HttpServletRequest request,
        HttpServletResponse response
    )throws ServletException, IOException{
        // 设置报头类型
        response.setContentType("text/html");
        // 必须通过println()输出HTML代码
        PrintWriter out = response.getWriter();
        out.println('<!DOCTYPE html>');
        out.println('<html lang="zh">');
        out.println('<head>');
        out.println('<meta charset="UTF-8">');
        out.println('<title>演示页面</title>');
        out.println('</head>');
        out.println('<body>');
        out.println('<p>你的IP地址是: ');
        out.println(request.getRemoteAddr());
        out.println('</p>');
        out.println('</body>');
        out.println('</html>');
    }

    public void destroy(){
        // TODO
    }
}
```

- 用户接收到的HTML代码，都是通过 println() 语句输出的。

什么是Servlet？

- 以上就是古老的 CGI 程序，需要把 HTML 代码当做字符串，通过输出语句一条一条的输出。互联网初期，CGI 程序大行其道，为互联网的发展做出了不可磨灭的贡献。

什么是JSP？

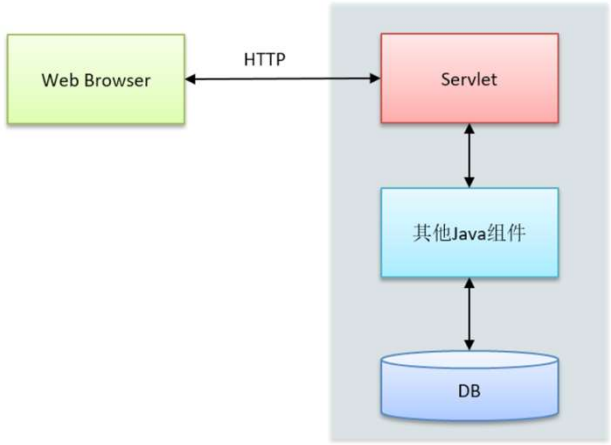
- JSP是为了简化Servlet的工作出现的替代品，Servlet输出HTML非常困难，JSP就是替代Servlet输出HTML的。
- JSP全名为Java Server Pages，java服务器页面。JSP是一种基于文本的程序，其特点就是HTML和Java代码共同存在。

为什么要学Servlet？

- 它是Java Web编程技术的根本。
- Servlet放在现在算是一个古老的技术了，现在的项目一般来说还是以SpringMVC / SpringBoot居多
- 但是现在Java的网络框架的底层都离不开Servlet，SpringMVC的核心用的就是Servlet。

Servlet简介

- Servlet是用来处理客户端请求的动态资源，也就是当我们在浏览器中键入一个地址回车跳转后，请求就会被发送到对应的Servlet上进行处理。
- 功能：
  - 接收请求数据：我们都知道客户端请求会被封装成HttpServletRequest对象，里面包含了请求头、参数等各种信息。
  - 处理请求：通常我们会在service、doPost或者doGet方法进行接收参数，并且调用业务层（service）的方法来处理请求。
  - 完成响应：处理完请求后，我们一般会转发（forward）或者重定向（redirect）到某个页面。

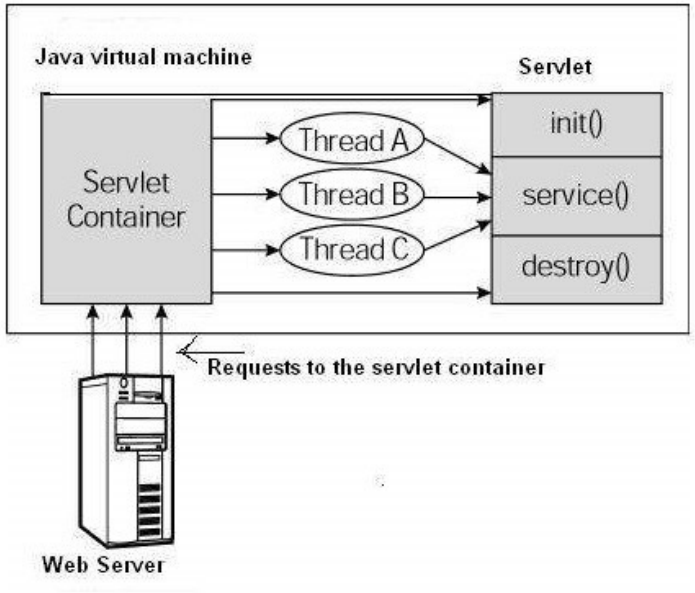


Servlet 包

- Java Servlet 是运行在带有支持 Java Servlet 规范的解释器的 web 服务器上的 Java 类。
- Servlet 可以使用 javax.servlet 和 javax.servlet.http 包创建，它是 Java 企业版的标准组成部分，Java 企业版（JavaEE）是支持大型开发项目的 Java 类库的扩展版本。
- Java Servlet 就像任何其他的 Java 类一样已经被创建和编译。在你安装 Servlet 包并把它们添加到您的计算机上的 Classpath 类路径中之后，就可以通过 JDK 的 Java 编译器或任何其他编译器来编译 Servlet。

Servlet 生命周期

- Servlet通过调用init()方法进行初始化。
- Servlet调用service()方法来处理客户端的请求。
- Servlet通过调用destroy()方法终止(结束)。
- Servlet最后是由JVM的垃圾回收器进行垃圾回收的。



Servlet 生命周期

- init（）：
  - 在Servlet的生命周期中，仅执行一次init()方法。它是在服务器装入Servlet时执行的，负责初始化Servlet对象。可以配置服务器，以在启动服务器或客户机首次访问Servlet时装入Servlet。无论有多少客户机访问Servlet，都不会重复执行init（）。Servlet调用service()方法来处理客户端的请求。
- service（）：
  - 它是Servlet的核心，负责响应客户的请求。每当一个客户请求一个HttpServlet对象，该对象的Service()方法就要调用，而且传递给这个方法一个“请求”（ServletRequest）对象和一个“响应”（ServletResponse）对象作为参数。在HttpServlet中已存在Service()方法。默认的服务功能是调用与HTTP请求[HTTPリクエスト]的方法相应的do功能。
- destroy（）：
  - 仅执行一次，在服务器端停止且卸载Servlet时执行该方法。当Servlet对象退出生命周期时，负责释放占用的资源。一个Servlet在运行service()方法时可能会产生其他的线程，因此需要确认在调用destroy()方法时，这些线程已经终止或完成。

Servlet 生命周期步骤

1. Web Client 向Servlet容器（Tomcat）发出HTTP请求[HTTPリクエスト]
2. Servlet容器接收Web Client的请求
3. Servlet容器创建一个HttpRequest对象，将Web Client请求的信息封装到这个对象中。
4. Servlet容器创建一个HttpResponse对象
5. Servlet容器调用HttpServlet对象的service方法，把HttpRequest对象与HttpResponse对象作为参数传给HttpServlet 对象。
6. HttpServlet调用HttpRequest对象的有关方法，获取HTTP请求[HTTPリクエスト]信息。
7. HttpServlet调用HttpResponse对象的有关方法，生成响应数据。
8. Servlet容器把HttpServlet的响应结果传给Web Client。

Servlet实例

- Servlet 是服务 HTTP 请求并实现 javax.servlet.Servlet 接口的 Java 类。 Web 开发程序员通常通过继承javax.servlet.http.HttpServlet来编写自己的servlet，处理HTTP请求[HTTPリクエスト]。

```java
// 导入必需的 java 库
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

// 扩展 HttpServlet 类
public class HelloWorld extends HttpServlet {
    private String message;

    public void init() throws ServletException {
        // 执行必需的初始化
        message = "Hello World";
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
        // 设置响应内容类型
        response.setContentType("text/html");

        // 实际的逻辑是在这里
        PrintWriter out = response.getWriter();
        out.println("<h1>" + message + "</h1>");
    }

    public void destroy() {
        // 什么也不做
    }
}
```

继承HttpServlet类

重写init方法
进行网页初始化

重写doGet方法
处理HTTP请求[HTTPリクエスト]

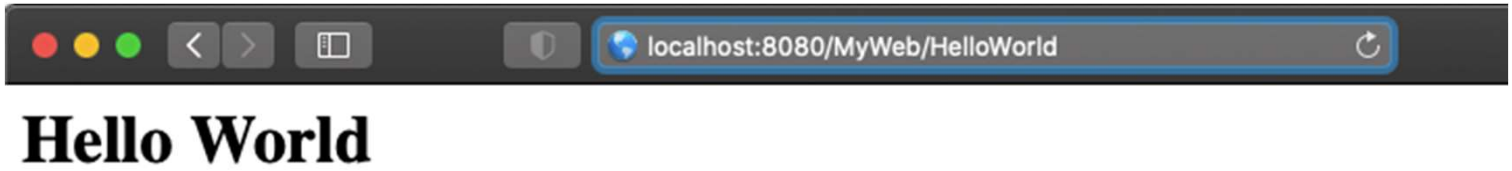
重写destory方法

Servlet开发Web应用基本步骤

- 创建Servlrt类
- 标明url路径：配置web.xml 或者 在类文件里加注释 @WebServlet("/xx")
- 发布到Tomcat服务器
- 启动Tomcat
- 打开浏览器，输入网址访问servlet：
  http://服务器ip:端口号/appName/Servlet提供的url

创建Web HelloWorld

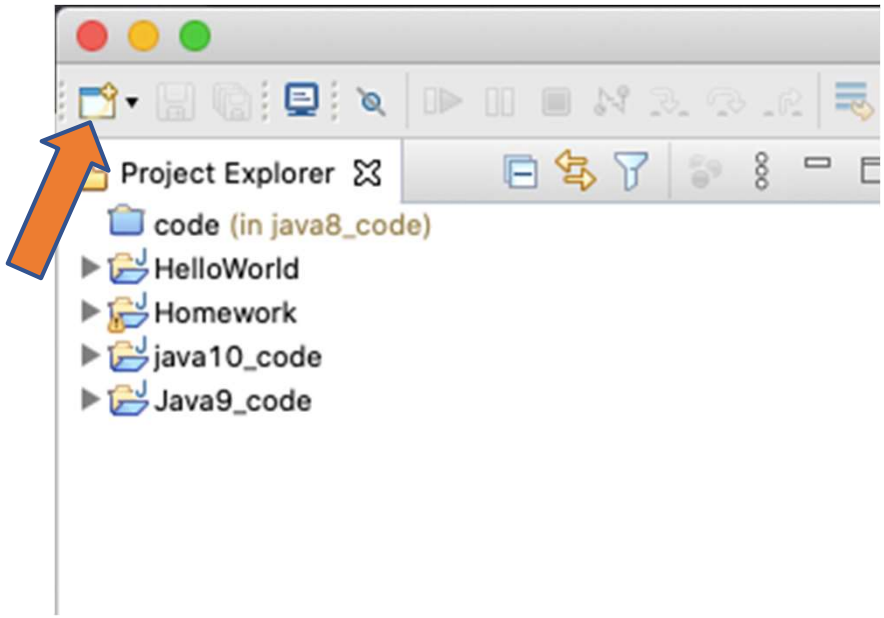
- 需要:
  - Eclipse EE
  - 刚才下好的 Tomcat
  - 课件code文件夹里的servlet api和tomcat api

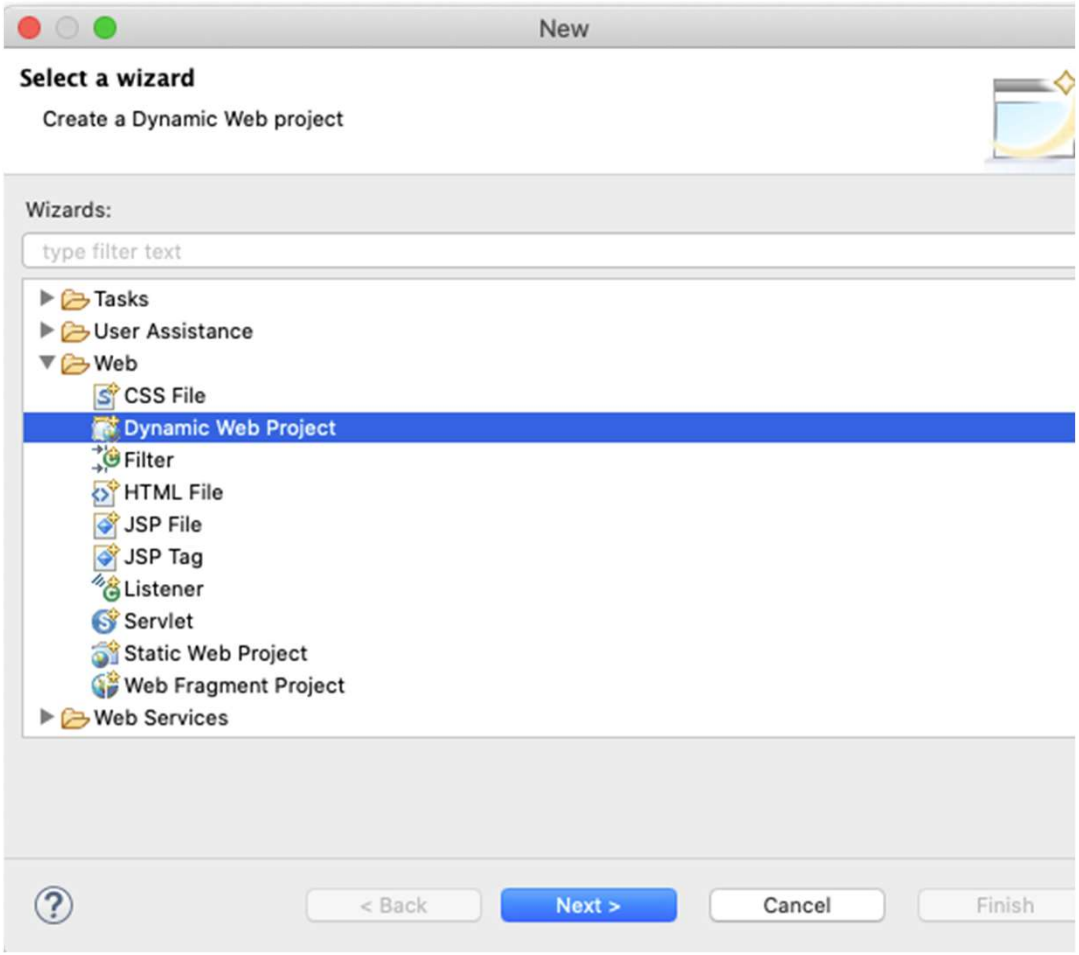


效果图

步骤一 创建项目

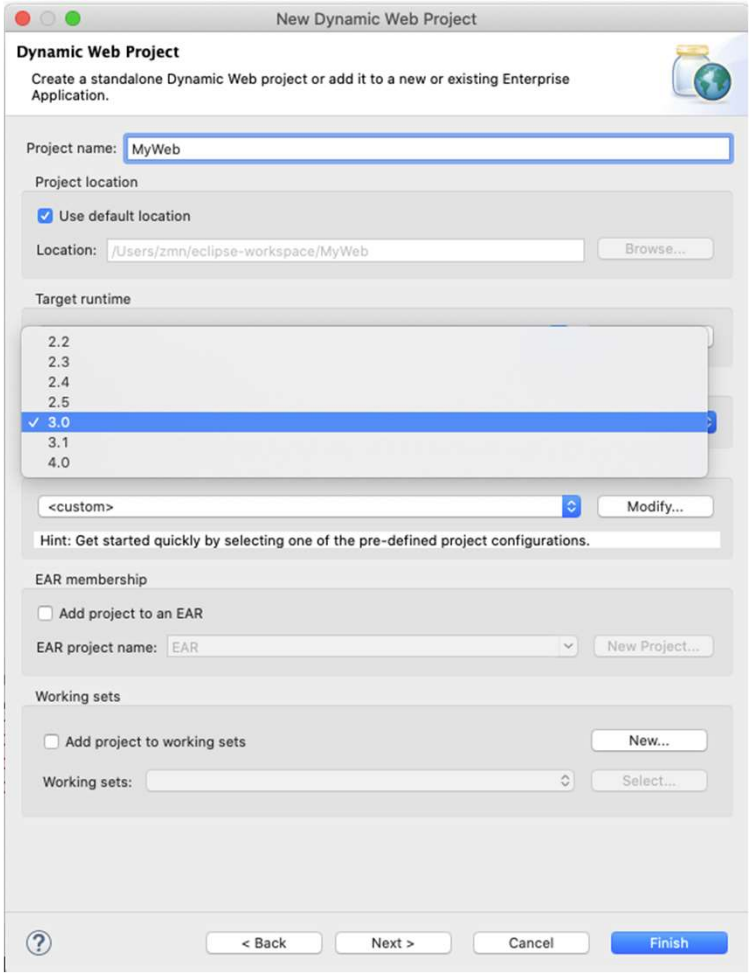
- 打开Eclipse，找到Dynamic Web Project，点击Next





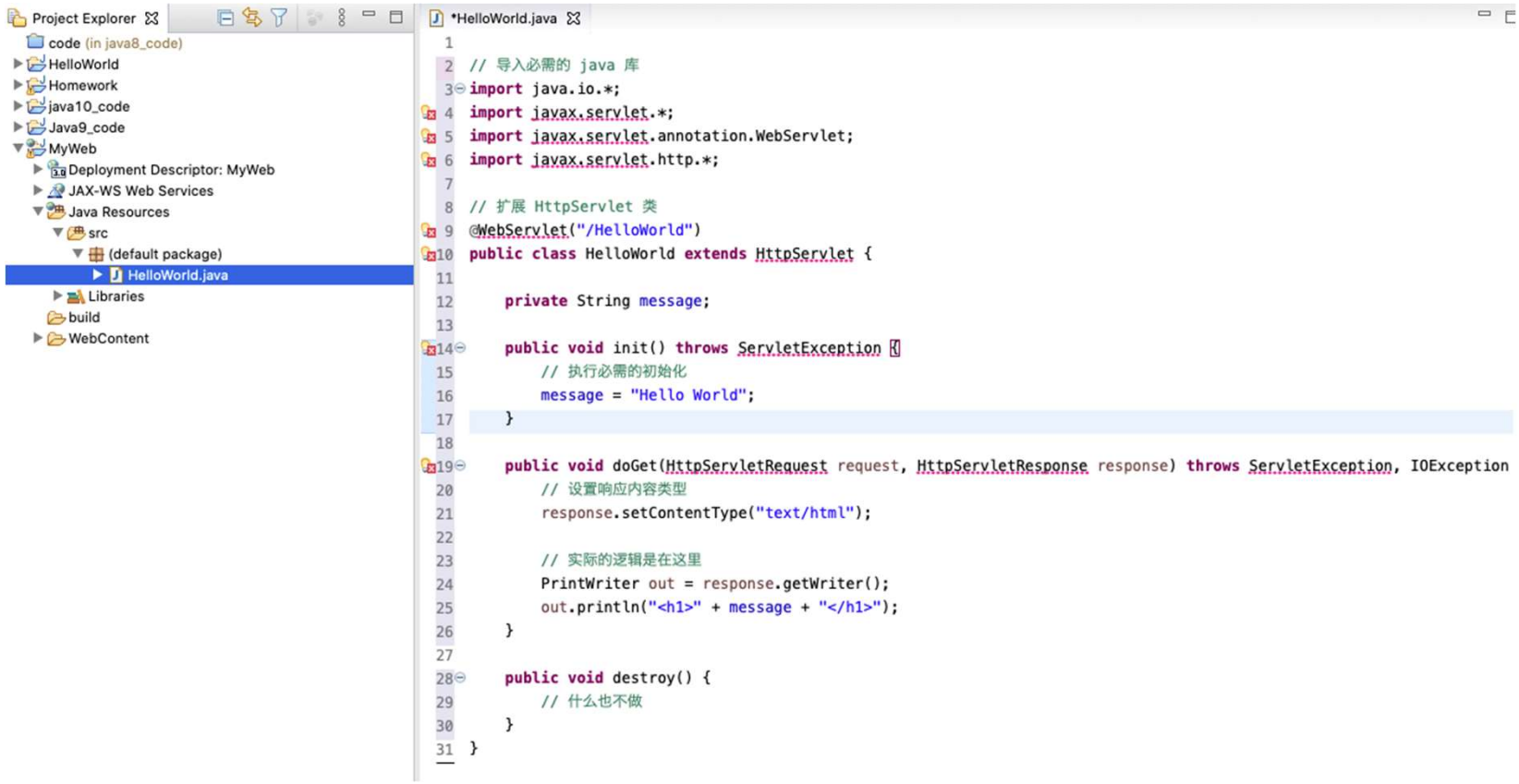
步骤一 创建项目

- 项目名为MyWeb，要手动点选一下Dynamic web module version 3.0，Finish



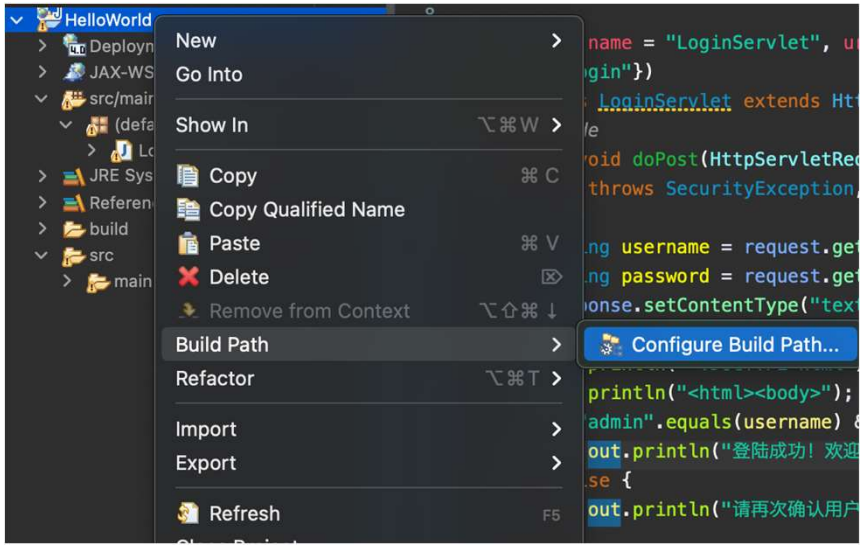
步骤二 创建Servlet文件

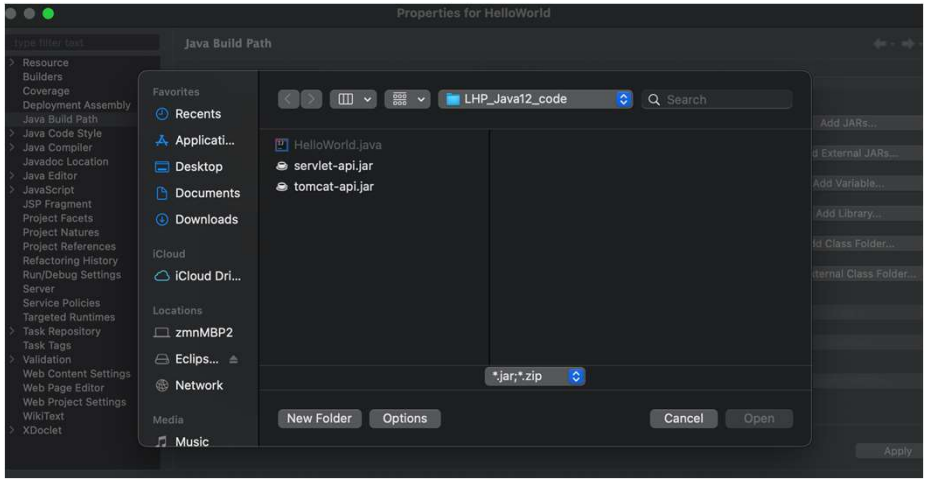
- 索引到MyWeb > Java Resources > src
- 创建HelloWorld.java，将课件的HelloWorld.java的代码复制过来



步骤三 导入jar包

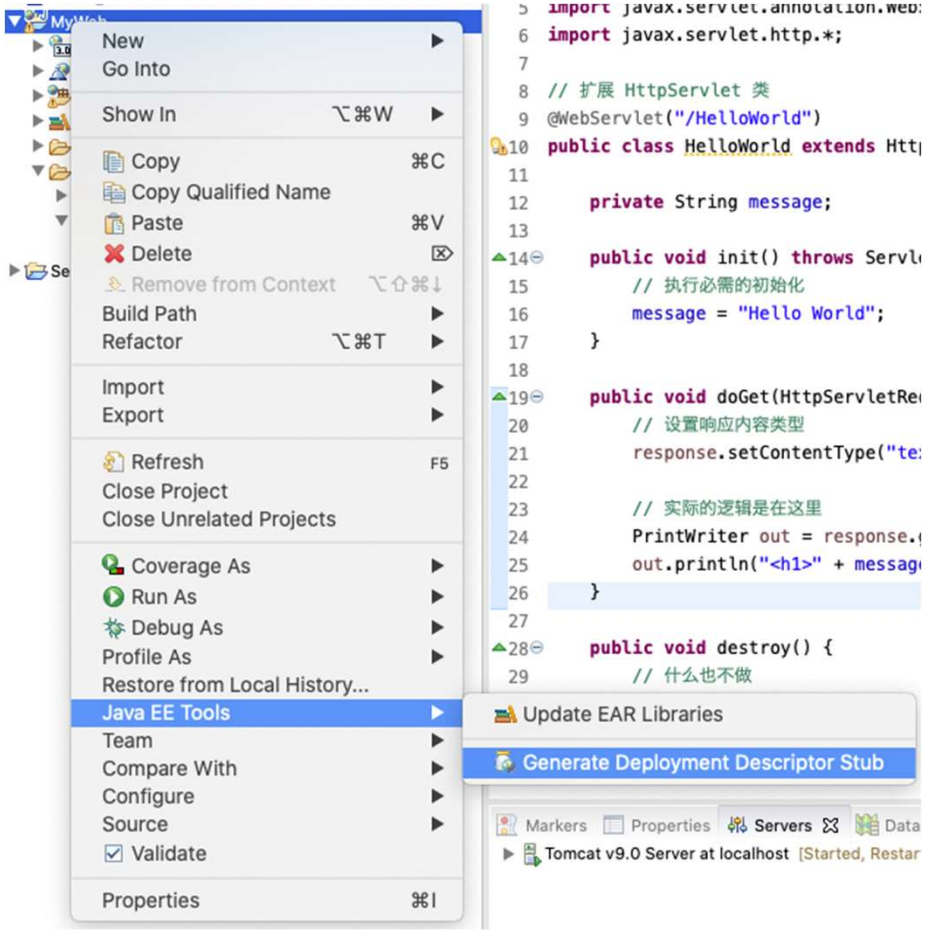
- 代码报错是因为还没有导入jar包
- MyWeb > Build Path > Configure Build Path
- 选择Libraries，点击Add External JARs
- 按shift选择课件代码中的jar包都添加进来
- Open，Apply and Close

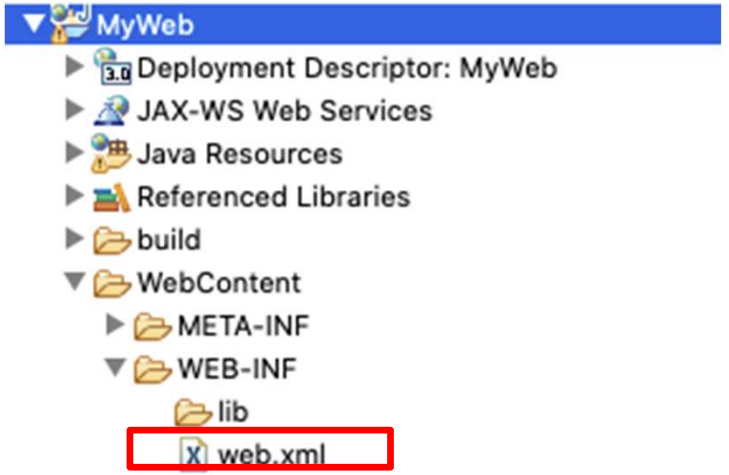




步骤四：添加web.xml文件

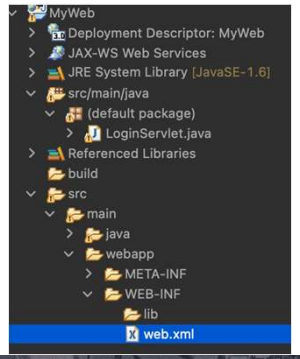
- 该文件是项目配置文件，就是用来声明哪个servlet对应哪个网址
- 添加方法：右键项目，选择Java EE Tools > Generate





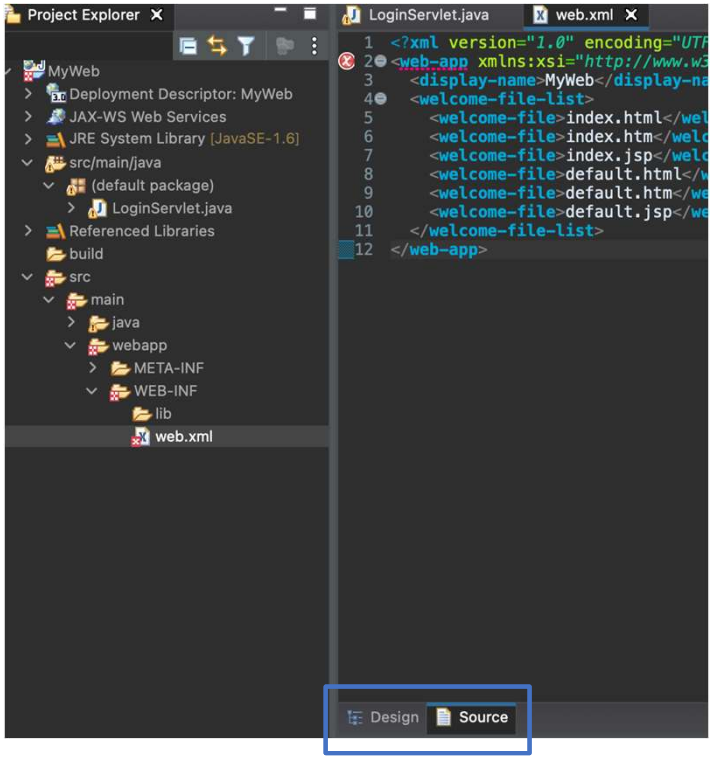
这里多出一个web.xml文件

或者是这样的



步骤四：添加web.xml文件

- web.xml文件可以呈现Design和Source两种模式，选择Source模式



步骤四：添加web.xml文件

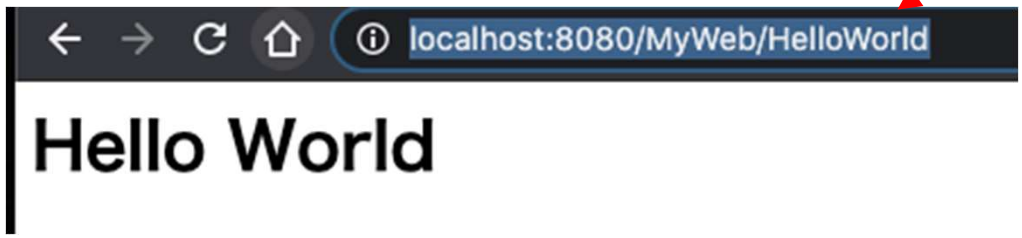
- 在web-app开始结束标签中添加以下标签并保存

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XML
  <display-name>MyWeb</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>HelloWorld</servlet-name>
    <servlet-class>HelloWorld</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>HelloWorld</servlet-name>
    <url-pattern>/HelloWorld</url-pattern>
  </servlet-mapping>
</web-app>
```

欢迎页面，可无视

servlet类名

name一致才能匹配上



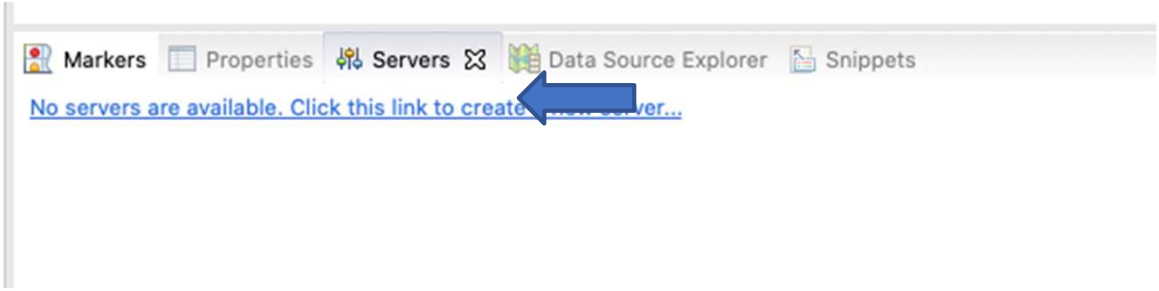
步骤四：添加web.xml文件

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://JAVA.sun.com/xml/ns
```

如果配置文件代码报错，可以尝试把这里的java改成大写再保存

步骤五：建立服务器

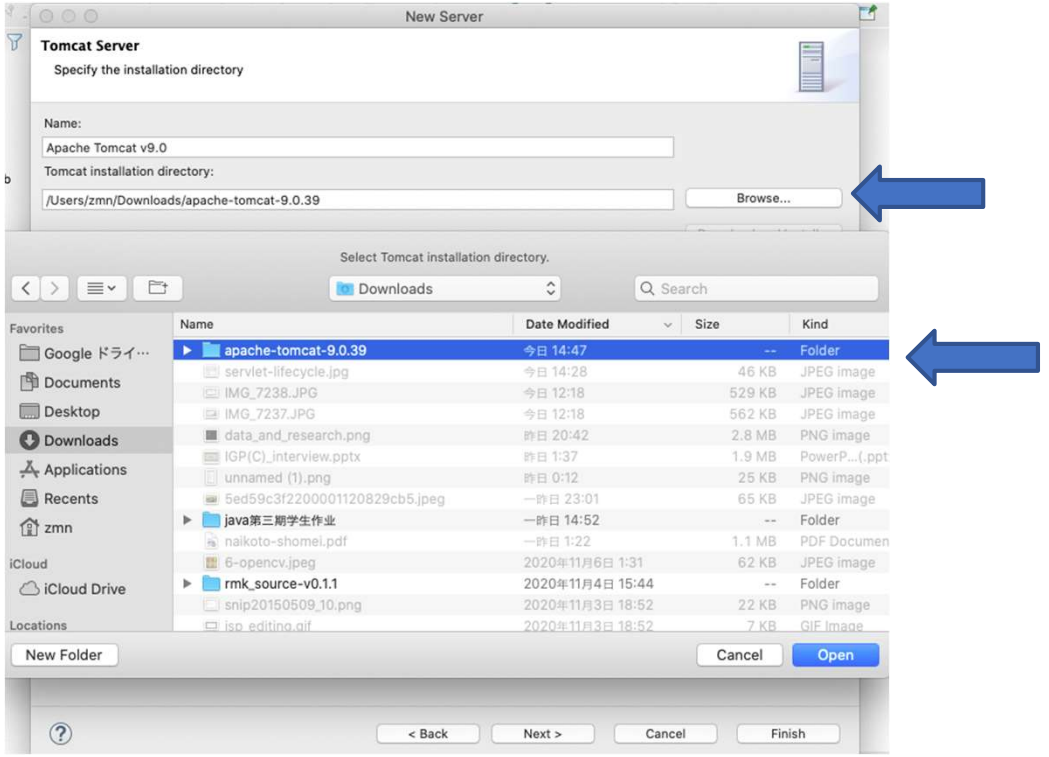
- Eclipse下面区域选择Servers，创建新server



- 选择Apache Tomcat v9.0, 点击Next

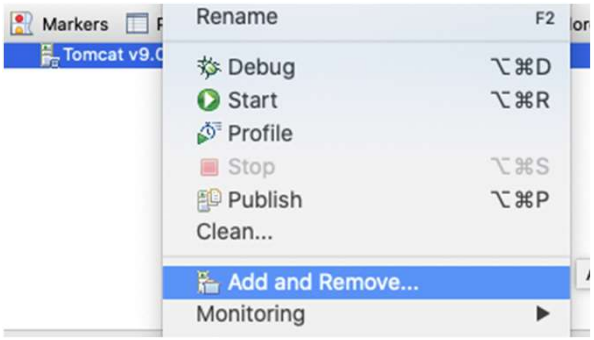
步骤五：建立服务器

- 因为是用Eclipse第一次创建服务器，需要把Tomcat的包传给Eclipse
- 点击Browse，找到刚下载好的Tomcat，点击Open，点击Finish

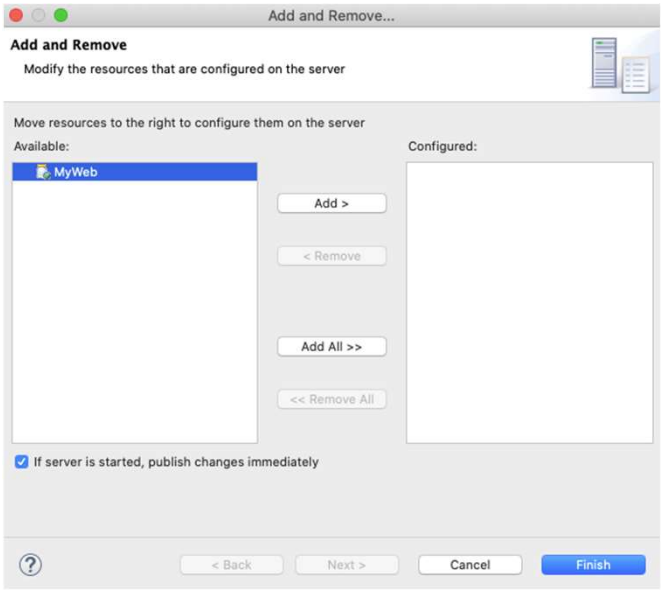


步骤六：将MyWeb添加到服务器

- 右键创建好的server，选择Add and Remove

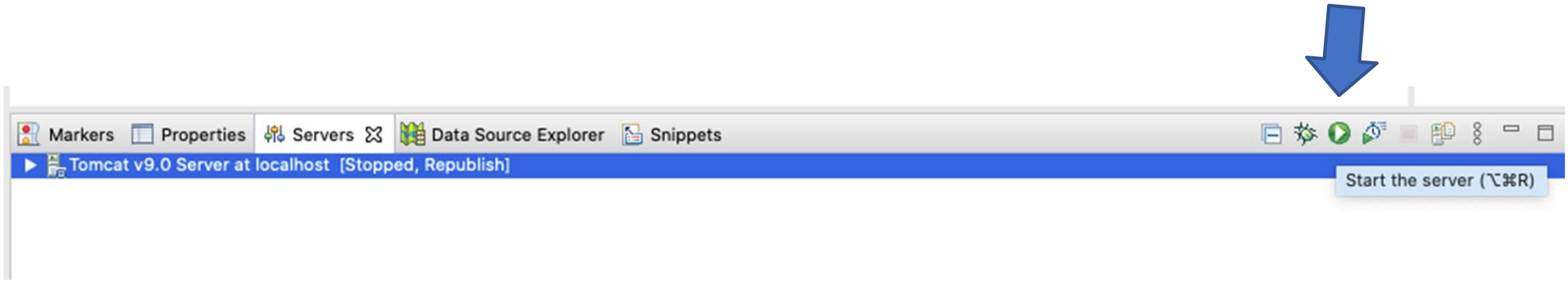


- 选择MyWeb，点击Add >， Finish

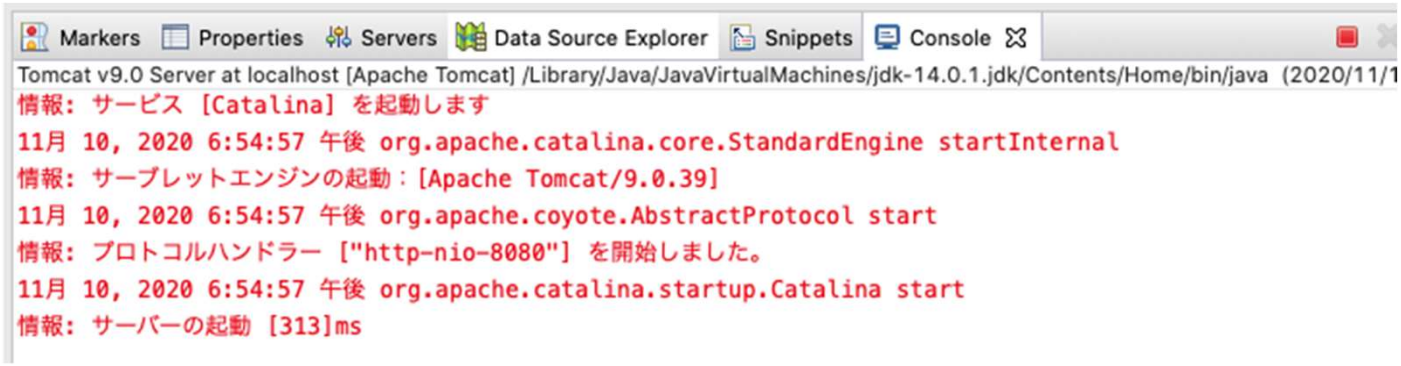


步骤七 启动服务器

- 选择下方服务器，点击绿色按钮



- 然后Console会显示服务器状态

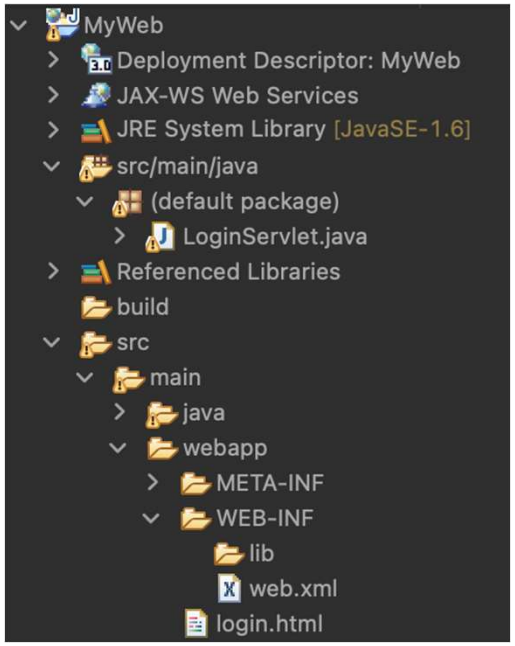


浏览器查看网页

- http://localhost:8080/MyWeb/HelloWorld

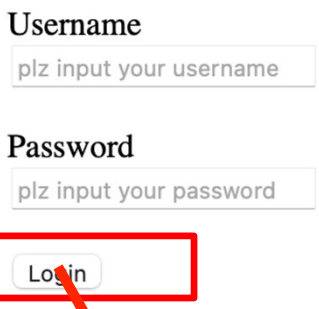
基于Servlet的动态登录网站

- 仍然使用MyWeb项目
- 在webapp或webContent下创建一个login.html
- 将课件代码中的login.html中的代码复制进去，保存
- 索引到src，创建LoginServlet.java，将课件的LoginServlet.java的代码复制过来



页面加载及跳转流程

访问
http://localhost:8080/MyWeb/login.html



http://localhost:8080/MyWeb/LoginServlet

请再次确认用户名和密码的正确性。-_-