第一讲 JavaWeb 开发入门

一、理解 B/S 架构

1. B/S 架构的技术原理

① B/S 架构

先来破除大多数人的一个误解，在很多同学眼里，windows 操作系统是软件，平时进行文本编辑的 word，office 是软件，使用的 qq 是软件，手机里面装的微信等 app 也是软件，那么当我们用浏览器打开京东网站，请问，这个京东网站是软件吗？有些同学就开始迟疑了，其实大家不必迟疑它就是一个软件程序，只不过它和我们的 office 相比，是另外一种形态的程序。

以 office，qq 为代表的软件程序，我们称为 C/S 架构。

C:代表的是客户端 client 的意思，S：Server 指的是服务器。

要使用 office 和 qq 时，先要在本地安装 office 和 qq。同时，office 和 qq 每年都会有版本升级，如果想用最新版的程序还必须手动更新，当然客户端功能升级了，服务器端的功能自然也要升级，只不过这个升级工作就不需要用户去操作了。

接下来解释一下 B/S 程序

S 和上面一样也是 Server 服务器的意思，这里的 B 指的是浏览器，我们在京东上挑选商品，下单，付款整个流程都是在京东网站上完成的，所以京东网站就是支持我们线上购物的软件系统，和 C/S 相比，我们使用京东商城不用安装客户端，只需要有个浏览器就可以了，而且服务器端升级，浏览器端不需要升级，比如京东以前是没有 PLUS 会员功能的，现在有了，但是我们不需要在本地做任何操作，还是可以去访问 PLUS 会员页面的。

总结一下二者区别

	B/S 架构	C/S 架构
软件安装	浏览器	需要专门的客户端应用
升级维护	客户端零维护	客户端需要单独维护和升级
平台相关	与操作系统平台的关系最小化	对客户端操作系统一般有限制

性能安全	在响应速度和安全性上需要花费更多设计成本	能充分发挥客户端处理能力，客户端响应快

在接下来的课程中，各位同学要学习的就是一个基于 B/S 架构的购物商城软件。

② 服务器

在这里，有必要统一一下大家对于服务器的理解，从硬件角度来讲，服务器和我们通常使用的台式机，笔记本没有本质上的区别，只不过性能上和稳定性上要比一般电脑要好些。因此一台电脑之所以被叫做服务器主要还是在软件上，首先就是操作系统，我们无论是 pc 机，还是苹果机，用的都是家用系统 windows ， mac os，而服务器上的操作系统绝大部分是 Linux，少数地方用 windows 的 Server 版本，也就是服务器版本的 windows 系统，还有就是更加古老一些的 unix 系统。

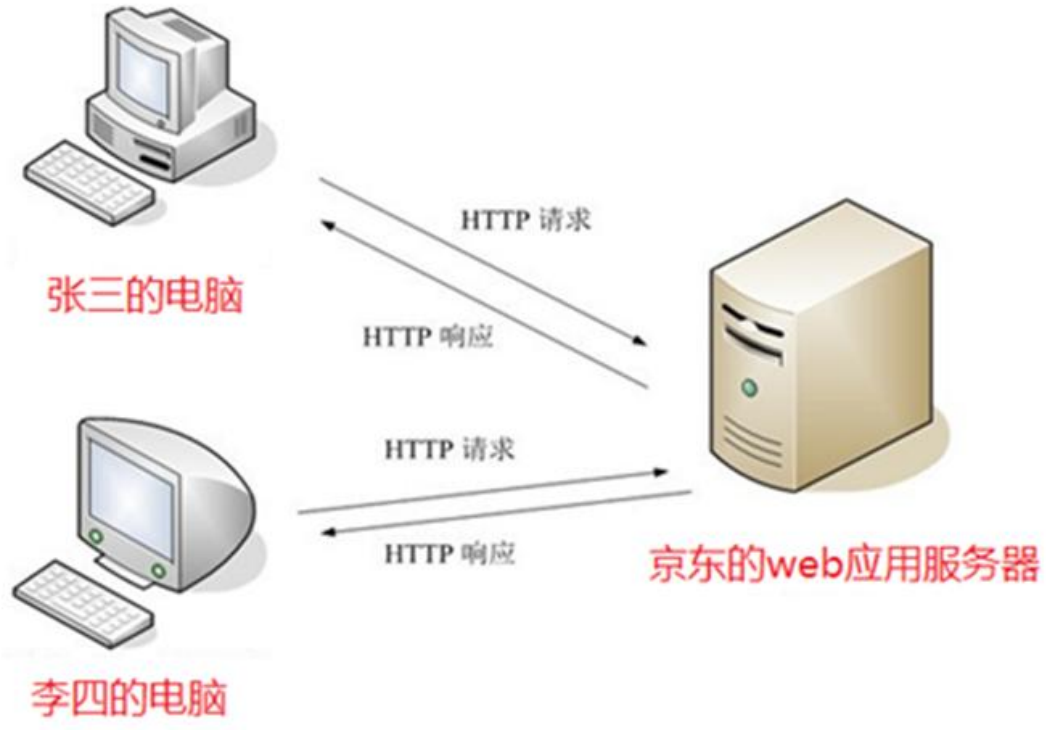
第二，服务器上安装的软件也不是常用的办公娱乐软件，什么 qq 了等等。假如一台服务器上安装了 mysql 数据库，那这台服务器就称它为数据库服务器，因为它提供数据管理服务，如果有一台服务器能够运行购物商城上的程序，对外提供网页访问服务的，我们称之为 web 服务器。所以，在下一 PART 里就会给大家介绍一个 web 服务软件：tomcat，它不但能对外提供网页，还能运行 java 程序。

2. URL

URL 全名叫统一资源定位符，用于定位网络上用户需要访问的资源。比如通过这个 url
https://img13.360buyimg.com/da/jfs/t15031/162/1989680528/30718/32290d07/5a6498e9N5ef4cde4.gif
就能访问到京东的 logo
在这个 url 中 https 是协议名称，关于 http 协议的详细内容，下面会说到
5a6498e9N5ef4cde4.gif 是文件名
da/jfs/t15031/162/1989680528/30718/32290d07 是服务器上存储 gif 文件的路径
img13.360buyimg.com 是域名，

3. HTTP 协议

http 协议全称是超文本传输协议，这个协议就是为了让我们能够在浏览器和服务器之间传输数据用的，它由两部分组成，HTTP 请求和 HTTP 响应。



看图，当我们在浏览器上输入网址访问京东网站的时候，浏览器会将输入的内容封装成一个 Http 请求，发送到服务器上，服务器接收到这个请求后会将返回给浏览器的内容封装成一个 Http 响应，然后返回给浏览器。

简单点讲，输入网址，或者点击按钮，链接等操作就是向服务器端发送请求，用户在浏览器上看到内容更新，是因为服务器返回了一个响应到浏览器端。

注意，先有请求，再有响应，不存在没有请求的响应。

为了让大家能更直观的看到请求和响应，在这里把请求和响应的内容抓取出来给大家看下

```
▼ Request Headers      view parsed
  GET / HTTP/1.1
  Host: www.baidu.com
  Connection: keep-alive
  User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537
  Upgrade-Insecure-Requests: 1
  Accept: text/html,application/xhtml+xml,application/xml;
  Accept-Encoding: gzip, deflate, br
  Accept-Language: zh-CN,zh;q=0.9
  Cookie: BIDUPSID=38A28B3FF727739C502A8A00980F7860; PSTM=
```

先来看下请求的内容：

在请求头部的第一行，首先显示的是请求方式

然后是请求资源的路径

接下来是使用的 http 协议的版本

其它的内容我们先放下以后用到再说
接下来看下响应的内容：

在响应里面第一行，同样有协议的版本，200 代表的是响应的状态，200 表示正常响应
我们网页的内容是放在最后的响应体里面

从这里我们可以得出一个结论所谓的请求和响应其实就是在浏览器和服务器之间传递的有固定格式的字符串。

https 协议，在京东，淘宝，百度上我们看到的是 http 的加密版本 https，二者在本质上是一样的。

二、Tomcat 安装与配置

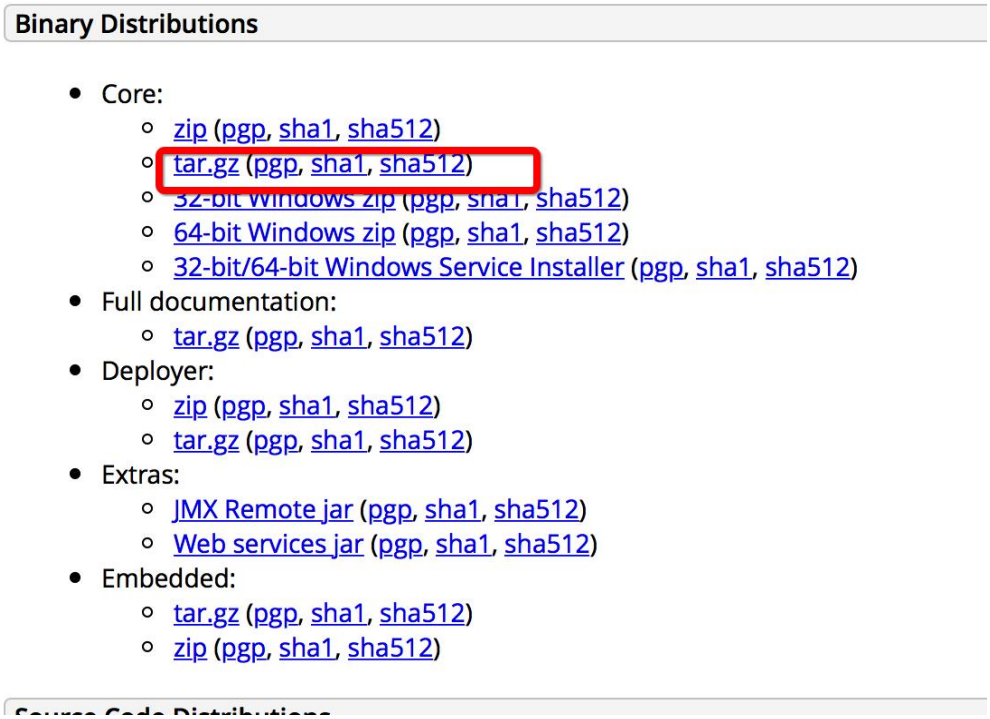
上小节曾讲到 B/S 架构的软件是由浏览器向服务器端发送请求，服务器对这个请求做出响应，返回请求处理后的结果给浏览器。
站在服务器端的角度上来说，肯定得有一个软件能够接收请求，并发送响应，而 tomcat 就是可以完成这个过程的软件。

接下来，就来学习下 tomcat 的安装和基本使用

1. Tomcat 的安装和配置

① MAC 版本

- 首先到官网下载 Tomcat：https://tomcat.apache.org/download-90.cgi



- 解压 tomcat 文件,最好把它文件名重命名为“Tomcat”，方便以后查找，最后把它放入 /Library(资源库中)
  - 点击 finder-->用户-->你电脑的名字-->资源库(有的也叫/Library)。
  - 有些苹果将 library 目录隐藏起来了，可以直接点左上角前往，前往文件夹，路径填 /资源库，就进入了资源库了。
- 用终端（Terminal）直接打开 Tomcat 了
  - 进入 Tomcat 的 bin 目录下：终端输入 cd /Library/Tomcat/bin ， 输完回车

    也可以打开 Tomcat 文件夹，把 bin 文件夹直接拖拉到终端，当然前提是先输入 cd+空格
  - 授权 bin 目录下的所有操作：终端输入 sudo chmod 755 *.sh，输完回车

    这时要输入密码，输完回车
  - 这时候就可以开启 Tomcat 了，终端输入 sudo sh ./startup.sh，输完回车

② Windows 版本

首先打开 tomcat 的官方主页　tomcat.apache.org
在页面的左侧 Download 下方的 tomcat8，在右侧的页面中选择合适的版本
比如 64 位的 windows 版本，注意不要选 installer 版本，这个版本需要安装，使用起来比较麻烦

下载解压缩版本，只需要解压就可以使用

解压完毕后，找到 tomcat 解压目录下的 bin 这个子目录，在这个子目录中有一个 startup.bat 文件，双击运行
这里会显示一个命令行窗口，输出运行的一些信息
然后打开浏览器，输入 localhost:8080
我们就能打开 tomcat 的首页

在这里需要注意的是，如果你机器上的环境变量配置的有问题的话，可能会出现闪退现象，原因是 tomcat 软件启动时, 会默认到系统的环境变量中查找一个名称叫 JAVA_HOME 的变量。所以出现闪退现象的同学，检查一下你机器上的环境变量
添加一个 JAVA_HOME 的变量　　JAVA_HOME= C:\Program Files\Java\jdk1.6.0_30
注意别配置到 bin 目录里

③ Tomcat 端口冲突

系统是靠端口号区分接收到的网络数据交给谁去处理
假设这台机器上有两个 tomcat
那么当用户请求到达服务器后，交给谁去处理呢，如果这两个 tomcat 都是从 8080 端口取数据，那么必然造成系统混乱
因此一个端口上，只能被一个程序使用，在这里我们可以做这样一个试验，我们再打开一个 tomcat
很显然第二个 tomcat 运行时报错了，注意这个报错信息，8080 端口被占用，以后遇到类似错误的时候，大家应该知道这是怎么一回事情了

```
Caused by: java.net.BindException: Address already in use
    at sun.nio.ch.Net.bind0(Native Method)
    at sun.nio.ch.Net.bind(Net.java:433)
    at sun.nio.ch.Net.bind(Net.java:425)
```

2. Tomcat 的目录结构

思考一个问题，通过 localhost:8080 访问的页面，放到哪里去了？如果能找到这个位置，是不是就可以认为，我们自己的网页也可以部署到 tomcat 上去了。

所以接下来，我们逐个认识一下 tomcat 中的目录

目录	说明
/bin	存放各种平台下用于启动和停止 Tomcat 的脚本文件
/conf	存放 Tomcat 服务器的各种配置文件
/lib	存放 Tomcat 服务器所需的各种 JAR 文件
/logs	存放 Tomcat 的日志文件
/temp	Tomcat 运行时用于存放临时文件
/webapps	当发布 Web 应用时，默认情况下会将 Web 应用的文件存放于此目录中
/work	Tomcat 把由 JSP 生成的 Servlet 放于此目录下

第一个 bin 目录，我们已经用过了，就是放置 tomcat 运行脚本命令的目录

第二个 conf 目录，配置文件的目录，里面有核心配置文件 server.xml，打开 server.xml 文件，可以搜索到 8080 端口，也就是说，如果我们要改 tomcat 的端口号，直接在这里改就可以了，注意改完以后要重新启动 tomcat，在这里我就不演示了，有兴趣的同学可以自己试试看

第三个是 lib，里面放着 tomcat 和 web 项目中需要使用的 jar 包

logs，放置的是一些日志文件

temp：存放一些临时文件

webapps 目录：这个就是默认情况下 WEB 项目存放的目录，进入 webapps 目录，会看到有个 root 文件夹，打开 root 文件夹，会发现 localhost:8080 页面上的资源都在这里，

因此如果想把自己的页面部署到 tomcat 上去，只需要在 webapps 中，新建一个文件夹 myweb，将自己的页面资源复制进去，然后打开浏览器，输入 localhost:8080/myweb/index.html 就可以看到购物街的首页了，注意这里的网址中要加上我们刚才新建的文件夹名称，之前的 root 文件夹可以不写在 url 中，是因为 root 是 tomcat 默认的网站路径

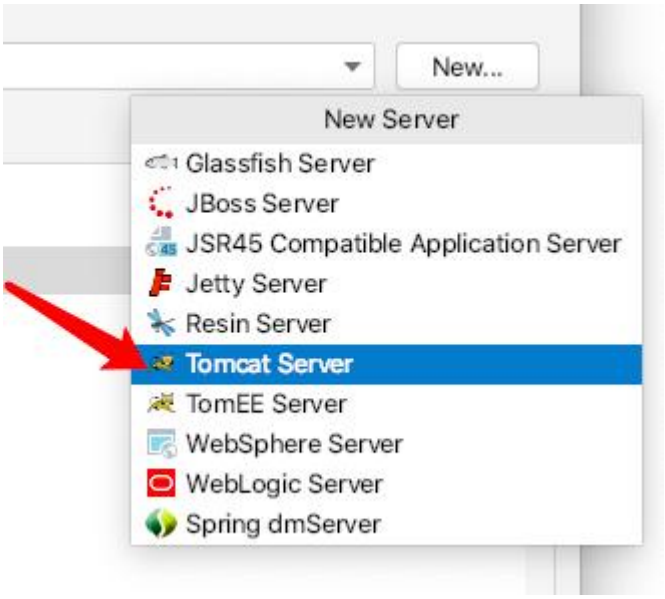
再给大家出一道思考题，现在我们在 myweb 下再创建一个子目录 child，再放一个 aaa.html 网页到 child 中去，那么访问 child 里面的 aaa 页面，网址路径该怎么写？

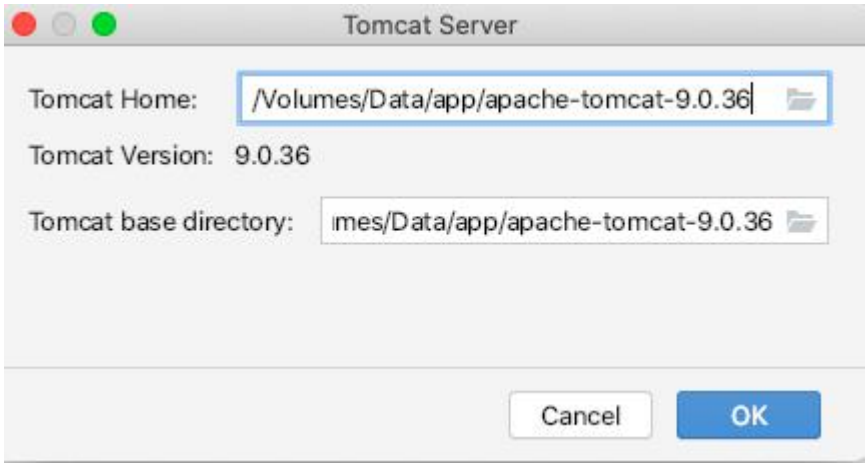
这个网页的路径就是 localhost:8080/myweb/child/aaa.html

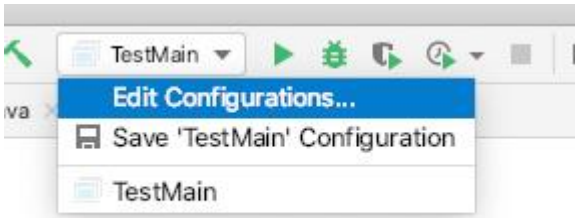
这个规律应该很好总结：localhost:8080 后面写的就是这个资源在 webroot 下的路径

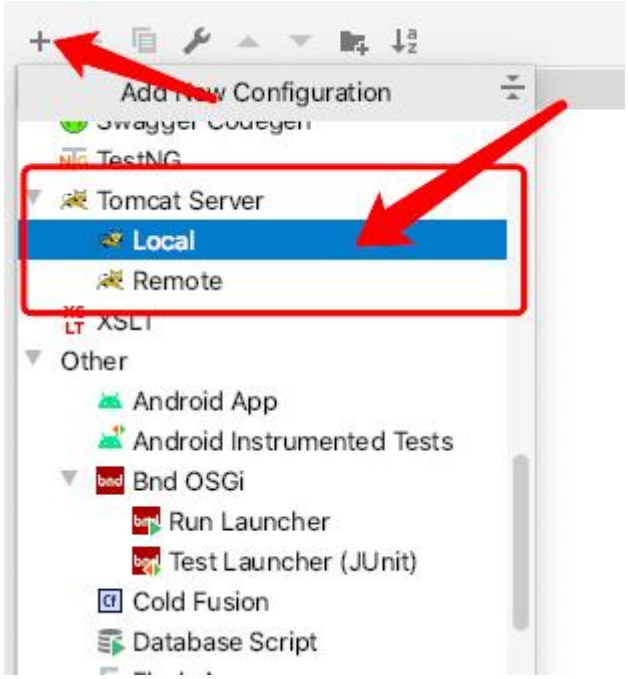
三、项目创建与部署

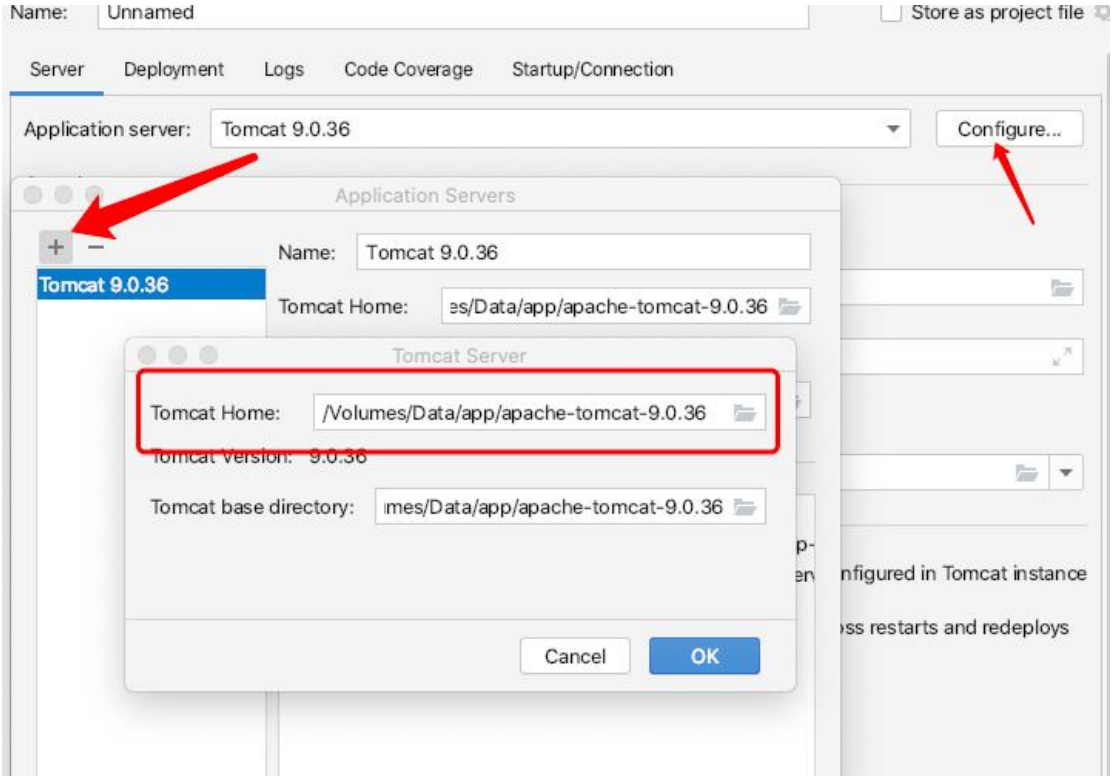
1. Tomcat 与 IDEA 集成

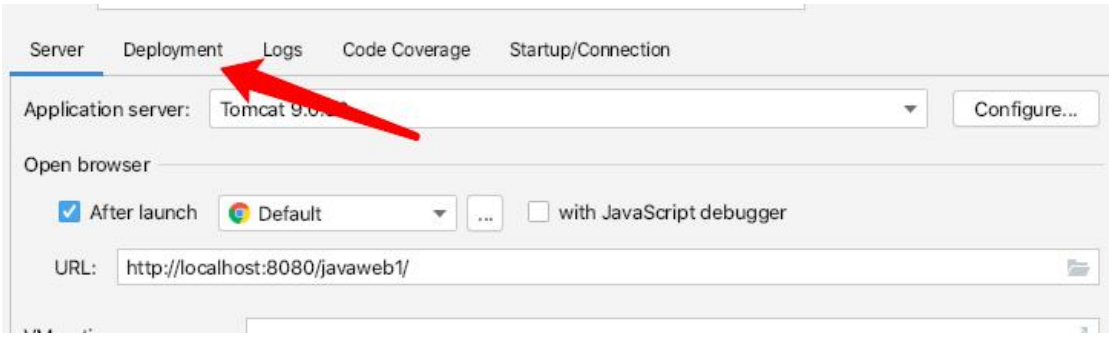


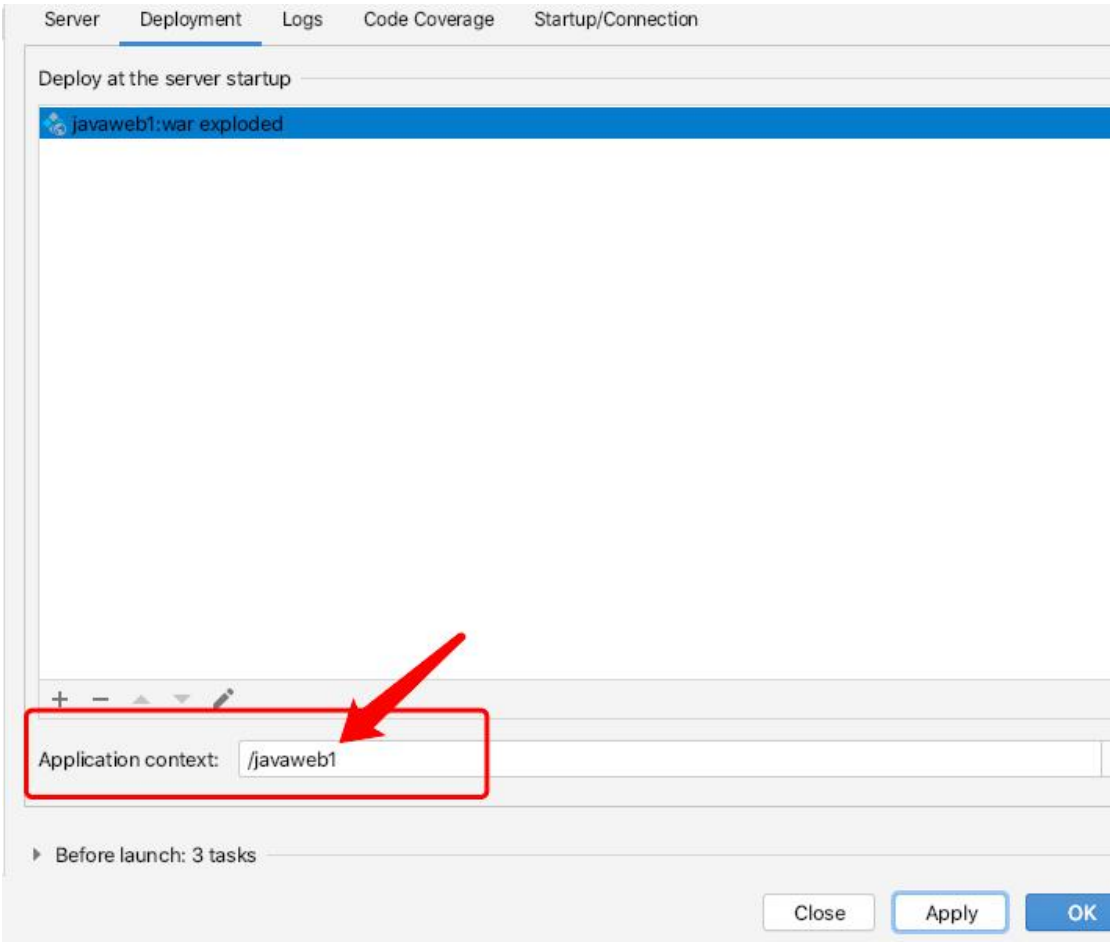

Tomcat Server
Tomcat Home:
/Volumes/Data/app/apache-tomcat-9.0.36
Tomcat Version: 9.0.36
Tomcat base directory:
mes/Data/app/apache-tomcat-9.0.36
Cancel
OK


TestMain
Edit Configurations...
Save 'TestMain' Configuration
TestMain


Add New Configuration
TestNG
Tomcat Server
Local
Remote
XSLT
Other
Android App
Android Instrumented Tests
Bnd OSGi
Run Launcher
Test Launcher (JUnit)
Cold Fusion
Database Script

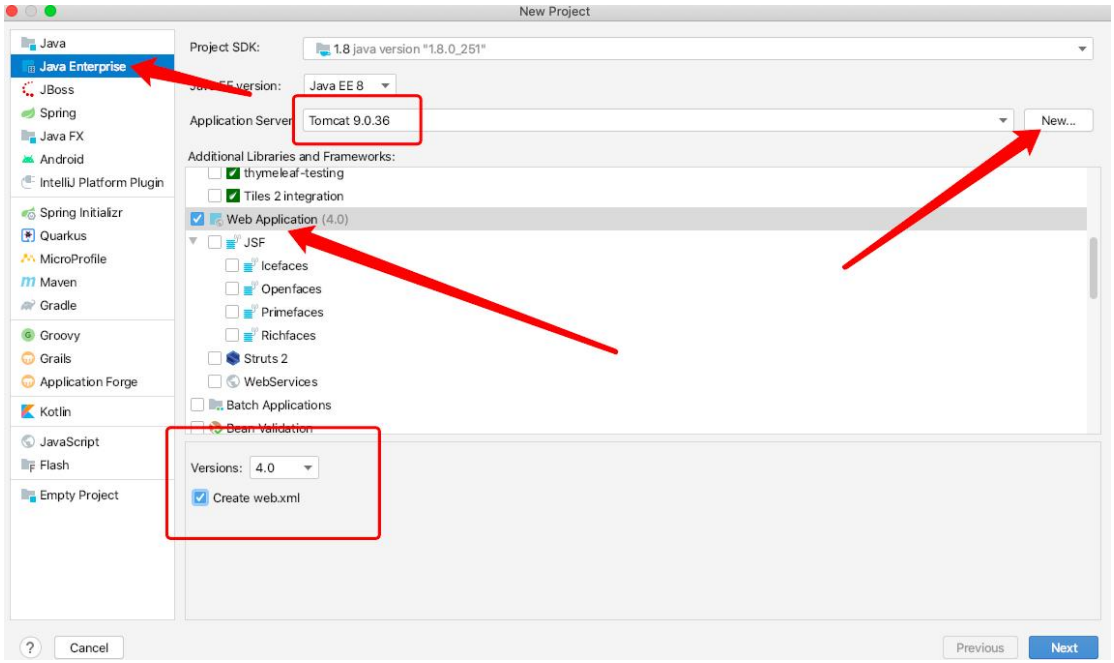

Name: Unnamed
Store as project file
Server
Deployment
Logs
Code Coverage
Startup/Connection
Application server: Tomcat 9.0.36
Configure...
Application Servers
Tomcat 9.0.36
Name: Tomcat 9.0.36
Tomcat Home: es/Data/app/apache-tomcat-9.0.36
Tomcat Server
Tomcat Home: /Volumes/Data/app/apache-tomcat-9.0.36
Tomcat Version: 9.0.36
Tomcat base directory: imes/Data/app/apache-tomcat-9.0.36
Cancel
OK
nfigured in Tomcat instance
ss restarts and redeploys


Server
Deployment
Logs
Code Coverage
Startup/Connection
Application server:
Configure...
Open browser
After launch
Default
with JavaScript debugger
URL: http://localhost:8080/javaweb1/

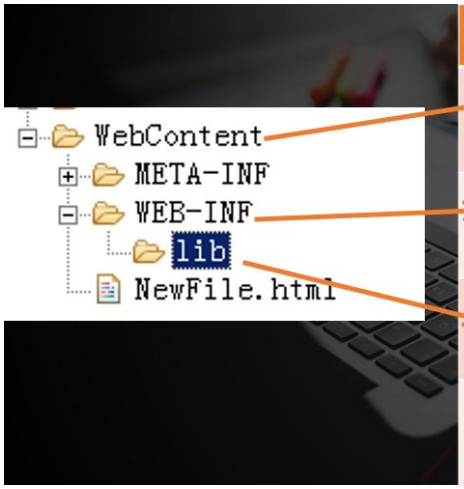

Server
Deployment
Logs
Code Coverage
Startup/Connection
Deploy at the server startup
javaweb1:war exploded
Application context:
/javaweb1
Before launch: 3 tasks
Close
Apply
OK


Server
Deployment
Logs
Code Coverage
Startup/Connection
Application server:
Tomcat 9.0.36
Open browser
After launch
Default
with JavaScript debugger
URL:
http://localhost:8080/javaweb1/
VM options:

2. 创建 JAVAWEB 工程



3. Web 项目的目录结构



目录	说明
/	Web应用的根目录，该目录下所有文件在客户端都可以访问(JSP、HTML等)
/WEB-INF	存放应用使用的各种资源，该目录及其子目录对客户端都是不可以访问
/WEB-INF/lib	存放Web应用使用的JAR文件
/WEB-INF/classes	存放Web项目的所有的class文件

四、什么是 Servlet

1. 什么是 Servlet

以用户注册功能为例，首先用户通过请求响应拿到了注册页面

在注册页面上填写完信息后，点击注册按钮，又一次的向服务器端发送请求

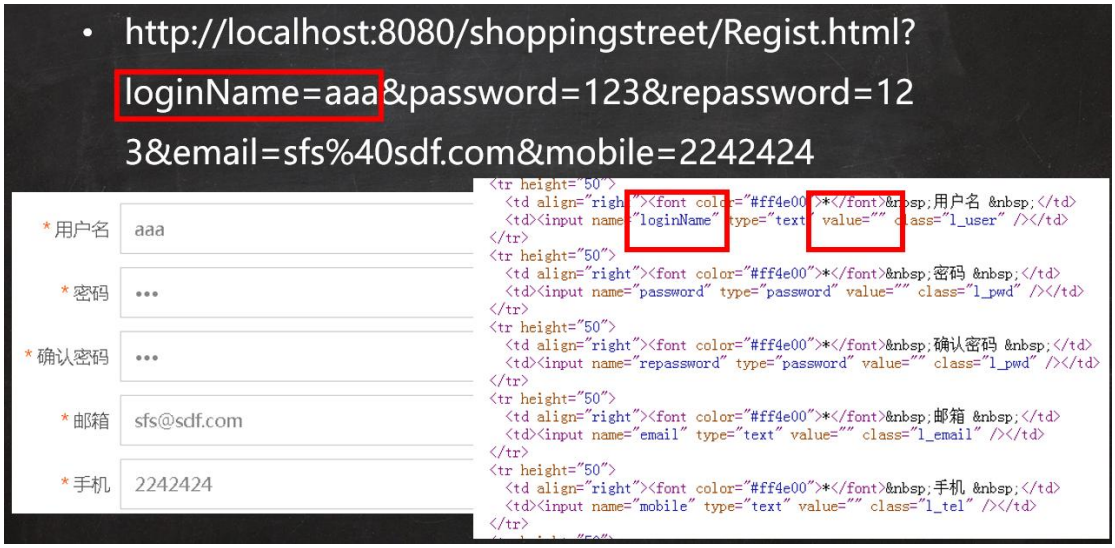
这里产生了第一个问题，用户信息是怎样通过请求发送到服务器的

接下来会有第二个问题，服务器端接收到这个请求后，怎么去处理，怎么取出请求中的用户信息

最后第三个问题，服务器怎么把注册成功的页面返回给浏览器

① 第一个问题：用户信息是怎样发送到服务器的

打开注册页面，输入用户名，密码等信息后，点击注册按钮，页面刷新了，注意看浏览器地址栏上的内容，出现了变化



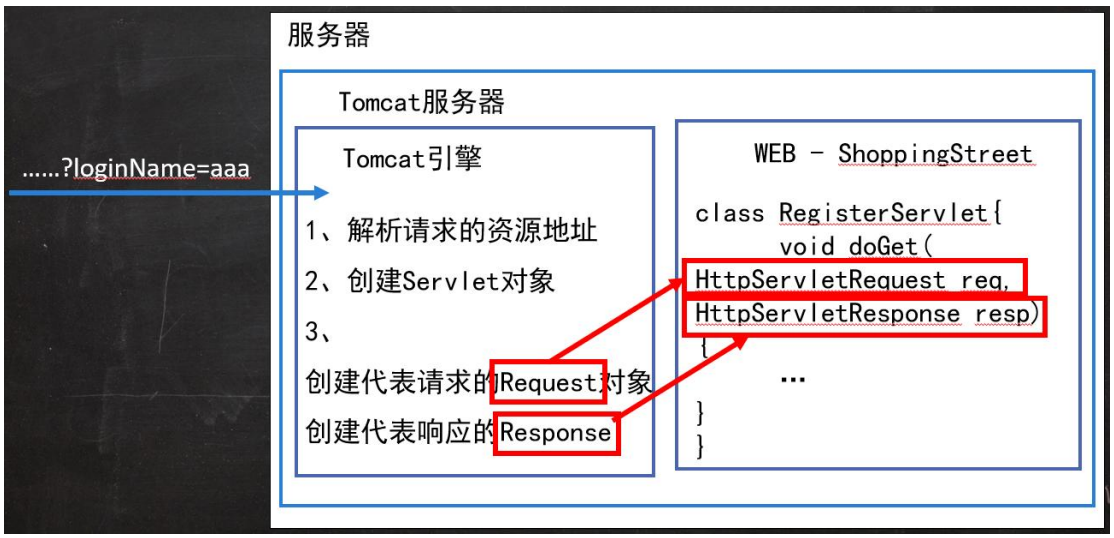
用户在文本框中输入的值，其实等同于 input 标签里面的 value 属性中的值。再看 url 路径，很明显，浏览器使用了标签 name 属性值=value 属性值得方式，把表单里面的数据，放在 url 中，每个表单元素之间用&符号分割，整个的表单元素内容放在正常的资源路径的后面，用？

分割开

这样随着 url 路径发送到服务器端，表单元素里面的值也被发送到服务器端了

② 第二个问题，请求到达服务器之后，怎么处理请求

先看第一幅图片、



如图，当请求到达服务器端之后，服务器运行一段 java 程序处理这个请求，并生成返回给浏览器的 html 代码

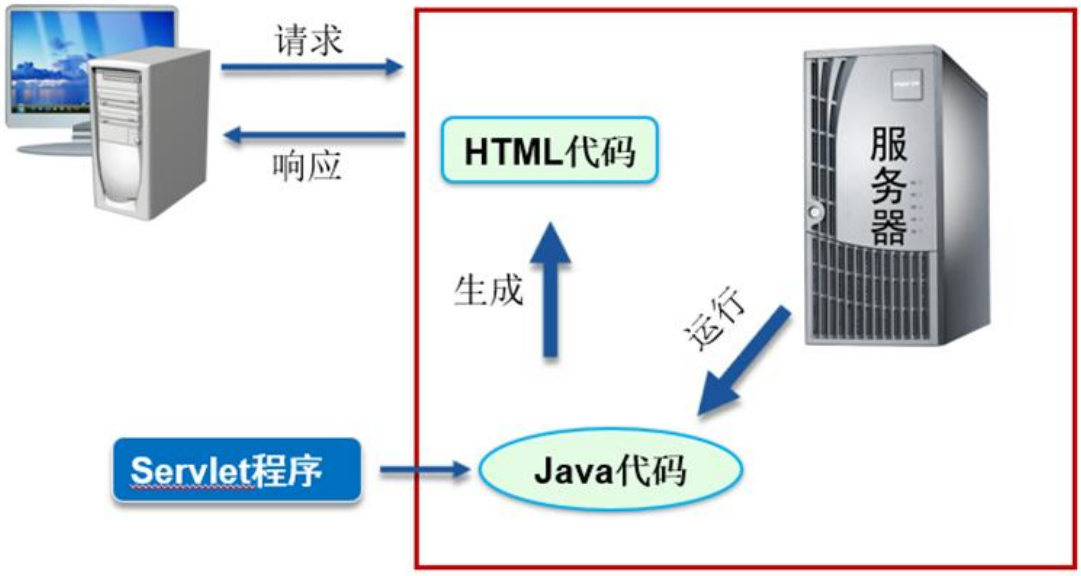
所以处理请求的是运行在服务器端的 java 程序，我们将这个 java 程序称为 Servlet

并且我们在这里也解决了第三个问题：

③ 如何将响应的内容返回给浏览器

也是通过这个 Servlet 程序

④ 总结



浏览器的请求到达服务器之后，首先由 tomcat 接收和处理，tomcat 引擎第一步就会把请求的地址给解析了，找到对应的 servlet 对象，当然如果这个对象不存在就看看能不能实例化一个，第三步就是创建一个 request 对象封装浏览器发送给服务器端的数据，再创建一个 response 对象封装我们返回给浏览器的数据

最后它会调用 servlet 中的方法去处理这些数据

所以在 servlet 中如果想获取用户发送过来的数据，就从 request 取，想把什么数据返回给浏览器就把数据放到 response 对象中

2. 简单 Servlet 编写与注解配置

```
package club.circle.controller;

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

```
import java.io.PrintWriter;


@WebServlet(name = "TestServlet",urlPatterns = "/test.do")
public class TestServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        System.out.println("doPost");
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        System.out.println("doGet");
    }
}
```

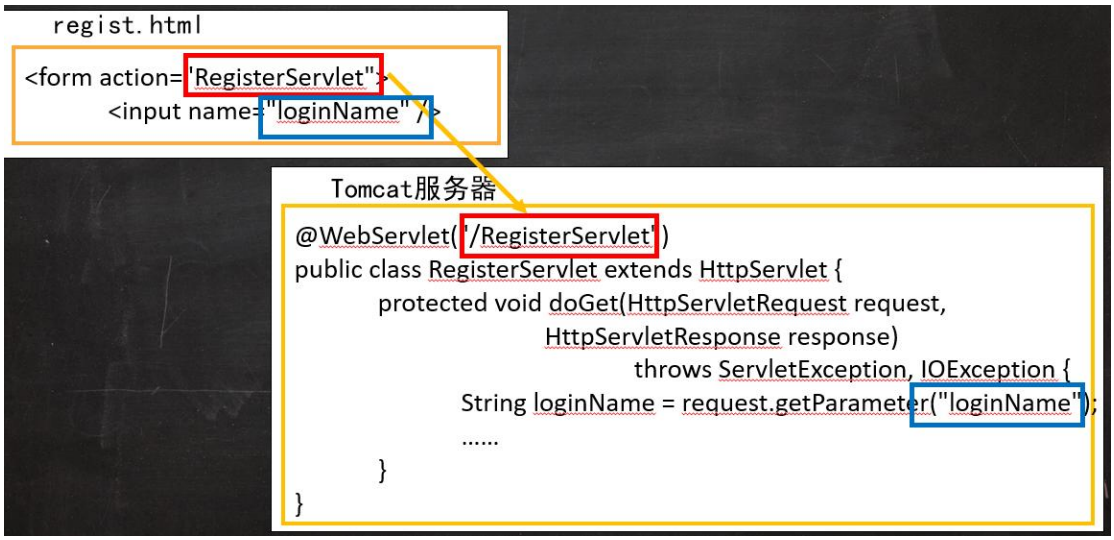
3. Response 输出页面

```

```

五、请求提交的两种方式

1. Request 对象接收参数详解



首先在 html 页面上，添加表单的 action 属性，属性的值对应着 servlet 上面 WebServlet 这个注解中的值

显然这个值代表着一个路径，tomcat 就是通过这个路径找到用户想要调用的 servlet

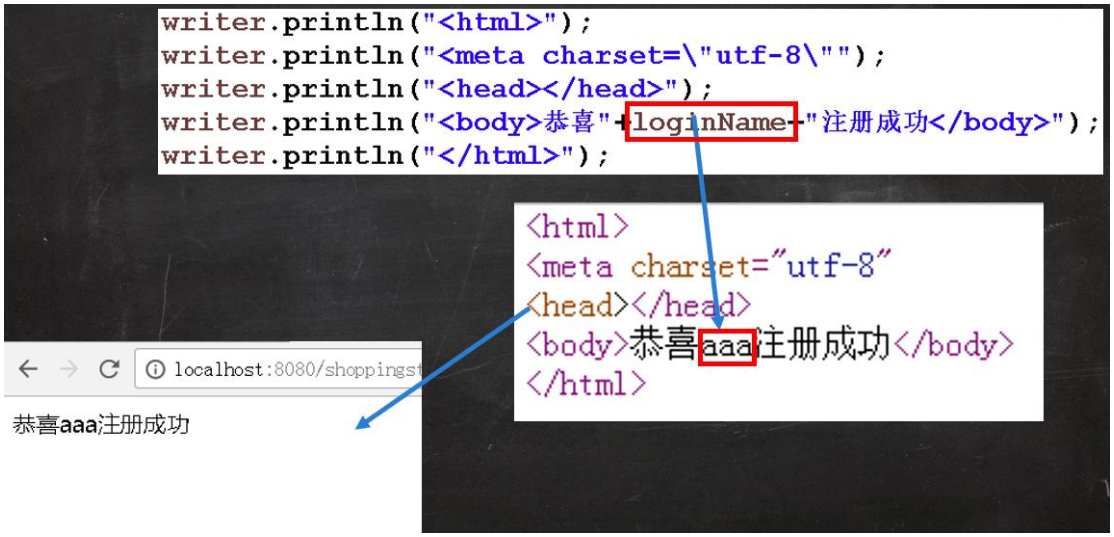
或者大家也可以这样去思考，一个服务器上肯定不止一个 servlet，那么用户请求到达服务器端后，该调用哪个 servlet 去处理用户请求呢，就是靠这个路径去找到对应的 Servlet

接下来就重要了，request 对象封装了用户的请求信息，自然就包含了用户在注册表单中填写的注册信息，用户名，密码什么的，怎么取出这些信息呢

使用 request 对象中的 getParameter 方法

用户提交的信息有很多，你要取出哪一个，在 getParameter 方法的参数中写上表单元素的 name 属性，你写上 loginName 就能取出用户名，写上 password 呢，就能取出密码

2. Response 输出页面代码详解



PrintWriter 实际就是一个 io 类，和之前学到的 bufferedWriter 是属于同样的输出流

现在开始打印输出一组 html 标签，这个就是输出到浏览器上的内容

3. Get 方式与 Post 方式提交

首先就是用户的注册数据是通过 url 路径传递到服务器端的
各位同学，你们觉得这样好吗
或者换个场景，如果是登录呢，用户把自己的登录名和密码暴露到地址栏上，这肯定是不对的。
另外不知道大家注意过没有，在 servlet 中我们用了 doGet 方法处理用户请求，它下面还有一个 doPost 方法是干什么的呢

Get 方式提交是默认的表单提交方式，除此之外，我们通过点击超链接访问服务器，也是用 Get 方式提交的，直接在浏览器地址栏上敲入一个网址，访问服务器也是 Get 方式提交

而 Post 方式提交，你需要手动的在 form 标签中进行指定 method
<form action="RegisterServlet" method="post">
method 默认值为 Get，这里改成 Post
回到 regist.html，刷新，重新提交表单
你会发现这时候在请求头部中显示的就是 post，资源路径的后面就没有表单元素的值了，在

浏览器地址栏中也是一样

值在哪里呢，我们这里看到的是请求的头部，还有请求的 body，请求体，表单内容在请求体里面
所谓的请求体，其实就是放在请求头部后面的字符串，不要被这个名词吓到

4. 实例：用户注册信息的提交

RegisterServlet

```
package com.shoppingstreet.servlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.shoppingstreet.entity.User;
import com.shoppingstreet.service.UserService;
import com.shoppingstreet.service.impl.UserServiceImpl;

/**
 * Servlet implementation class RegisterServlet
 */
@WebServlet("/RegisterServlet")
public class RegisterServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

	/**
	 * @see HttpServlet#HttpServlet()
	 */
	public RegisterServlet() {
```

```
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String loginName = request.getParameter("loginName");
        String password = request.getParameter("password");
        String repassword = request.getParameter("repassword");
        String email = request.getParameter("email");
        String mobile = request.getParameter("mobile");
        String isAgree = request.getParameter("isAgree");

        User user = new User();
        user.setLoginName(loginName);
        user.setPassword(password);
        user.setMobile(mobile);
        user.setEmail(email);

        UserService userService = new UserServiceImpl();
        boolean flag = userService.add(user);

        response.setCharacterEncoding("UTF-8");
        PrintWriter writer = response.getWriter();

        writer.println("<html>");
        writer.println("<meta charset=\"utf-8\"");
        writer.println("<head></head>");
        if(flag==true){
            writer.println("<body>恭喜"+loginName+"注册成功</body>");
        }
        else{
            writer.println("<body>注册失败</body>");
```

```
            }
            writer.println("</html>");

            writer.flush();
            writer.close();
        }

        /**
         * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
         */
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
            // TODO Auto-generated method stub
            doGet(request, response);
        }

}
```

4. 表单提交深入

在注册表单还有一个复选框，和之前的文本框就不一样，如何让服务器知道它已经被选中了呢

首先我们给这个复选框一个 name 属性, 就叫 isAgree 好了, 再给它设置一个 value 属性, value 设置一个默认值 true

<input type="checkbox" name="isAgree" value="true" />

然后在服务器端写上接收参数的代码

```
String isAgree = request.getParameter("isAgree");
System.out.println(isAgree);
```

重启服务器，我们测试一下

可以看到完全没有问题，但是如果我不选中就直接提交到服务器端呢，
服务器端显示的是 null

具体情况可以把表单的 method 改为 get 方式提交，这样方便查看表单提交的内容

可以看到 isAgree 并没有提交过去，所以服务器端显示 null

好了下面看另一个细节问题，还是和复选框有关
一个非常简单的表单，3 个复选框，表示的是一组兴趣爱好，它们的 name 属性，都叫做 fav
有个 submit 类型的提交按钮，用于将表单提交给服务器

这里我们再新建一个 TestServlet，让表单的 action 指向这个 Servlet

接下来的问题是，在 Servlet 中该怎样写，才能接收到多个复选框中的值，我们可以先看下提交的参数是什么形式的
可以看到同一个 fav 对应了三个值，传递到服务器端，如果你是 javaweb 的架构师，你会用什么方式去接收这三个值?

没错，就是数组
在 request 对象中有个能接收相同参数名，但是有个多个值得方法，getParameterValues
我们一起用用看

```
String[] values = request.getParameterValues("fav");
for(String s : values){
    System.out.println(s);
```

```
}
```

六、Servlet 的 xml 配置

1. Servlet 的 XML 配置

```
    <servlet>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>com.shoppingstreet.servlet.LoginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LoginServlet</servlet-name>
        <url-pattern>/loginServlet</url-pattern>
    </servlet-mapping>
```

2. 案例：登录功能完成

```
package com.shoppingstreet.servlet;

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

import com.shoppingstreet.entity.User;
import com.shoppingstreet.service.UserService;
import com.shoppingstreet.service.impl.UserServiceImpl;

/**
 * Servlet implementation class LoginServlet
```

```
 */
public class LoginServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

	/**
	 * @see HttpServlet#HttpServlet()
	 */
	public LoginServlet() {
		super();
		// TODO Auto-generated constructor stub
	}

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		String loginName = request.getParameter("loginName");
		String password = request.getParameter("password");

		UserService userService = new UserServiceImpl();
		User user = userService.getLoginUser(loginName, password);
		if(user!=null){
			System.out.println("登录成功");
		}else{
			System.out.println("登录失败");
		}
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
		doGet(request, response);
```

```
    }
}
```

3. 请求重定向

录成功后，我们想显示首页怎么办，
把首页的内容直接输出到页面上去？这样太麻烦了，可以换一种思路，比如告诉浏览器，你接下来需要访问的页面是 index.html，然后浏览器重新发送一次请求，去获取 index.html 页面，不就可以了吗
那，如何让浏览器知道要重新发一次请求，去获取首页呢
这就得用到我们的第三个知识点，请求重定向技术
response.sendRedirect();

复制 index 页面到根目录
在 servlet 中用 Response.sendredirect 替换掉输出语句

```
package com.shoppingstreet.servlet;

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

import com.shoppingstreet.entity.User;
import com.shoppingstreet.service.UserService;
import com.shoppingstreet.service.impl.UserServiceImpl;

/**
 * Servlet implementation class LoginServlet
 */
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
```

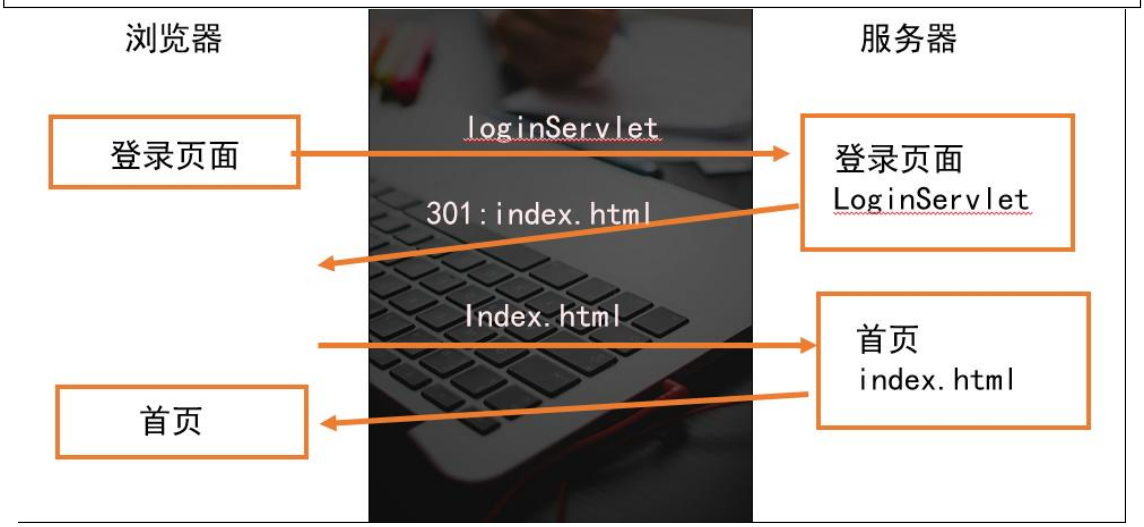
```
    /**
     * @see HttpServlet#HttpServlet()
     */
    public LoginServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String loginName = request.getParameter("loginName");
        String password = request.getParameter("password");

        UserService userService = new UserServiceImpl();
        User user = userService.getLoginUser(loginName, password);
        if(user!=null){
            //System.out.println("登录成功");
            response.sendRedirect("index.html");
        }else{
            System.out.println("登录失败");
        }
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }
```

```
}
```



首先用户通过登录表单向服务器发送登录请求，请求的是 LoginServlet，在 Servelt 中我们判断登录成功，返回响应到浏览器，这个时候响应的内容主要包括两块，一个是 301 状态码，之前我们将 http 响应头部的时候，看到过 200 状态码，代表的是正常响应，301 状态码的含义是告诉浏览器，我给你的不是页面内容，而是一个 url，你得重新发送一次新的请求，，在这个例子里面就是浏览器接收到 index.html 这个页面地址，所以浏览器重新向服务器端发送了新的请求，这次请求的就是 index 页面，服务器接收到这个请求之后，就把 index 页面响应给了浏览器，浏览器上就显示了首页内容