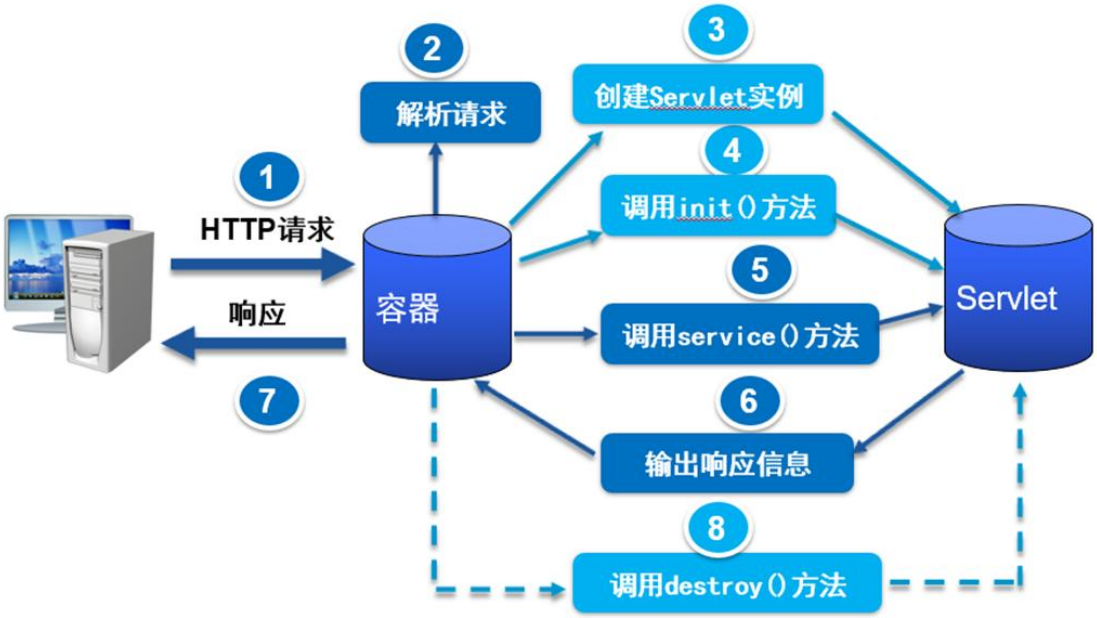
第二讲 MVC 架构模式

一、Servlet 生命周期



打开自己定义的 servlet，对照着图，看一下图中讲到的方法是不是都出现在了代码里面。

会发现一个都不在，无论是初始化的 init 方法，还是 service，还是最后的 destroy 方法都不在。

那么这些方法在哪里呢，仔细看 servlet 的定义，它还有一个父类，这些方法是不是在父类中呢，可以打开源代码看下父类的代码

看父类 HttpServlet 的源码，在代码的后半段看到了 service 方法，在这个方法中，我们可以看到它在调用 doGet 方法和 doPost 方法，这一点请记好，很重要的，后面在讨论生命周期的时候会用到，那这一段代码到底是什么个意思呢

Service 的方法中会根据请求的方式，来决定是调用 doGet 方法处理用户请求还是调用 doPost 方法处理用户请求
如果各位同学看代码仔细些，你就会发现和 post 提交方式相同的还有 put 方式，head 方式，只不过这些请求提交方式，我们暂时用不上，所以不用去管

还有就是在 HTTPServlet 中，你还是找不到 init 方法和 destroy 方法，而 service 方法你看仔细写，这个方法上面有 override 很显然它也是重写了父类方法，我们继续往上找父类 GenericServlet，它有 init，有 destroy 等方法

在自己定义的 Servlet 中，重新定义 init 方法和 destroy 方法行
在 ServletA 中重写 init 方法和 destroy 方法
右键点 source 选择 override，选中 init 和 destroy

然后我们分别在 init，destroy，构造方法，doGet 方法中各自写上一句话，运行后我们就可以看到他们的调用顺序了

这里我们用 doGet 方法，代替 service 方法就可以了

启动 ServletA

可以看到它们的调用顺序

Servlet 构造方法先调用
然后是 init 初始化方法
接下来是 doGet 方法
最后 destroy 这个方法没调用
从名字上我们可以猜出来，这是一个销毁 servlet 对象时调用的方法，现在它没调用说明 servlet 没有被销毁

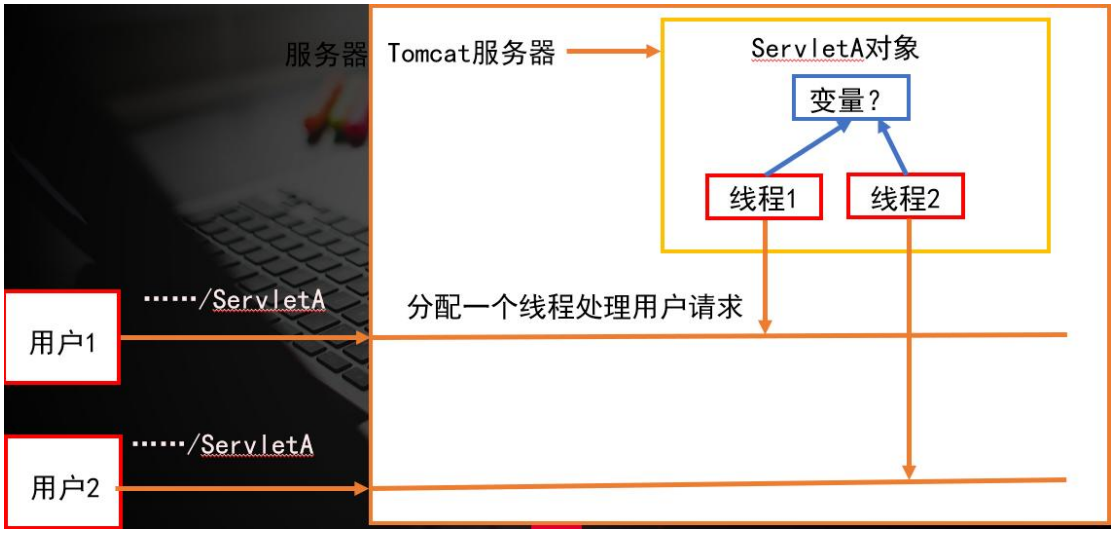
可以这样验证，刷新一下页面，重新发送请求去调用 Servlet
如果 servlet 已经销毁了，它必然要重新调用构造方法实例化，如果它还在服务器内存中，那么当新的请求到达服务器端，就不需要重新实例化，它只要运行 doGet 方法去处理新请求就好了

很明显这次只调用了 doGet 方法处理我的请求
因此 Servlet 一旦实例化完成后就会常驻内存，一般不会被销毁，只有当当前的 Servlet 从 tomcat 移除的时候 destroy 这个方法才会被调用
那什么时候 Servlet 会从 tomcat 移走呢，这个就不好说了，因为这个操作不是我们用代码控制的

总结一下

首先当用户第一次请求到达服务器端之后，
tomcat 解析用户请求，找到它所要调用的 Servlet，接下来调用 Servlet 的构造方法实例化 Servlet，得到它的实例对象，接下来 tomcat 会调用 servlet 的 init 方法，对 servlet 进行初始化，初始化完毕后，tomcat 会调用 service 去处理用户的请求，在处理过程中 service 会判断用户的请求方式，如果是 get 方式请求，就调用 doGet 方法，如果是 post 方式请求就调用 doPost 方法，最后会将结果封装到 response 对象中，交给容器，把响应信息传送到浏览器
服务器端的 Servlet 在处理完这次用户响应之后，会继续待在服务器内存中，等待下一次请求，当第二次请求以至于后面的任意一次请求到达服务器端后，tomcat 都不会再重新实例化 serlvet 了，而是调用第一次请求时创建好的 servlet 去处理用户请求，因此，我们会看到，后面的请求的输出结果都是 doGet，而不会重新做实例化，所以看好，图上面深蓝色部分是每次请求到达服务器端后，都要执行的操作
而浅蓝色部分只执行一次
最后 destroy 方法的调用也是只有一次，就是当 servlet 对象从 tomcat 移除的时候才会执行

二、多线程并发的安全性



一个网站，一个 servlet 不可能每次都处理一个用户的请求，如果有多个用户同时请求这个 serlvet 该怎么办呢

在这里需要重点强调一句话，servlet 是以单实例多线程的方式执行用户请求的

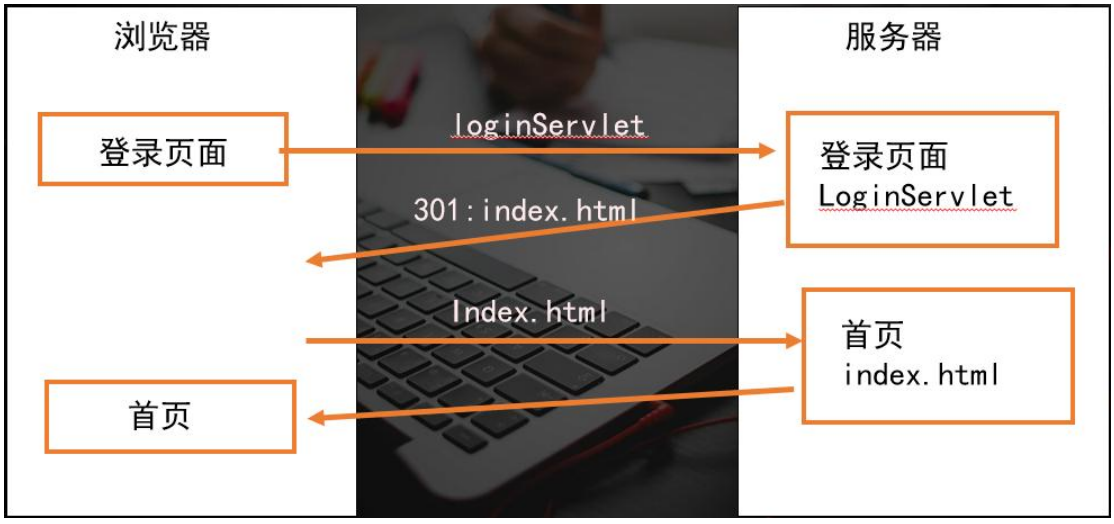
这里用一个简单图示来描述这个过程
这里有用户 1 和用户 2 同时向服务器请求 servletA，这时 tomcat 会找到 ServletA 实例，当然如果这个实例不存在就去实例化一个，然后会为每一个用户分配一个线程去执行用户请求

这就是单实例多线程的含义，既然是多线程，我们就需要避免多线程并发的问题
比如线程 1 和线程 2 如果同时去修改蓝色框中的变量，这不就有可能出现问题了嘛
所以为了避免这个问题，我们在编写 servlet 的时候，不使用类成员变量，只使用在 doGet 或 doPost 方法中定义的局部变量

三、请求转发

1. 什么是请求转发

response.sendRedirect("新页面 url")



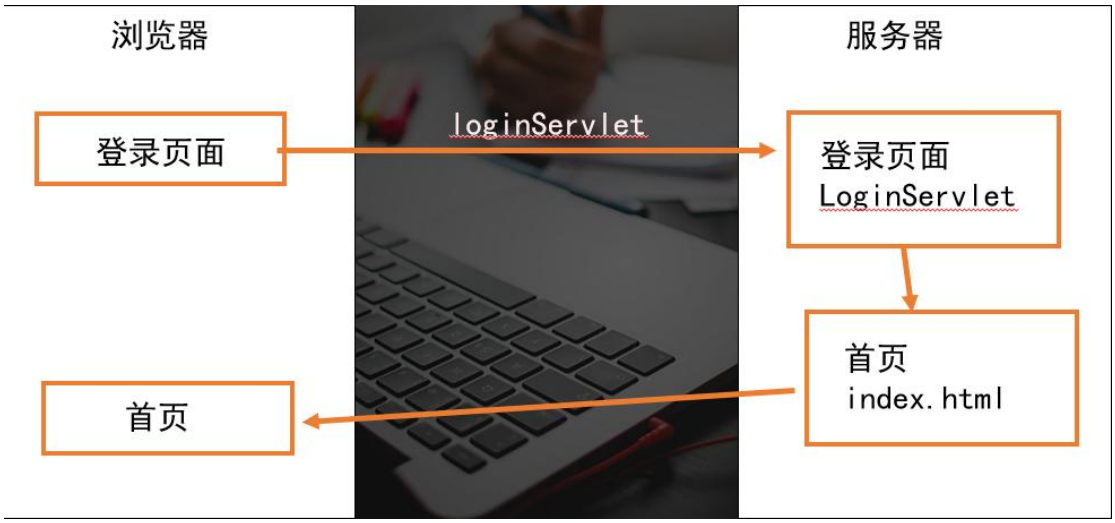
request.getRequestDispatcher(“页面地址") .forward(request, response);
重定向指的是在 servlet 中使用 response.sendRedirect 将新的页面地址返回给浏览器，浏览器重新向服务器端发送请求，请求新的页面从而达成页面跳转的目的

但是这种跳转有个缺陷，什么缺陷呢
如果我们要把数据传递到新的页面，就不太好办了

比如我们在首页输入查询关键字，然后在 Servlet 中查询完毕后，我们怎么把查询结果传递

到，结果显示页面呢，用重定向就不好做这种传值操作

所以接下来我们就讲解一下新的页面跳转方式，叫做请求转发



那么使用过请求转发的页面跳转和使用重定向做出的跳转功能有什么区别呢

用户打开登录页面，请求 LoginServlet，服务器调用 LoginServlet 处理登录请求，这一步和之前的重定向没有任何区别，接下来就是区别所在了
用户处理完登录请求后，想让用户回到首页，这时候可以不需要让用户重新发送请求，因为 index 页面就在服务器端啊，这期间也不需要用户进行什么操作，何必让浏览器再重新发送一次请求，大家想想是不是这个道理

这样就节约了一个请求和响应，所以后面学习的时候，你们会发现，框架默认的页面跳转方式都是请求转发，

```
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");
			request.getRequestDispatcher("index.html").forward(request, response);
			}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);
    }

}
```

2. 请求转发与重定向的区别

转发

转发是在服务器端发挥作用，同将一请求在服务器资源之间进行传递

客户端浏览器的地址栏不会显示转向后的地址

重定向

重定向是在客户端发挥作用，通过发送一个新的请求实现页面转向

在地址栏中可以显示转向后的地址

3. 首页搜索

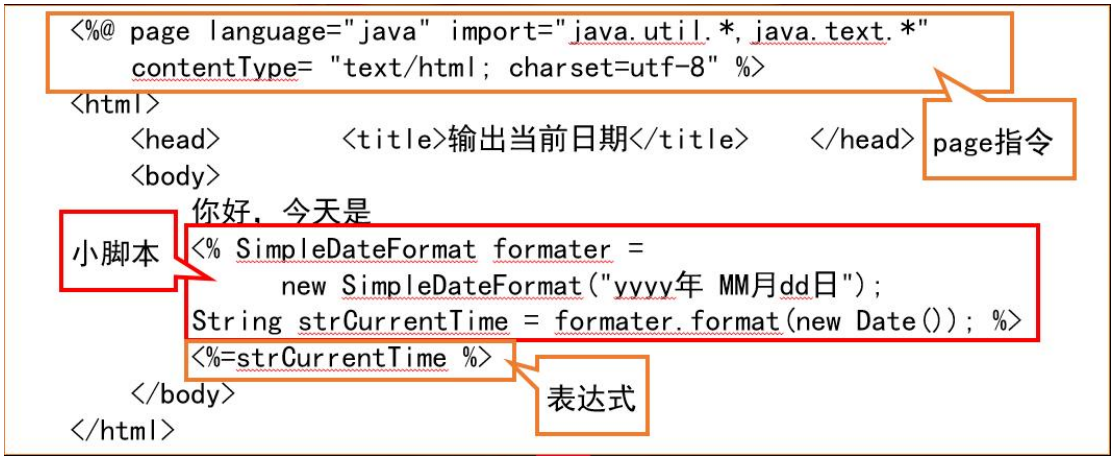

```
package com.shoppingstreet.servlet;

import java.io.IOException;
import java.util.List;

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.Product;
import com.shoppingstreet.service.ProductService;
import com.shoppingstreet.service.impl.ProductServiceImpl;

/**
 * Servlet implementation class SearchServlet
 */
@WebServlet("/SearchServlet")
public class SearchServlet extends HttpServlet {
```

```
    private static final long serialVersionUID = 1L;

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

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String key = request.getParameter("key");
        ProductService productService = new ProductServiceImpl();
        List<Product> productList = productService.getProductList("华为");
        request.setAttribute("productList", productList);
        request.getRequestDispatcher("searchResult.html").forward(request, response);
    }

    /**
     * @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);
    }

}
```

四、什么是 JSP

1. 什么是 JSP

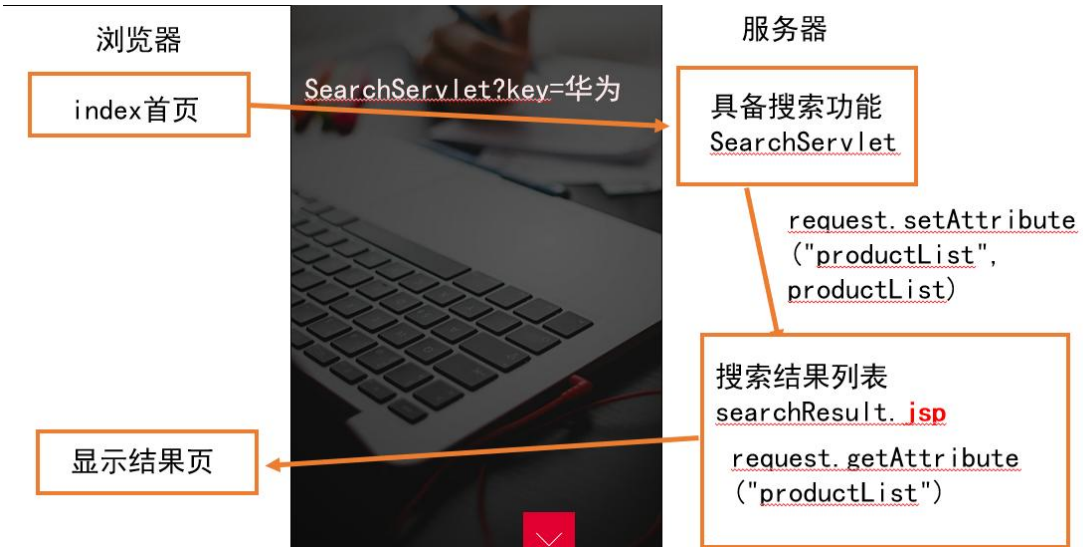
JSP ： Java Server Pages　　在 HTML 中嵌入 Java 脚本代码



2. page 指令

属性	描述	默认值
language	指定 JSP 页面使用的脚本语言	java
import	通过该属性来引用脚本语言中使用到的类文件	无
contentType	用来指定 JSP 页面所采用的编码方式	text/html, ISO-8859-1

3. 添加搜索结果页



五、重构登录页面

1. 客户端与服务器端验证

打开 login.html 把这个登录按钮的类型改成普通 button

再继续添加登录按钮的单击事件代码

```
$(function(){
    $("#loginBtn").click(function(){
        var name = $("#loginName").val();
        var password = $("#password").val();
        if(name.trim()!="" && password.trim()!=""){
            alert("提交表单");
        }else{
            alert("用户名或密码不能为空");
        }
    });
});
```

2. js 提交表单

给 form 添加一个 id：loginForm
然后记住这一行关键代码
$("#loginForm").submit();

客户端验证了，服务器端需不需要验证呢？
很明显是需要的，因为 js 代码的运行实在客户端运行的，所以用户很容易绕过客户端验证，直接向服务器端发送请求，因此所有在客户端验证的操作，都需要在服务器端再做一次，既然服务器端也要做，客户端验证的必要性提心在哪里，很简单嘛，防止用户的误操作，向服务器端提交一些无效请求，能够节省服务器端的资源

进入 LoginServlet

```
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");

        if(loginName=="" || password==""){
            request.setAttribute("errorMsg", "用户名或密码不能为空");
            request.getRequestDispatcher("login.jsp").forward(request, response);
        }
        else{
            UserService userService = new UserServiceImpl();
            User user = userService.getLoginUser(loginName, password);
            if(user!=null){
                //System.out.println("登录成功");
                //response.sendRedirect("index.html");
    //              request.getRequestDispatcher("index.html").forward(request, response);
                    request.getRequestDispatcher("index.jsp").forward(request, response);
            }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);
    }

}
```

在 login.jsp 中显示错误信息

```
<%
    Object obj = request.getAttribute("errorMsg");
    if(obj!=null){
        String errorMsg = obj.toString();
        %>
            <tr height="50" valign="top">
              <td colspan="2">
                  <span style="color:#ff4e00;"><%=errorMsg %></span>
              </td>
            </tr>
            <%
    }
%>
```

六、Servlet 与 JSP

1. Jsp 的本质

结论 jsp 在本质上就是一个 Servlet

在 eclipse 中添加一个 a.jsp

在 a.jsp 中，任意输出点内容，我们这里就输出个当前日期好了

import="java.util.*"

<%=new Date() %>

复制 jsp 到 tomcat 的 root 目录下

启动 tomcat 访问 a.jsp

http://localhost:8080/a.jsp

打开 tomcat 的 work 目录，一路点击下去

你会发现这里有个 a_jsp.java 文件，这个就是一个 java 文件

找到 jspService 方法，在它的输出方法中，有个输出语句

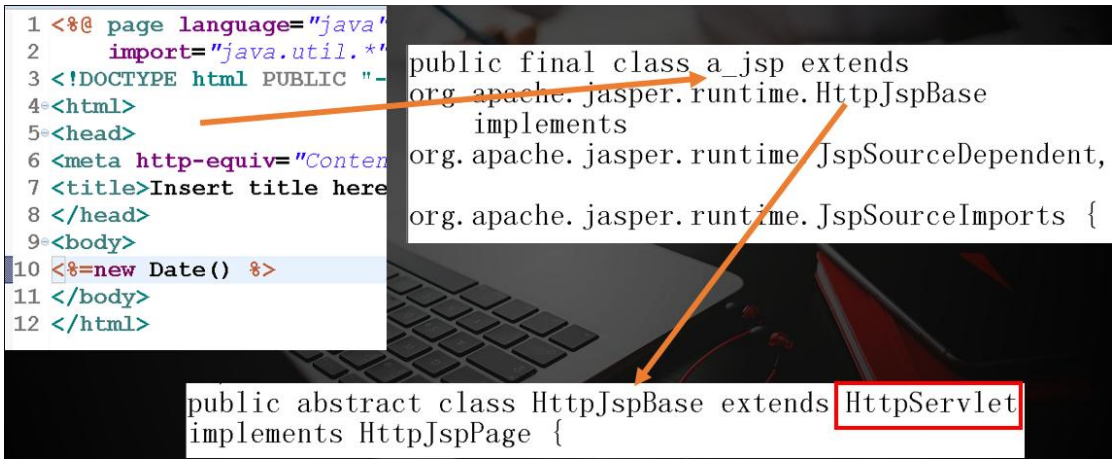
out.print(new Date());

可以佐证，这个 java 文件就对应着 a.jsp 页面

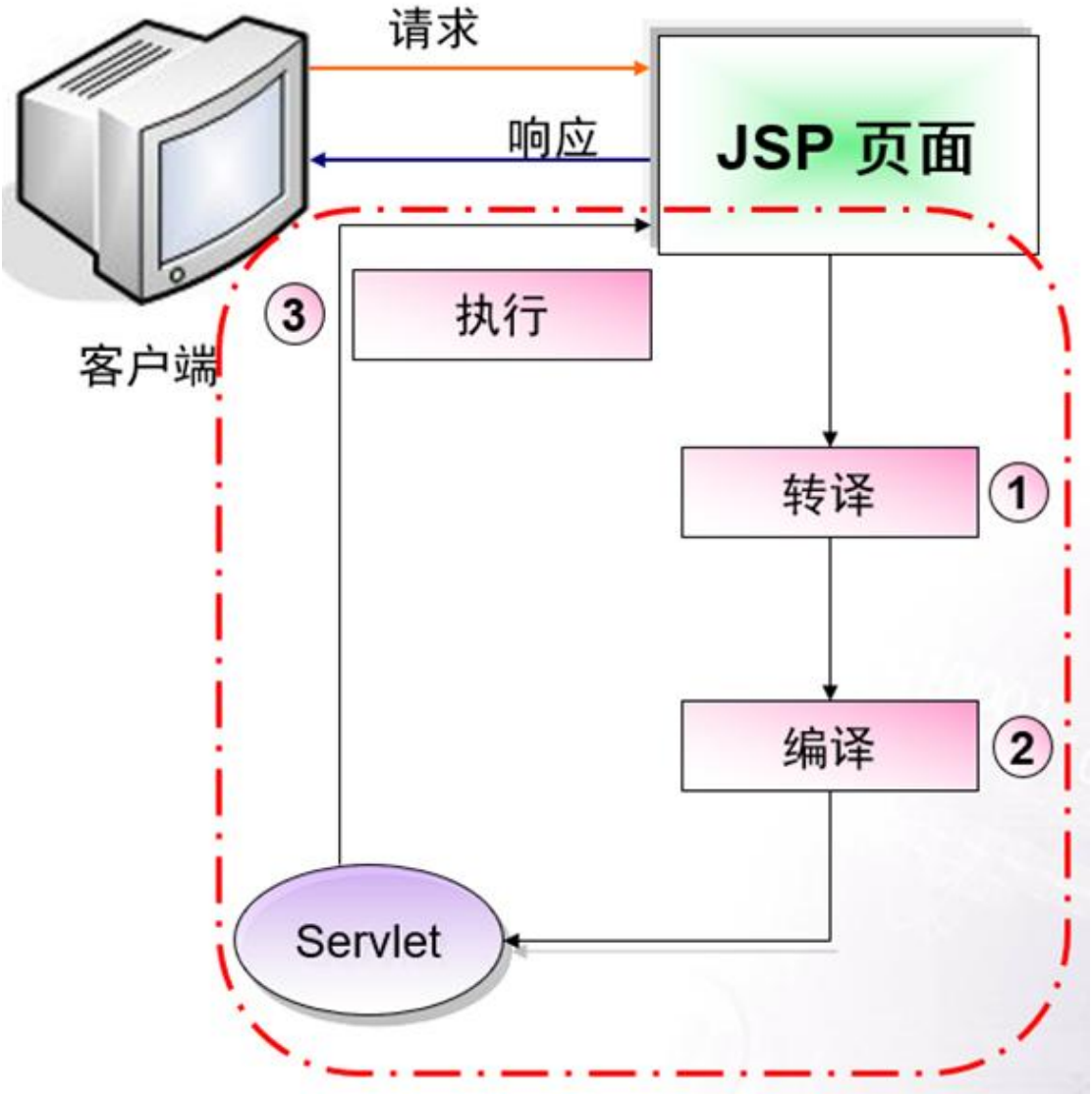
那么这个 class 到底是个什么东西呢，看下这个类的声明

它继承了 HttpJspBase

我们去翻一下代码，就会发现 HttpJspBase 的父类就是 HttpServlet



2. Jsp 的生命周期



当第一次去访问 jsp 页面的时候

tomcat 会将这个 jsp 页面翻译成 servlet java 文件

这个翻译结果就是这里看到的 a_jsp.java 这是一个源代码文件，是不能够执行的对吧

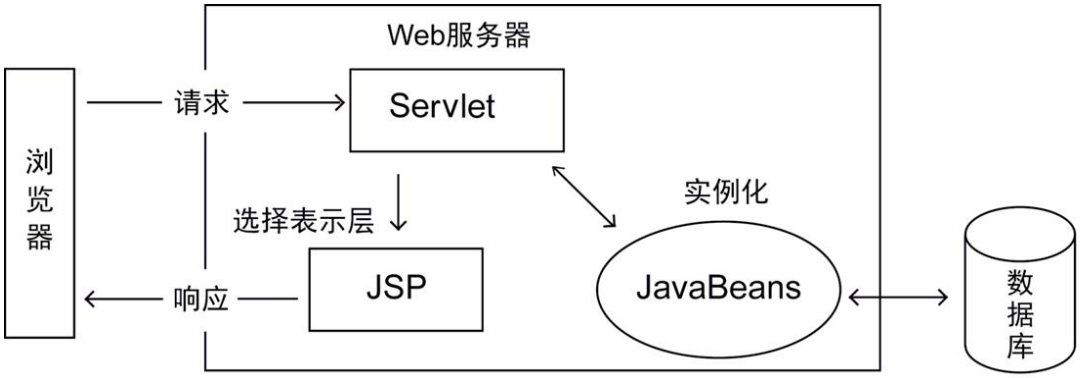
所以 tomcat 继续要做的是把这个 java 源代码文件，编译成 class 字节码文件，也就是这里的 a_jsp.class

然后执行这个 class，这个字节码文件的执行过程就完全和我们的 servlet 执行过程是一致的

当我们第二次访问这个 jsp 页面的时候，还需不需要做翻译和编译的工作?

对的，不需要，只要这个 jsp 页面的代码没有被修改，第二次访问这个 jsp 页面，就直接去调用 servlet 中的 service 方法去处理这个请求就可以了

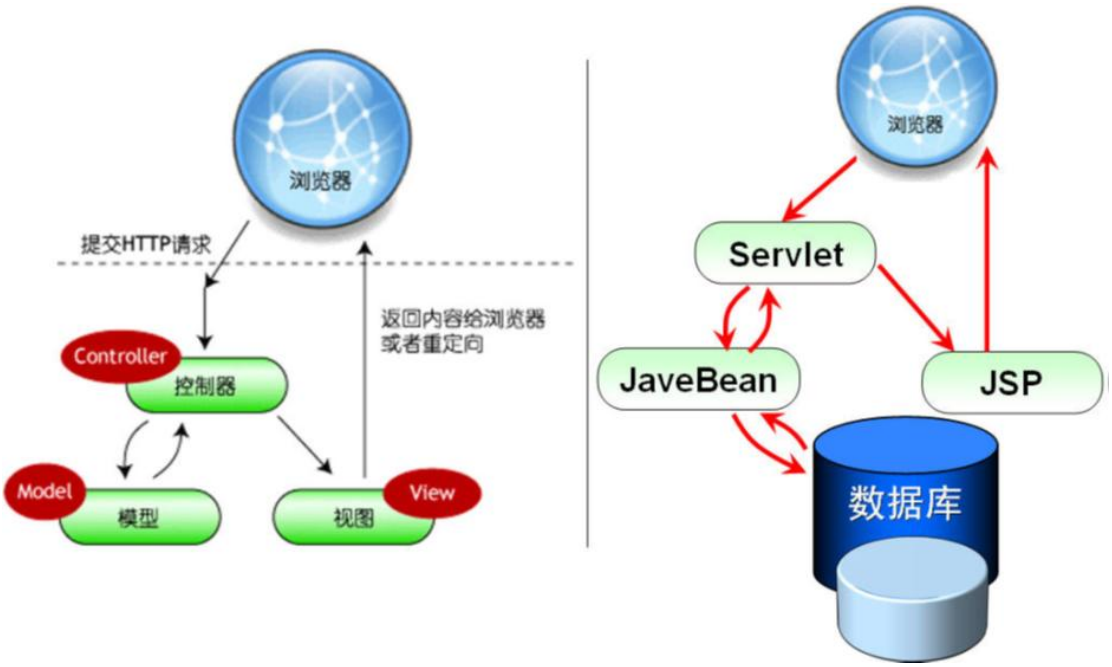
七、MVC



用户请求先到达 Servlet，在 Servlet 中调用 JavaBean 去处理业务，这个 javaBean 是个名词，具体到工程中就是我们这里的 service 和 dao

然后根据 javabean 的返回结果，选择合适的 jsp 页面返回

这个模型就是我们 javaweb 的开发模型，如果把这个模型再抽象，让它不仅仅适用于 javaweb 这一个技术方向，那就变成了我们的 mvc 架构模式



mvc 其实是三个单词的缩写

m 代表 model　模型

v 代表 view　　视图

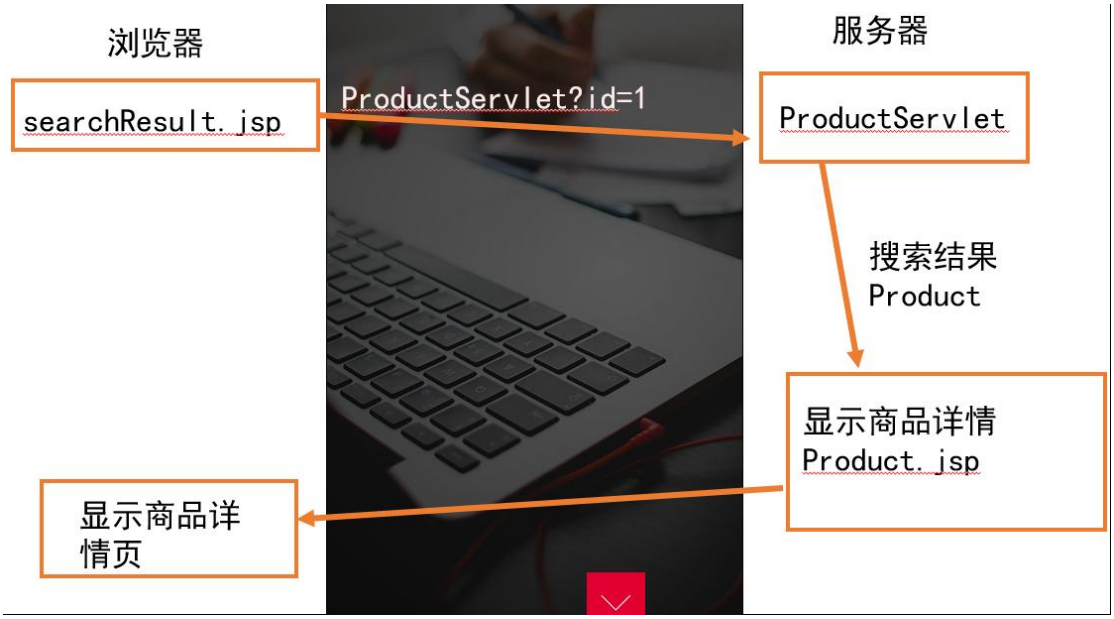
c 代表 controller　控制器

在 javaweb 中 service 和 dao 就是模型，代表着业务和数据模型

视图使用 jsp 技术实现的，用于和用户的交互，或者简单点说，视图就是用户界面，jsp 就是显示一个界面给用户

控制器就是 Servlet 来实现的，它的作用就是控制请求和响应的流程，比如由 Servlet 来决定，调用哪些 service 去处理请求，并且由 servlet 决定，返回给浏览器的是哪个 jsp 页面

案例：商品详情



ProductServlet

```
package com.shoppingstreet.servlet;

import java.io.IOException;
import java.util.List;

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.Product;
import com.shoppingstreet.service.ProductService;
import com.shoppingstreet.service.impl.ProductServiceImpl;


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

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

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		String id = request.getParameter("id");
		ProductService productService = new ProductServiceImpl();
		Product product = productService.getProductById(Integer.valueOf(id));
		request.setAttribute("product", product);
		request.getRequestDispatcher("Product.jsp").forward(request, response);;
	}

	/**
	 * @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);
    }

}
```

到 webcontent 创建 product.jsp
复制 product.html 页面内容到 jsp
添加实体类包的引用
import="com.shoppingstreet.entity.*"
回到 jsp
在这里我们简化一下，只显示商品名称和价格
其余的先不动，我们先看到效果，再去完善

```
<%
    Product product = (Product)request.getAttribute("product");
%>

<div class="pro_des">
    <div class="des_name">
        <p><%=product.getName() %></p>
        “开业巨惠，北京专柜直供”，不光低价，“真”才靠谱!
    </div>
    <div class="des_price">
        本店价格：<b>￥<%=product.getPrice() %></b><br />
        消费积分：<span>28R</span>
```