Java Web

日期: /

maven：架构管理工具

约定大于配置

环境变量（记录一下）：

M2_HOME：bin目录

MAVEN_HOME：mvn目录

PATH：+ MAVEN_HOME

conf → setting.xml：配置文件

镜像 <mirrors>

本地仓库 <setting> → <localRepository>

出现依赖包无法下载成功时

可以单独下载jar包后放到本地仓库

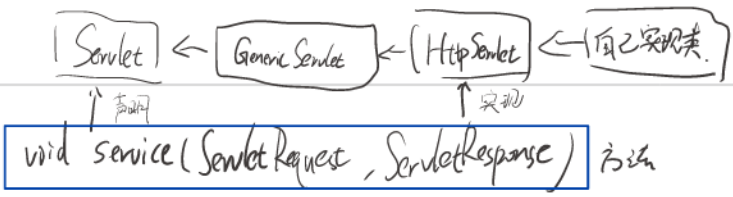
日期: /

初用 Servlet:

1. 编写一个类实现 Servlet 接口
2. 把开发好的类部署到服务器上.

Sun 公司提供了2个接口来实现: HttpServlet / ....

HttpServlet:



请求方式: GET, HEAD, POST, PUT, PELETE, OPTIONS, TRACE

例: HelloServlet.java:

```
PrintWriter writer = resp.getWriter();
writer.print("Hello");
```

在 web.xml:

1. 注册 Servlet. →

```
<servlet>
    <servlet-name></servlet-name>
    <servlet-class></servlet-class>
</servlet>
```

2. Servlet 请求路径. →

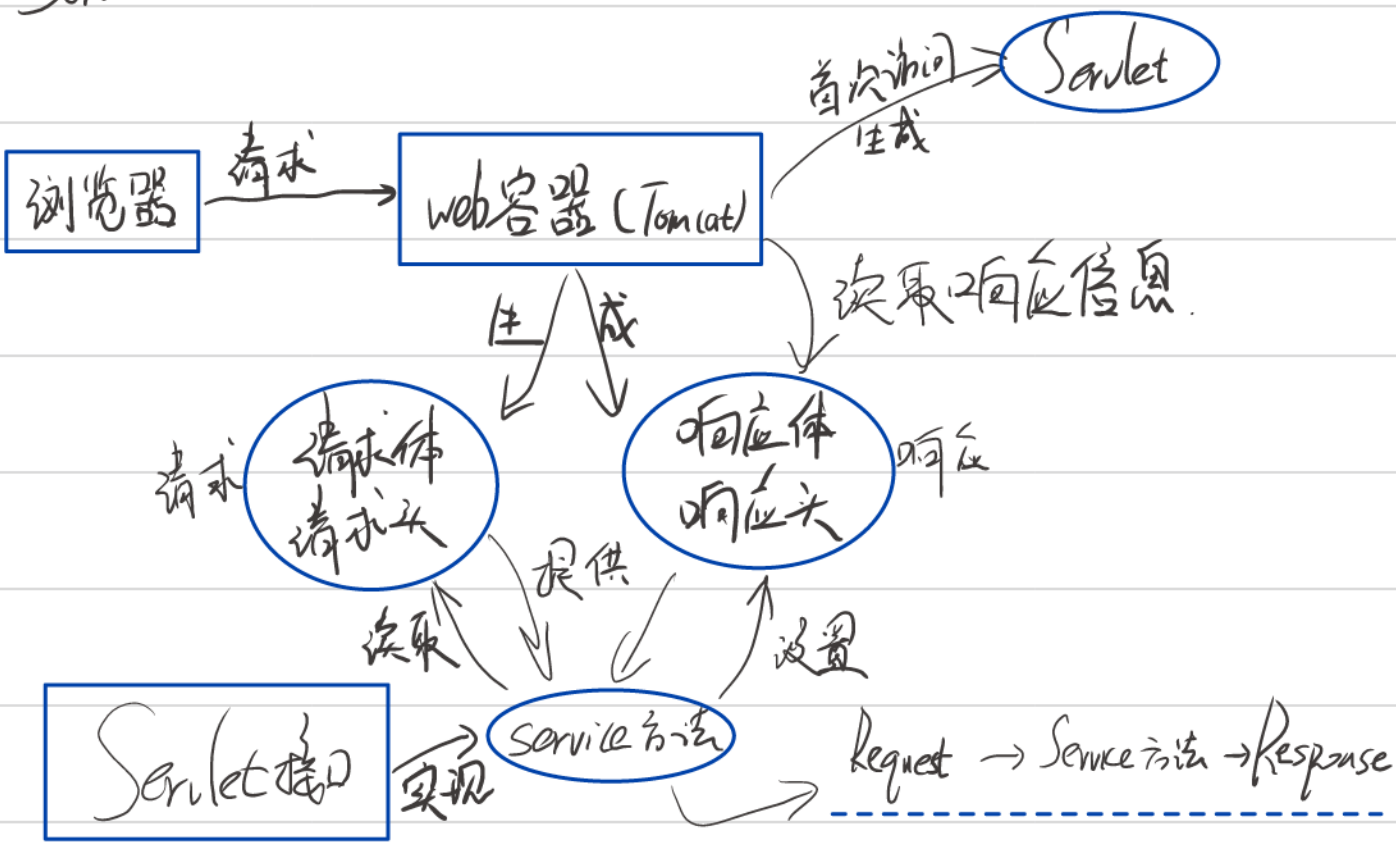
```
<servlet-mapping>
    <servlet-name></servlet-name>
    <url-pattern></url-pattern>
</servlet-mapping>
```

(Mapping)

最后启动 Tomcat.

日期: /

Servlet原理:

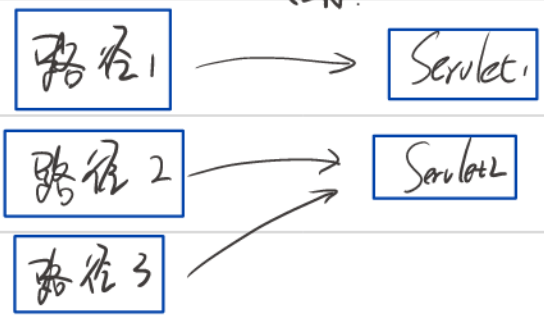


我们自己编写实现类并重写方法.

1. 接收处理请求
2. 封装响应信息

Mapping:

映射



通配符: *

- /hello/*
- /*
- *.do

固定路径优先级最高

日期: /

HttpServletResponse
resp.setContentType("text/html");
resp.setCharacterEncoding("utf-8");

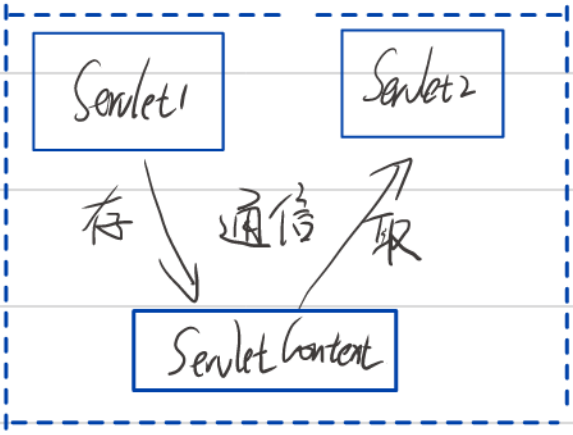
ServletContext 上下文 this.getServletContext();

用处:

① 共享数据

```
String name = "Hashqi";
Context.setAttribute("name", name);
```

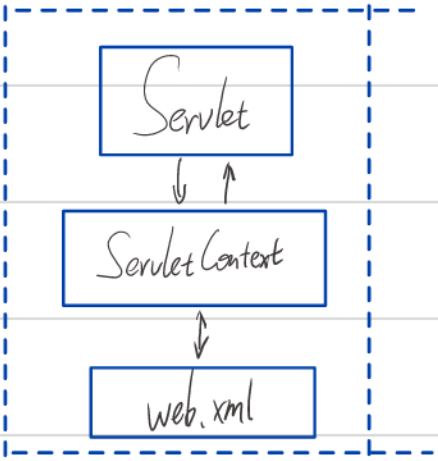
```
(String) Context.getAttribute("name");
```



② 获取 web.xml 中的值

```
<context-param> → { <param-name>
                    <param-value>
```

```
Context.getInitParameter("name");
```



③ 请求转发

```
RequestDispatcher rd = Context.getRequestDispatcher("/hello");
rd.forward(request, response); // 此转发不同于重定向, url 不变
```

④ 读取资源文件 property

```
new Properties().load(Context.getResourceAsStream("/WEB-INF/classes/db.properties"));
    prop.getProperty("name");
```

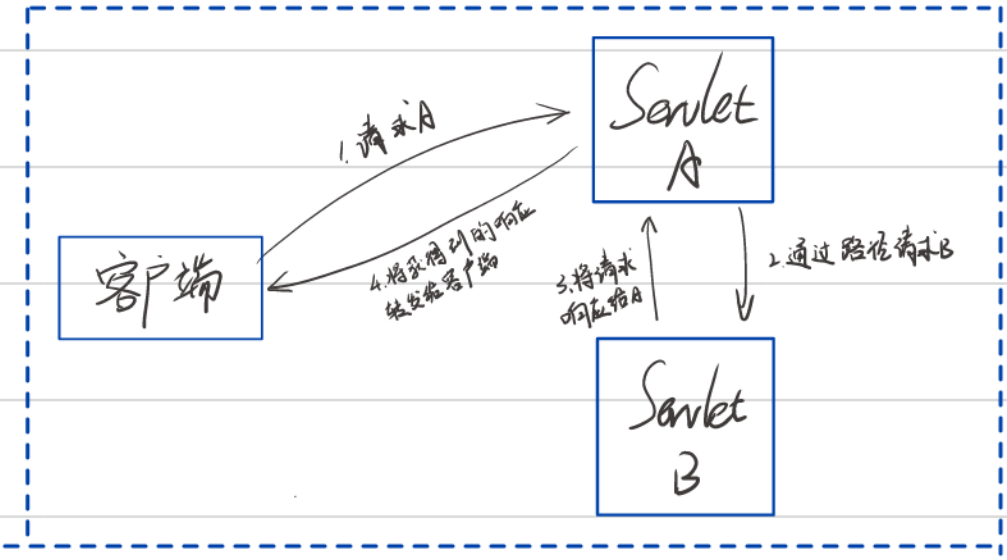
日期: /

forward 和 include

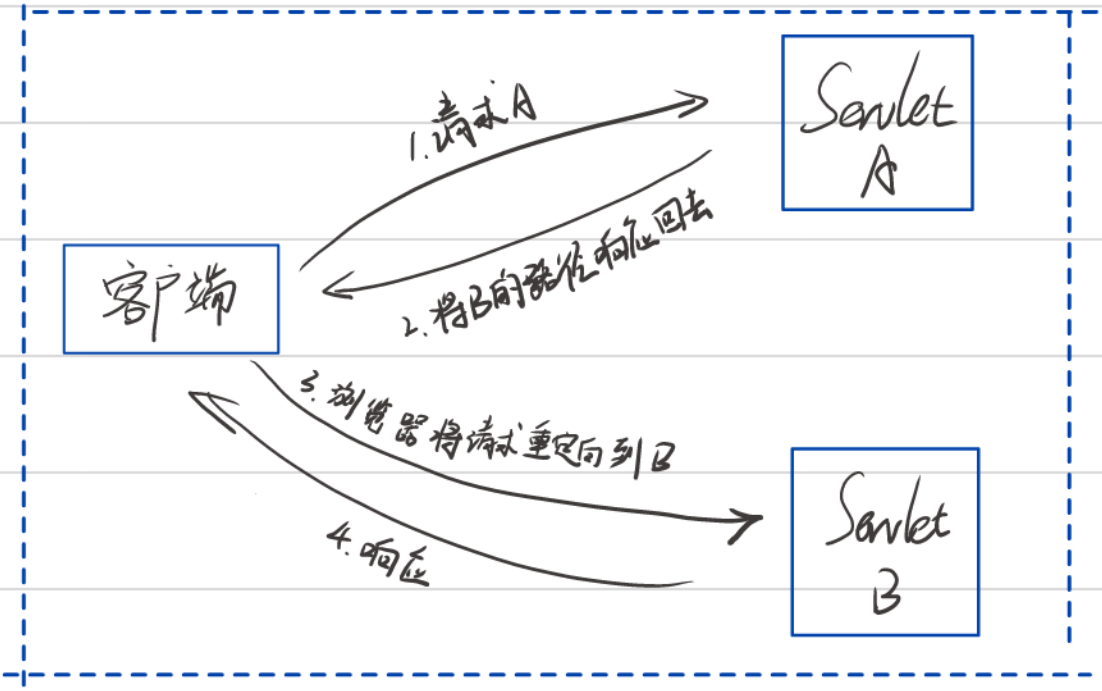
```
this.getServletContext().getRequestDispatcher("/hello").forward(req, resp)
```

~~```
this.getServletContext().getRequestDispatcher("/hello").include(req, resp)
```~~

## forward




## include


```

日期: /

HttpServletResponse

web 服务器接收到客户端的请求后，针对这个请求，分别创建一个 Request 和一个 Response 对象

方法分类：

1. 向客户端发送数据　2. 向客户端发送响应头

3. 响应状态码

常见应用

1. 向浏览器输出信息

2. 下载/上传文件

(1) 获取下载路径.　(2). 下载文件名.

(3) 让浏览器支持下载文件　(4) 获取下载文件的输入流

(5) 创建缓冲区.　(6) 获取 OutputStream 对象

(7) 将 FileOutputStream 写入到 buffer 缓冲区

(8) 使用 OutputStream 将缓冲区中的数据输出到客户端

Header: Content-Disposition: attachment; filename = xxx

日期: /

Header: refresh 用来设置几秒刷新一次

Response 实现登录功

设置响应头

```
setContentType("image/jpeg") → 设置格式为图片
setDataHeader("expires", -1) → 设置不使用缓存
setHeader("Cache-Control", "no-cache")
setHeader("Pragma", "no-cache")
```

将图片用 ImageIO 写入 HttpServletResponse

```
ImageIO.write(image, "jpg", response.getOutputStream())
```

Response 重定向

```
response.sendRedirect("/hello")
```

在工程中新建模块 (补)

在父工程中的 pom.xml 文件中，<modules> 标签中会新增 <module>

如果想设置 module 的访问路径，可以在配置"服务器启动"处配置

日期: /

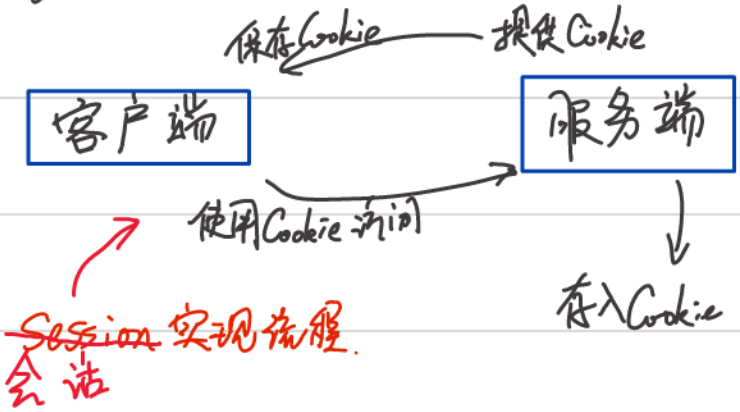
Request 应用

HttpServletRequest代表客户端的请求，用户通过Http协议访问服务器，Http中所有的信息会被封装到这个类中，并通过这个类的方法获得所有的信息

主要功能：

1. 获得传入的参数
2. 请求转发

Cookie 讲解




```
Cookie[] c = request.getCookies();
response.addCookie(new Cookie("a","b"));
```

一个浏览器对同一个站点默认存放30个Cookie，浏览器整体上限在300个左右，Cookie大小也有限制 大约为4kb. 将Cookie的有效时间为0 = 删除Cookie.

日期: /

Session

服务器会给每个用户创建一个 Session 对象.

常用方法

SetAttribute	存入对象.
getId	获取编码.
isnew	判断是否创建过.
removeAttribute	移除对象.
invalidate	注销 Session.

在 web.xml 设置

<session-config> → <session-timeout> 可以设置有效时间(分钟)

Cookie 和 Session 的区别

Cookie 由用户保存. Session 由服务器保存.

Session 由服务创建.

日期: /

Jsp.

Java Server Pages：Java 服务器端页面. 用于动态 WEB 技术.

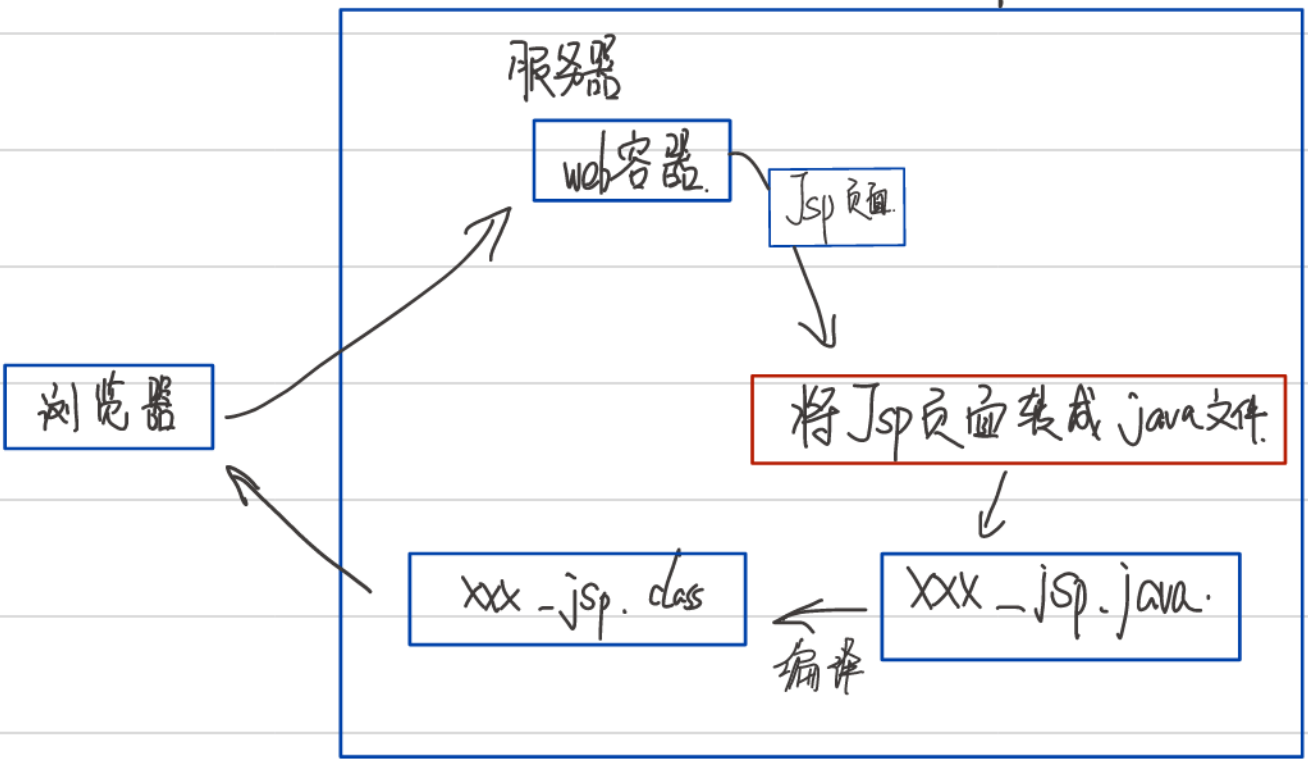
Jsp 可以内嵌 Java 代码.

浏览器向服务器发送请求. 一定会生成一个 Servlet.

Jsp 本质上就是 Servlet.

<% Java代码 %>

<%= param %>



日期: /

Jsp基础语法

<% -- 注释 --%>

<% = param %>. 直接输出 param 的值. (el表达式: ${param})

<% java代码 %> 内嵌 Java 代码.

<%! 外部代码 %> 用来创建全局变量或静态代码块.

<% @ include/page/taglib %> 指定.

例: 设置错误页面.

<% page errorpage = "error/500.jsp" %> 设置错误后的500页面.

web.xml:

```
<error-page>
    <error-code> 404 </error-code>
    <location> /error/404.jsp </location>
<error-page>
```

例: footer 和 header.

<%@ include file = "common/footer.jsp" %>.

日期: /

Jsp 九大内置对象.

PageContext Request Response Session Application (ServletContext)

(ServletConfig) Config out page exception

Jsp 标签.

<jsp :include page="..."> 结果类似于 <%@ include %>. 不同的是实现方式.

<jsp :forward page="" > 页面重定向.

```
<jsp :forward page="common/page.jsp">
        <jsp :param name="name" value="Hashqi" > </jsp:param>
</jsp:forward>
```

(剩余待补充)

日期: /

JavaBean.

特点: 1. 有一个无参构造.
2. 所有属性私有化
3. 有对应的 geter/seter

JavaBean 一般用作和数据库的字段作映射 ORM.

过滤器

chain.doFilter(request, response);

Filter: 1. 导包 2. 继承并重写 Filter 类中的方法
3. 注册过滤器并设置过滤条件

```
<filter>
    <filter-name> AFilter </filter-name>
    <filter-class> com.AFilter </filter-class>
</filter>
<filter-mapping>
    <filter-name> AFilter </filter-name>
    <url-pattern> /servlet/* </url-pattern>
</filter-mapping>
```

/servlet/* : servlet下的所有请求都会经过该过滤器

注: Filter 会随着 Tomcat 容器启动 (init)

javax.servlet.Filter

日期: /

监听器

例. HttpSessionListener

这是一个可以监听 Session 的监听器接口，实现这个接口就可以监听创建 Session，销毁 Session 等事件.

注册监听器.

```
<listener>
    <listener-class> com.listener.MyListener </listener-class>
</listener>
```

A 原子性 Atomicity
C 一致性 Consistency
I 隔离性 Isolation
D 持久性 Durability

保证数据安全

日期：　　/

日期：　　/