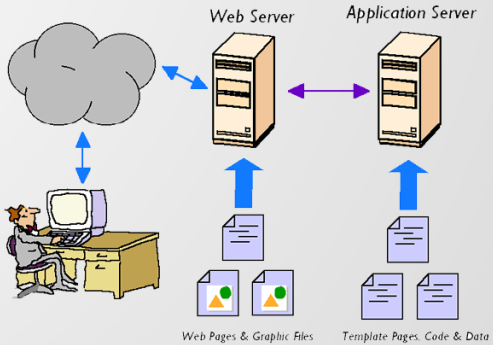
java.server

目录

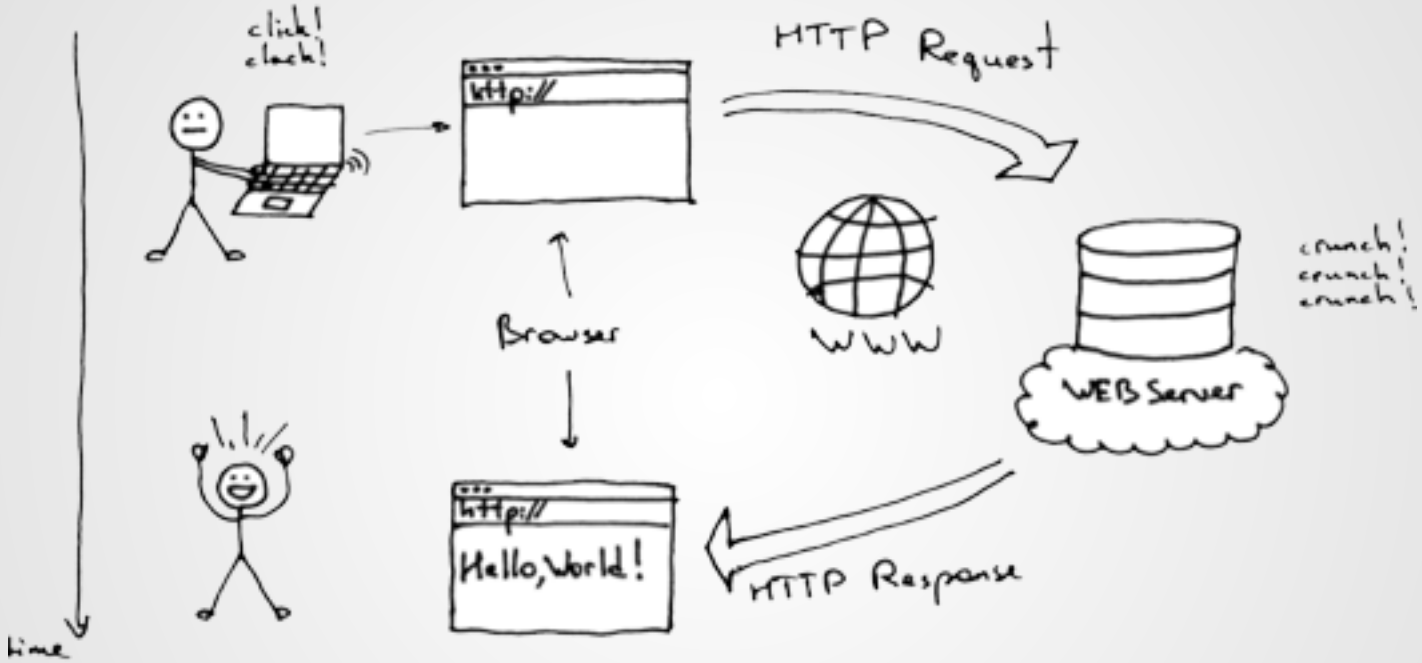
简介

上网浏览网页，离不开服务器，客户请求页面，服务器响应内容，响应的内容是根据每个web请求来产生动态内容的，其内部即启动多个线程来产生不同内容。这种请求响应式的交互，都是基于HTTP协议的。

当然现在随着业务越来越多样化，web服务器变得复杂了，拥有了像缓存、安全和session管理这些附加功能。



简介



web请求都是使用 Request 和Response 式的交流

反射

反射Reflection: 把java类中的各种结构(方法、属性、构造器、类名)映射成一个个的Java对象。利用反射技术可以对一个类进行解剖，反射是框架设计的灵魂。

```
//在运行期间，一个类，只有一个Class对象产生。
1、源头: 获取class对象
Class clz =Class.forName(“com.shsxt.Student")
2、创建对象:
com.shsxt.Student stu
=(com.shsxt.Student)clz.newInstance();
```

XML解析

XML：Extensible Markup Language，可扩展标记语言，作为数据的一种存储格式或用于存储软件的参数，程序解析此配置文件，就可以到达不修改代码就能更改程序的目的。

```
<?xml version="1.0" encoding="UTF-8" ?>
<persons>
   <person>
     <name>至尊宝</name>
     <age>9000</age>
   </person>
   <person>
     <name>白晶晶</name>
     <age>7000</age>
   </person>
</persons>
```

```
//SAX解析
//1、获取解析工厂
SAXParserFactory factory=SAXParserFactory.newInstance();
//2、从解析工厂获取解析器
SAXParser parse =factory.newSAXParser();
//3、加载文档 Document 注册处理器
//4、编写处理器
PersonHandler handler=new PersonHandler();
parse.parse(Thread.currentThread().getContextClassLoader()
.getResourceAsStream("person.xml")
,handler );
```

XML解析

```
<?xml version="1.0" encoding="UTF-8"?>
 <web-app>
 <servlet>
  <servlet-name>login</servlet-name>
  <servlet-class>com.shsxt.LoginServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>login</servlet-name>
  <url-pattern>/login</url-pattern>
  <url-pattern>/g</url-pattern>
 </servlet-mapping>
  <servlet>
  <servlet-name>reg</servlet-name>
  <servlet-class>com.shsxt.RegisterServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>reg</servlet-name>
  <url-pattern>/reg</url-pattern>
 </servlet-mapping>
 </web-app>
```

解析服务器的配置文件,
加入反射获取对象

HTML

HyperText Markup Language : 超文本标记语言，简单理解为浏览器使用的语言。

- 固定结构

```
<html>----开始标签
<head> 网页上的控制信息 <title>页面标题
</title></head>
<body>页面显示的内容</body>
</html>----结束标签
```

- 常用标签

```
<form method="post" action="http://localhost:8888/index.html">
        用户名:<input type="text" name="uname" id="uname"/>
        密码:<input type="password" name="pwd" id="pwd"/>
        <input type="submit" value="登录"/>
</form>
```

- h1~h6
- p
- div
- span
- form
- input
- …

HTTP协议

超文本传输协议（HTTP，HyperText Transfer Protocol)是互联网上应用最为广泛的一种网络协议，所有的WWW文件都必须遵守这个标准。

- 请求协议

1、请求行:方法（GET/POST）、*URI*、协议/版本
2、请求头:*(Request Header)*
3、请求正文:

- 响应协议

1、状态行:协议/版本 状态码 状态描述
2、响应头(Response Header)
3、响应正文:

HTTP请求协议

典型GET的请求协议：

1、请求行
GET /index.html? name=test&pwd=123456 HTTP /1.1
2、请求体
Accept: text/html, application/xhtml+xml, * / *
Accept-Language: zh-CN
User-
Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
Accept-Encoding: gzip, deflate
Host: localhost
Connection: Keep-Alive
3、请求正文

典型POST的请求协议：

1、请求行 POST /index.html HTTP/1.1
2、请求体
Accept: text/html, application/xhtml+xml, * / *
Accept-Language: zh-CN
User-
Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
Accept-Encoding: gzip, deflate
Host: localhost
Connection: Keep-Alive
3、请求正文

name=test&pwd=123456

HTTP响应协议

典型的响应协议:

1、状态行:HTTP/1.0 200 OK
2、请求头:
Date:Mon,31Dec209904:25:57GMT
Server:shsxt Server/0.0.1;charset=GBK
Content-type:text/html
Content-length:39725426
3、请求正文（注意与请求头之间有个空行）

xxxxxx

状态码	说明
- 1xx	指示信息—表示请求已接收，继续处理。
- 2xx	成功—表示请求已经被成功接收、理解、接受。 如:200 OK 客户端请求成功
- 3xx	重定向—要完成请求必须进行更进一步的操作。
- 4xx	客户端错误—请求有语法错误或请求无法实现。 如404，Not Found 请求的资源不存在，例如，输入了错误的URL。
- 5xx:	服务器端错误—服务器未能实现合法的请求。

手写服务器

获取请求协议

1、创建ServerSocket
2、建立连接获取Socket
3、通过输入流获取请求协议
注意： GET与POST不一致的地方

手写服务器

返回响应协议

1、准备内容
2、获取字节数的长度
3、拼接响应协议
注意：空格与换行
4、使用输出流输出

手写服务器

封装响应信息

Response
1、动态添加内容print
2、累加字节数的长度
3、根据状态码拼接响应头协议
4、根据状态码统一推送出去

调用处: 动态调用print +传入状态码推送

手写服务器

封装请求信息

Request
通过分解字符串获取method URL和请求参数 POST请求参数可能在 请求体中还存在

手写服务器

处理请求参数

Request
通过Map封装请求参数 两个方法
考虑一个参数多个值和中文

手写服务器

引入SERVLET

Sevlet
将业务代码解耦到对应的业务类中(具体的Serlvet)

手写服务器

整合配置文件

整合配置文件
根据配置文件动态的读取类名，再进行反射获取具体的Servlet来处理业务，真正的以不变应万变

手写服务器

封装分发器

Dispatcher
加入了多线程，可以同时处理多个请求，
使用的是短连接

手写服务器

404及首页处理

读取错误、首页内容即可

感谢您的支持与信任

THANK YOU FOR WATCHING