

钱进培训是哈法地区资深工程师组成的培训机构，通过各位老师的现身说法，帮助各位学员迅速掌握实战知识，为求职打下坚实的基础。电子邮件：jin.qian.canada@gmail.com 钱老师报名、答疑微信号：qianjincanada，或扫描以下二维码添加：



钱老师 
加拿大



Java高级速成班

本课程针对有一定编程基础，希望在短时间内对Java企业级开发有所了解的同学。

通过本课程的学习，学员能够顺利掌握J2EE的体系结构，了解常见的Java框架并熟练使用，具体内容包括：

1. B/s体系结构，HTTP协议栈相关内容（HTML，CSS，JS）
2. 数据库访问的不同方法（Hibernate使用）
3. Spring IOC在企业开发中的应用
4. Restful API/SOAP开发及应用
5. SVN/GIT/MAVEN的应用



假设现在我想查询id为2的那本书的书名,使用session.get(...)方法:

```
1 Session session=HibernateUtil.getSession();
```

```
2 Book book =(Book) session.get(Book.class,2);
```

```
3 System.out.println(book.getName());
```

执行完第二行代码,还未执行第三行时,控制台已经打印出了sql语句,执行第三行时打印出书名"斗破苍穹".

id	author	name	price	pubDate	category_id
1	金庸	鹿鼎记	12.35	2016-03-06 09:25:41	1
2	土豆	斗破苍穹	22.35	2016-03-06 09:25:41	2
3	番茄	吞噬星空	27.35	2016-03-06 09:25:41	2
4	都梁	亮剑	42.35	2016-03-06 09:25:41	3
5	耳根	邪气凛然	12.35	2016-03-06 09:25:41	4

Book表

id	name
1	武侠类
2	玄幻类
3	历史类
4	都市类

category表

而如果使用session.load(..)查询时:

```
1 Session session=HibernateUtil.getSession();
```

```
2 Book book =(Book) session.load(Book.class,1);
```

```
3 System.out.println(book.getName());
```

执行完第二行代码还未执行第三行时,控制台什么都没有打印,执行第三行时,控制台打印出sql语句和书名"斗破苍穹".

id	author	name	price	pubDate	category_id
1	金庸	鹿鼎记	12.35	2016-03-06 09:25:41	1
2	土豆	斗破苍穹	22.35	2016-03-06 09:25:41	2
3	番茄	吞噬星空	27.35	2016-03-06 09:25:41	2
4	都梁	亮剑	42.35	2016-03-06 09:25:41	3
5	耳根	邪气凛然	12.35	2016-03-06 09:25:41	4

Book表

id	name
1	武侠类
2	玄幻类
3	历史类
4	都市类

category表

使用get方法查询时,程序立即去访问数据库(实际上是先去一级缓存session中查询,没有发现的话再去二级缓存,再没有的话才去访问数据库),得到id=2的Book,并且打印出sql语句,而是用load方法查询时,load并未立即去访问数据库,他先是返回了一个Book的代理对象,当你真正要用到Book中信息时,才去访问数据库.load支持延迟加载,get不支持延迟加载,当然如果设置了lazy=false,get和load都会直接去访问数据库,都变成即时加载.

get/load方法还有一个很重要的区别就是:

load方式检索不到的话会抛出
org.hibernate.ObjectNotFoundException异常

get方法检索不到的话会返回null

即时加载和延时加载的概念,通俗的说,即时加载,就是立即去数据库查找,延迟加载,就是真正需要的时候才去数据库查找,这类似于单例模式中的懒汉式和饿汉式的加载方式.

假设我现在想通过查询Book,来得到Book所对应的Category,如果设置为即时加载,当加载Book时,会自动加载Category,如果设置为延迟加载,则加载Book时,不会加载Category,只有当第一次调用getCategory(),时,才去执行sql语句,加载Category.

一般来说,延迟加载要比即时加载节省资源,但是如果处理不当,延迟加载容易抛出延迟加载异常(LazyInitializationException).这是因为延迟加载时,只有第一次调用getCategory()时才会加载Category数据,如果这时候数据库连接已经关闭了,就会因为无法加载数据而抛出异常.

第四讲：Web相关

Apache http server

Tomcat

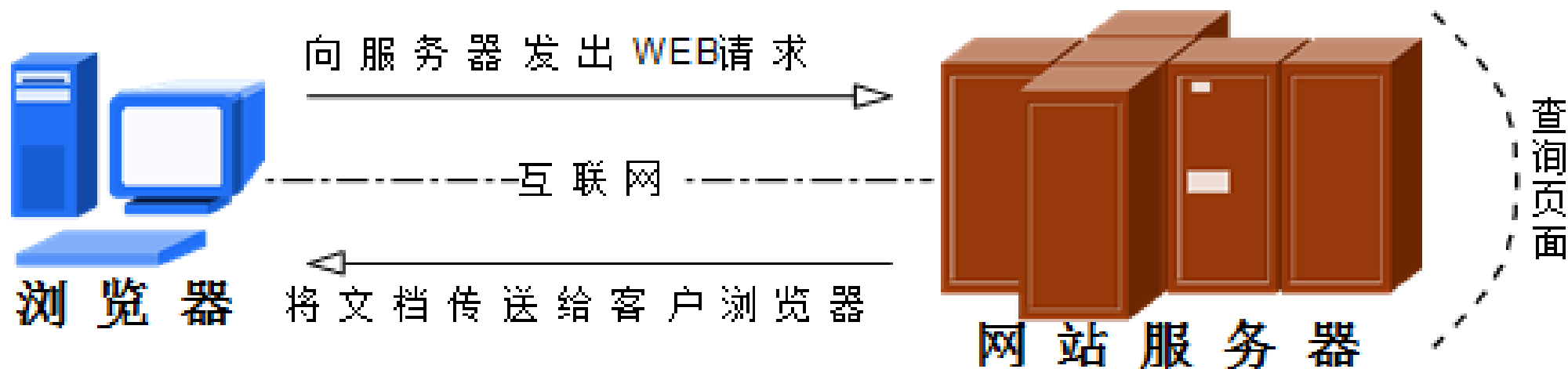
Servlet

JSP

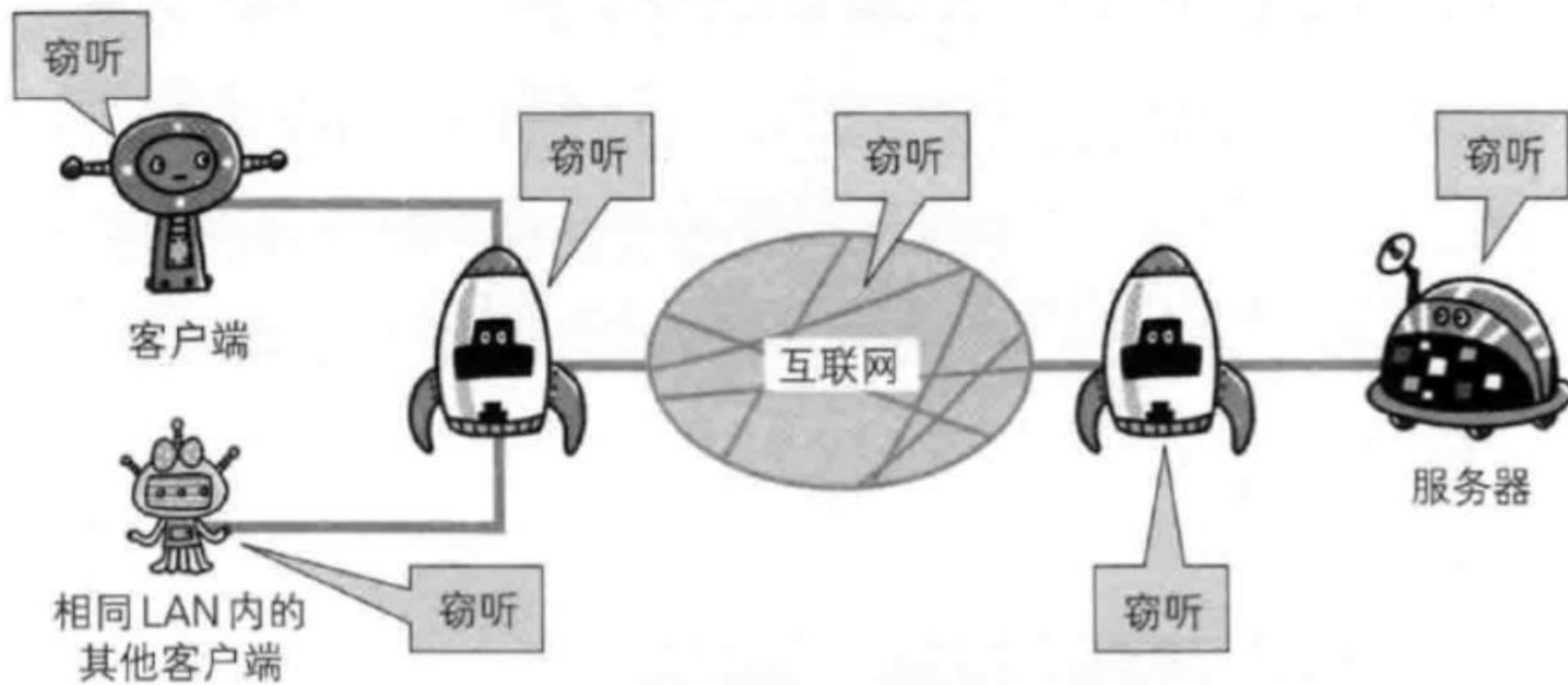
最初在1970年前后，互联网技术仅仅是处在雏形阶段的一种构想，经过多年发展后才慢慢逐步向非军方或科研部门外开始接入使用，虽然那时“互联网”的规模还不如现在局域网成熟，但依然为网络应用技术爆发式增长打下了扎实的基础。

我们平时访问的网站服务就是Web网络服务也叫WWW万维网(World Wide Web)，一般是指能够通过让用户通过浏览器访问到互联网中文档等资源的服务。

如图所示，Web网站服务是一种被动访问的服务程序，即只有接收到互联网中其他计算机发出的请求后才会响应，最终Web服务器会通过HTTP(超文本传输协议)或HTTPS（超文本安全传输协议）把指定文件传送到客户机的浏览器上。



http的缺点



HTTP+加密+认证+完整性保护 = HTTPS.

为了解决上述问题,需要再http上再加入加密处理和认证等机制.我们把添加了加密和认证机制的http称之为https(http secure)

公开密钥加密(public-key cryptography)->加密算法是公开的,而密钥是保密的.加密和解密都会用到密钥.但反过来说只要持有密钥就能解密.

共享密钥加密(common key crypto system)->也被叫做对密钥加密.以共享密钥方式加密时必须将密钥也发给对方.在互联网上转发密钥时,若果通信被监听那么密钥就会落入攻击者之手,另外还得设法安全的保管接收到的密钥.

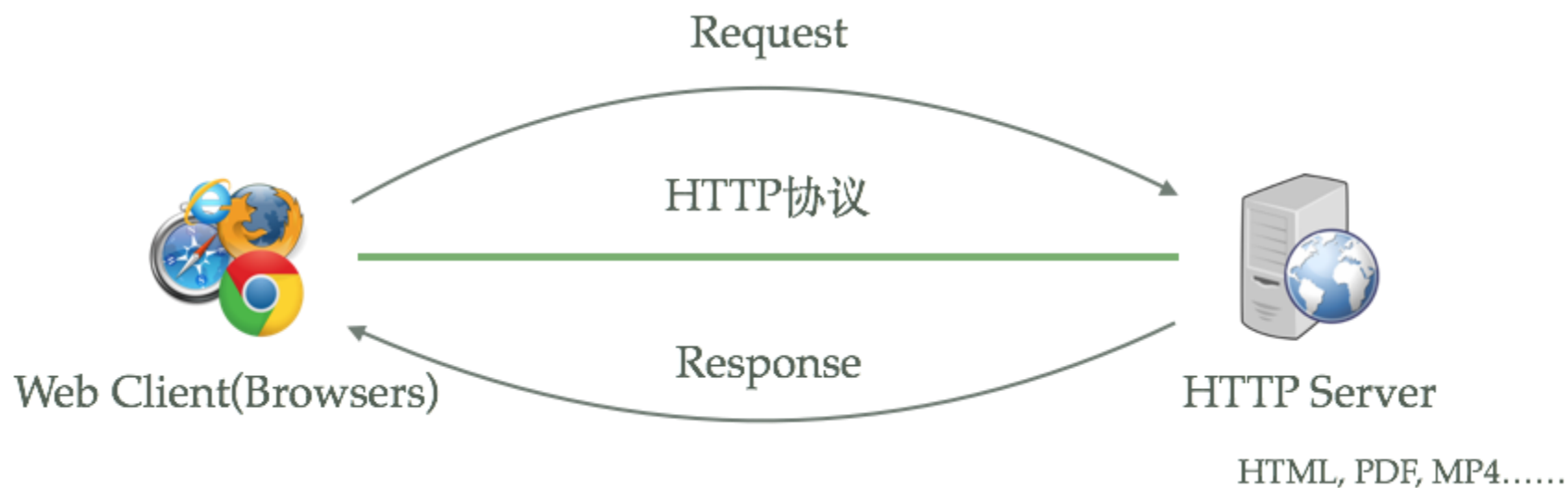
目前能够提供WEB网络服务的程序有Apache、Nginx或IIS等等，在Windows系统中默认Web服务程序是IIS互联网信息服务(Internet Information Services)，这是一款图形化的网站管理工具，IIS程序不光能提供Web网站服务，还能够提供FTP、NMTP、SMTP等服务功能，但只能在Windows系统中使用。

2004年10月4日为俄罗斯知名门户网站而开发的Web服务程序Nginx出世了，Nginx程序作为一款轻量级的网站服务软件，因其稳定性和丰富的功能而快速占领服务器市场，但最最最被认可的还当属是低系统资源、占用内存少且并发能力强，目前国内如新浪、网易、腾讯等门户网站均使用。

Apache程序是目前拥有很高市场占有率的Web服务程序之一，其跨平台和安全性广泛被认可且拥有快速、可靠、简单的API扩展，Apache的名字取自美国印第安人土著语，寓意着拥有高超的作战策略和无穷的耐性。

Apache服务程序可以运行在Linux系统、Unix系统甚至是Windows系统中，支持基于IP、域名及端口号的虚拟主机功能、支持多种HTTP认证方式、集成有代理服务器模块、安全Socket层(SSL)、能够实时监视服务状态与定制日志消息，并有着各类丰富的模块支持。

HTTP服务器本质上也是一种应用程序——它通常运行在服务器之上，绑定服务器的IP地址并监听某一个tcp端口来接收并处理HTTP请求，这样客户端（一般来说是IE, Firefox, Chrome这样的浏览器）就能够通过HTTP协议来获取服务器上的网页（HTML格式）、文档（PDF格式）、音频（MP4格式）、视频（MOV格式）等等资源。



apache 配置文件位于 `/etc/httpd/conf`，主要的配置文件是 `/etc/httpd/conf/httpd.conf`，此文件会引用其它文件。

用默认配置可以启动一个简单的服务，有用户访问时会提供目录 `/srv/http` 下的内容。

启动 `httpd.service` `systemd` 服务，Apache 就会启动，从浏览器中访问 `http://localhost/` 会显示一个简单的索引页面

TOMCAT目录下放几个HTML文件试试

Tomcat

一、静态web页面：

- 1、在静态Web程序中，客户端使用Web浏览器（IE、FireFox等）经过网络(Network)连接到服务器上，使用HTTP协议发起一个请求（Request），告诉服务器我现在需要得到哪个页面，所有的请求交给Web服务器，之后WEB服务器根据用户的需要，从文件系统（存放了所有静态页面的磁盘）取出内容。之后通过Web服务器返回给客户端，客户端接收到内容之后经过浏览器渲染解析，得到显示的效果。
- 2、为了让静态web页面显示更加好看，使用javascript / VBScript / ajax（AJAX即“**Asynchronous Javascript And XML**”（异步JavaScript和XML），是指一种创建交互式网页应用的网页开发技术。）但是这些特效都是在客户端上借助于浏览器展现给用户的，所以在服务器上本身并没有任何的变化。
- 3、静态web无法连接数据库；
- 4、静态web资源开发技术：HTML；
- 5、由于现在的web页面中，大量使用JS，导致浏览器打开页面，就会占用大量的内存，服务端的压力是减轻了，但压力转移到了客户端。

二、动态web页面：

动态WEB中，程序依然使用客户端和服务端，客户端依然使用浏览器（IE、FireFox等），通过网络(Network)连接到服务器上，使用HTTP协议发起请求（Request），现在的所有请求都先经过一个WEB Server来处理。

如果客户端请求的是静态资源(*.htm或者是*.html)，则将请求直接转交给WEB服务器，之后WEB服务器从文件系统中取出内容，发送回客户端浏览器进行解析执行。

如果客户端请求的是动态资源（*.jsp、*.asp/*.aspx、*.php），则先将请求转交给WEB Container(WEB容器)，在WEB Container中连接数据库，从数据库中取出数据等一系列操作后动态拼凑页面的展示内容，拼凑页面的展示内容后，把所有的展示内容交给WEB服务器，之后通过WEB服务器将内容发送回客户端浏览器进行解析执行。

Client

Network

HTTP Request

Server

Web Browser

Web Server Plugin

静态
请求
资源

动态
请求
资源

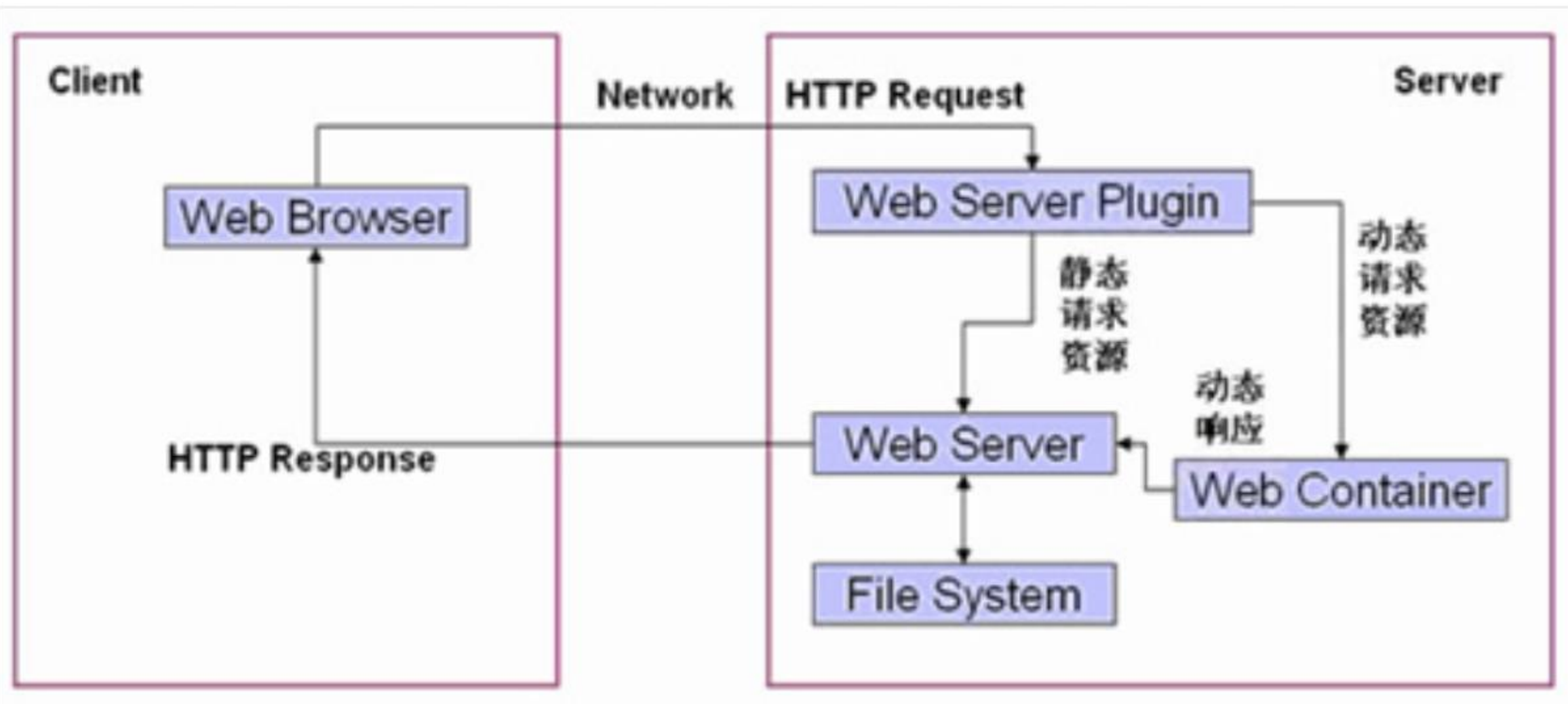
HTTP Response

Web Server

动态
响应

Web Container

File System



1、静态页面就是设计者把页面上所有东西都设定好、做死了，然后放上去，不管是谁在任何时候看到的页面内容都是一样的，一成不变（除非手动修改页面内容）。静态html页面文件，可以直接用本地的浏览器打开。比如：

file:///Users/Phil/Documents/DevOps/HBuilderProjects/testJSP/index.html。

2、动态页面的内容一般都是依靠服务器端的程序来生成的，不同人、不同时候访问页面，显示的内容都可能不同。网页设计者在写好服务器端的页面程序后，不需要手工控制，页面内容会按照页面程序的安排自动更改变换。

3、html是w3c规范的一种网页书写格式，是一种统一协议语言，静态网页。我们上网看的网页都是大部分都是基于html语言的。jsp是一种基于动态语言，jsp可以实现html的所有任务

4、HTML（Hypertext Markup Language）文本标记语言，它是静态页面，和JavaScript一样解释性语言，为什么说是解释性语言呢？因为，只要你有一个浏览器那么它就可以正常显示出来，而不需要指定的编译工具，只需在TXT文档中写上HTML标记就可以正常显示。

JSP（Java Server Page）是Java服务端的页面，所以它是动态的，它是需要经过JDK编译后把内容发给客户端去显示，我们都知道，Java文件编译后会产生一个class文件，最终执行的就是这个class文件

5、JSP的前身是servlet

6、html和jsp的表头不一样，这个是JSP的头“<%@ page language="java" import="java.util.*" pageEncoding="gbk"%>”在表头中有编码格式和倒入包等。也是很好区分的，在jsp中用<%%>就可以写Java代码了，而html没有<%%>。

7、，不认识jsp或者asp什么的，但是有时候界面需要逻辑控制，所以我们就用相应的技术来实现，这样比较方便。而jsp在后台通过服务器解析为相应的html，然后在供浏览器识别显示。例如

```
<%  
if(flag == a){  
<label>a<label>  
}else {  
<label>b<label>  
}  
%>
```

服务器在读取到这段代码后，根据相应的业务逻辑，编译成相应的servlet，再由servlet输出到页面（输出的就是html）。

JSP（全称Java Server Pages）是由 Sun Microsystems 公司倡导和许多公司参与共同创建的一种使软件开发者可以响应客户端请求，而动态生成 HTML、XML 或其他格式文档的Web网页的技术标准。

JSP 技术是以 Java 语言作为脚本语言的，JSP 网页为整个服务器端的 Java 库单元提供了一个接口来服务于HTTP的应用程序。

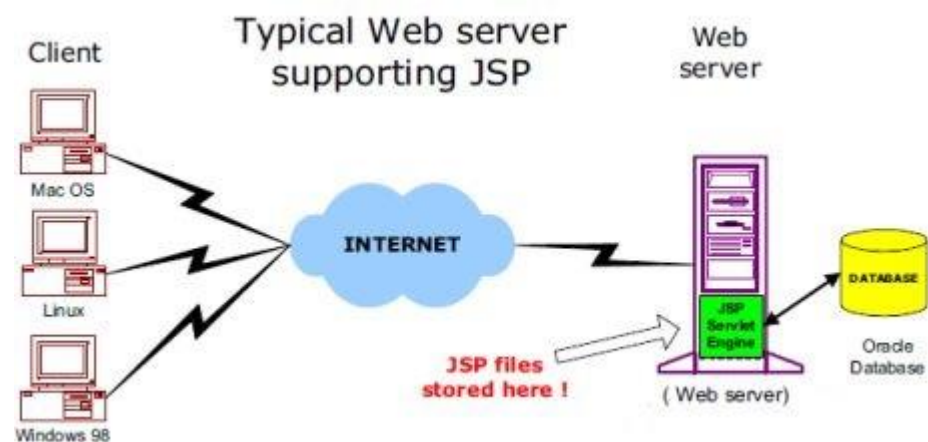
JSP文件后缀名为 *.jsp 。

JSP开发的WEB应用可以跨平台使用，既可以运行在 Linux 上也能运行在 Windows 上。

```
<html>
  <head>
    <title>第一个 JSP 程序</title>
  </head>
  <body>
    <%
      out.println("Hello World ! ");
    %>
  </body>
</html>
```

网络服务器需要一个JSP引擎，也就是一个容器来处理JSP页面。容器负责截获对JSP页面的请求。JSP容器与Web服务器协同合作，为JSP的正常运行提供必要的运行环境和其他服务，并且能够正确识别专属于JSP网页的特殊元素。

下图显示了JSP容器和JSP文件在Web应用中所处的位置。



以下步骤表明了Web服务器是如何使用JSP来创建网页的：

就像其他普通的网页一样，浏览器发送一个HTTP请求给服务器。

Web服务器识别出这是一个对JSP网页的请求，并且将该请求传递给JSP引擎。通过使用URL或者.jsp文件来完成。

JSP引擎从磁盘中载入JSP文件，然后将它们转化为servlet。这种转化只是简单地将所有模板文本改用println()语句，并且将所有的JSP元素转化成Java代码。

JSP引擎将servlet编译成可执行类，并且将原始请求传递给servlet引擎。

Web服务器的某组件将会调用servlet引擎，然后载入并执行servlet类。在执行过程中，servlet产生HTML格式的输出并将其内嵌于HTTP response中上交给Web服务器。

Web服务器以静态HTML网页的形式将HTTP response返回到您的浏览器中。

最终，Web浏览器处理HTTP response中动态产生的HTML网页，就好像在处理静态网页一样。

JSP 内置对象

JSP有9个隐式对象，这些对象由Web容器创建，可用于所有jsp页面。

可用的隐式对象有out, request, config, session, application等。

```
<html>
```

```
<body>
```

```
<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
```

```
</body>
```

```
</html>
```

JSP请求是HttpServletRequest类型的隐式对象，即由web容器为每个jsp请求创建的。它可以用于获取参数，头信息，远程地址，服务器名称，服务器端口，内容类型，字符编码等请求信息。它也可以用于从jsp请求范围设置，获取和删除属性。

Test1.html

```
<form action="test1.jsp">  
<input type="text" name="uname">  
<input type="submit" value="go"><br/>  
</form>
```

Test1.jsp

```
<%  
String name=request.getParameter("uname");  
out.print("welcome "+name);  
%>
```

在JSP中，response是HttpServletResponse类型的隐式对象。HttpServletResponse的实例由Web容器为每个jsp请求创建。它可以用于添加或操纵响应，如重定向响应到其他资源，发送错误等。

Test2.html

```
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
```

Test2.jsp

```
<%
response.sendRedirect("http://www.google.com");
%>
```

JSP Session内置对象

在JSP中，`session`是`HttpSession`类型的隐式对象。Java开发人员可以使用此对象来设置，获取或删除属性或获取会话信息。

Test3.html

```
<html>
```

```
<body>
```

```
<form action="test3.jsp">
```

```
<input type="text" name="uname">
```

```
<input type="submit" value="go"><br/>
```

```
</form>
```

```
</body>
```

```
</html>
```

```
<html>
```

```
<body>
```

```
<%
```

```
String name=request.getParameter("uname");
```

```
out.print("Welcome "+name);
```

```
session.setAttribute("user",name);
```

```
<a href="test4.jsp">second jsp page</a>
```

```
%>
```

```
</body>
```

```
</html>
```

Test3.jsp

```
<html>
```

```
<%
```

```
String name=(String)session.getAttribute("user");
```

```
out.print("Hello "+name);
```

```
%>
```

Test4.jsp

SP指令

JSP指令

jsp指令是告诉Web容器如何将JSP页面翻译成相应的servlet的消息。

有三种类型的指令：

页面指令

include指令

taglib伪指令

JSP page指令

page指令定义适用于整个JSP页面的属性。

JSP page指令的语法

```
<%@ page attribute="value" %>
```

JSP page指令的属性

import

contentType

extends

info

buffer

language

isELIgnored

isThreadSafe

autoFlush

session

pageEncoding

errorPage

isErrorPage

Web服务器的基本功能就是提供Web信息浏览服务。它只需支持HTTP协议、HTML文档格式及URL。与客户端的网络浏览器配合。因为Web服务器主要支持的协议就是HTTP，所以通常情况下HTTP服务器和WEB服务器是相等的

上世纪90年代，随着Internet和浏览器的飞速发展，基于浏览器的B/S模式随之火爆发展起来。

最初，用户使用浏览器向WEB服务器发送的请求都是请求静态的资源，比如html、css等。但是可以想象：根据用户请求的不同动态的处理并返回资源是理所当然必须的要求。

WEB服务器接收一个用户请求；

WEB服务器将请求转交给WEB服务器关联的Servlet容器；

Servlet容器找到对应的Servlet并执行这个Servlet；

Servlet容器将处理结果返回给WEB服务器；

WEB服务器把结果送回用户；

Servlet有两种意思：

广义上是：基于Java技术的Web组件，被容器托管，用于生成动态内容。

再详细点说，Servlet是JavaEE组件中的 -> Web组件的 -> 一种。

（其它两种是JavaServer Faces和JavaServer Page ）

狭义上说：是JavaEE API中的一个interface , `javax.servlet.Servlet` ；

Servlet 容器/引擎:

Servlet容器也可以叫引擎， Container/Engine， 用于执行Servlet。

容器是以内嵌或者附加组件的形式存在于Web服务器或者应用服务器中的。

容器本身（不依赖Web服务器）就提供了基于请求/响应发送模型的网络服务，解码基于MIME的请求，格式化基于MIME的响应。

所有容器必须实现HTTP协议的请求/响应模型。其它协议不强求，如HTTPS。

Servlet接口只定义了一个服务方法就是service，而HttpServlet类实现了该方法并且要求调用下列的方法之一：

doGet：处理GET请求

doPost：处理POST请求

当发出客户端请求的时候，调用service方法并传递一个请求和响应对象。Servlet首先判断该请求是GET操作还是POST操作。然后它调用下面的一个方法：doGet或doPost。如果请求是GET就调用doGet方法，如果请求是POST就调用doPost方法。doGet和doPost都接受请求(HttpServletRequest)和响应(HttpServletResponse)。

get和post这是http协议的两种方法，另外还有head, delete等

这两种方法有本质的区别，get只有一个流，参数附加在url后，大小个数有严格限制且只能是字符串。post的参数是通过另外的流传递的，不通过url，所以可以很大，也可以传递二进制数据，如文件的上传。

在servlet开发中，以doGet()和doPost()分别处理get和post方法。

另外还有一个doService(), 它是一个调度方法，当一个请求发生时，首先执行doService(), 不管是get还是post。在HttpServlet这个基类中实现了一个角度，首先判断是请求时get还是post, 如果是get就调用doGet(), 如果是post就调用doPost()。你也可以直接过载doService()方法，这样你可以不管是get还是post。都会执行这个方法。

Web.xml

```
<?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/javaee"  
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
```

```
<servlet>
```

```
    <servlet-name>Sta</servlet-name>
```

```
    <servlet-class>ca.commonservice.statistics.CommonStatistics</servlet-class>
```

```
</servlet>
```

```
<servlet-mapping>
```

```
    <servlet-name>Sta</servlet-name>
```

```
    <url-pattern>/sta</url-pattern>
```

```
</servlet-mapping>
```


WAR(Web Archive file)网络应用程序文件，是与平台无关的文件格式，它允许将许多文件组合成一个压缩文件。**war**专用在**web**方面。

JAVA WEB工程，都是打成WAR包进行发布。

典型的war包内部结构如下：

```
webapp.war
|  index.jsp
|
|— images
|— META-INF
|— WEB-INF
    |  web.xml          // WAR包的描述文件
    |
    |— classes
    |    action.class  // java类文件
    |
    |— lib
        other.jar      // 依赖的jar包
        share.jar
```



```
public class HelloWorld extends HttpServlet {  
    private String message;  
    public void init() throws ServletException  
    {  
message = "Hello World";  
    }  
    public void doGet(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException  
    {  
response.setContentType("text/html");  
PrintWriter out = response.getWriter();  
    out.println("<h1>" + message + "</h1>");  
    }  
    public void destroy()  
    {  
        // do nothing.  
    }  
}
```


Apache与Tomcat都是Apache开源组织开发的用于处理HTTP服务的项目，两者都是免费的，都可以做为独立的Web服务器运行。Apache是Web服务器而Tomcat是Java应用服务器。

Apache：是C语言实现的，专门用来提供HTTP服务。

Tomcat：是Java开发的一个符合JavaEE的Servlet规范的JSP服务器（Servlet容器），是Apache的扩展。

特性：免费的Java应用服务器

- 1、主要用于解析JSP/Servlet，侧重于Servlet引擎；
- 2、支持静态页，但效率没有Apache高；支持Servlet、JSP请求；

在市场上有许多 Web 服务器支持 Servlet。有些 Web 服务器是免费下载的，Tomcat 就是其中的一个。

Apache Tomcat 是一款 Java Servlet 和 JavaServer Pages 技术的开源软件实现，可以作为测试 Servlet 的独立服务器，而且可以集成到 Apache Web 服务器。下面是在电脑上安装 Tomcat 的步骤：

从 <http://tomcat.apache.org/> 上下载最新版本的 Tomcat。

一旦您下载了 Tomcat，解压缩到一个方便的位置。例如，如果您使用的是 Windows，则解压缩到 C:\apache-tomcat-5.5.29 中，如果您使用的是 Linux/Unix，则解压缩到 /usr/local/apache-tomcat-5.5.29 中，并创建 CATALINA_HOME 环境变量指向这些位置。

在 Windows 上，可以通过执行下面的命令来启动 Tomcat：

```
%CATALINA_HOME%\bin\startup.bat
```

或者

```
C:\apache-tomcat-5.5.29\bin\startup.bat
```

Tomcat运行在JVM之上，它和HTTP服务器一样，绑定IP地址并监听TCP端口，同时还包含以下职责：管理Servlet程序的生命周期将URL映射到指定的Servlet进行处理与Servlet程序合作处理HTTP请求——根据HTTP请求生成HttpServletResponse对象并传递给Servlet进行处理，将Servlet中的HttpServletResponse对象生成的内容返回给浏览器

Tomcat 启动后，可以通过在浏览器地址栏输入 `http://localhost:8080/` 访问 Tomcat 中的默认应用程序。

Create maven web application

```
<dependency>
```

```
    <groupId>javax.servlet</groupId>
```

```
    <artifactId>javax.servlet-api</artifactId>
```

```
    <version>3.1.0</version>
```

```
</dependency>
```


- springmvcexample
 - .settings
 - src
 - main
 - java
 - com
 - howtodoinjava
 - demo
 - resources
 - webapp
 - test
 - java
 - target
 - .classpath
 - .project
 - pom.xml

target

lassp

rojec

om.x

Refresh

Close Project

Close Unrelated Projects

Validate

Show in Remote Systems view

Profile As

Debug As

Run As

Team

Compare With

Restore from Local History...

Maven

Java EE Tools

Configure

Source

Add Dependency

Add Plugin

New Maven Module Project

Download JavaDoc

Download Sources

Update Project...

Alt+F5

Select Maven Profiles...

Ctrl+Alt+P

Disable Workspace Resolution

Disable Maven Nature

Assign Working Sets...

exe (Feb 3, 2

type filter text

- ▶ Resource
- Builders
- Deployment Assembly
- Java Build Path
- ▶ Java Code Style
- ▶ Java Compiler
- ▶ Java Editor
- Javadoc Location
- ▶ JavaScript
- JSP Fragment
- ▶ Maven
- Project Facets
- Project References
- Run/Debug Settings
- Server
- Service Policies
- Targeted Runtimes
- ▶ Task Repository
- Task Tags
- ▶ Validation
- Web Content Settings
- Web Page Editor
- Web Project Settings
- WikiText
- ▶ XDoclet

Java Build Path

Source Projects Libraries Order and Export

Source folders on build path:

- ▲ springmvcexample/src/main/java
 - Output folder: springmvcexample/target/classes
 - Included: (All)
 - Excluded: (None)
 - Native library location: (None)
 - Ignore optional compile problems: No
- ▲ springmvcexample/src/main/resources
 - Output folder: (Default output folder)
 - Included: **/*.java
 - Excluded: **/*.java
 - Native library location: (None)
 - Ignore optional compile problems: No
- ▲ springmvcexample/src/test/java
 - Output folder: springmvcexample/target/test-classes
 - Included: (All)
 - Excluded: (None)
 - Native library location: (None)
 - Ignore optional compile problems: No

☒ Allow output folders for source folders

Default output folder:

springmvcexample/target/classes

Add Folder...

Link Source...

Edit...

Remove

Browse...

OK

Cancel

apache-tomcat-8.5.12

```
response.setContentType("text/html");  
PrintWriter out = response.getWriter();  
out.println("<h1> this is a servlet</h1>");
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"    pageEncoding="ISO-8859-1"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
```

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
```

```
<title>Hello World - JSP tutorial</title>
```

```
</head>
```

```
<body>
```

```
    <%= "Hello World this is jsp!" %>
```

```
</body>
```

```
</html>
```

```
<!DOCTYPE web-app PUBLIC
"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd" >
```

```
<web-app>
  <display-name>Archetype Created Web Application</display-name>
  <servlet>
    <servlet-name>TestServlet1</servlet-name>
    <display-name>TestServlet1</display-name>
    <description></description>
    <servlet-class>com.test.TestServlet1</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>TestServlet1</servlet-name>
    <url-pattern>/TestServlet1</url-pattern>
  </servlet-mapping>
</web-app>
```


```
<dependency>  
    <groupId>javax.servlet</groupId>  
    <artifactId>javax.servlet-api</artifactId>  
    <version>3.1.0</version>  
</dependency>
```

<http://localhost:8123/testMavenWeb/index.jsp>

<http://localhost:8123/testMavenWeb/TestServlet1>

钱进培训是哈法地区资深工程师组成的培训机构，通过各位老师的现身说法，帮助各位学员迅速掌握实战知识，为求职打下坚实的基础。电子邮件：jin.qian.canada@gmail.com 钱老师报名、答疑微信号：qianjincanada，或扫描以下二维码添加：



钱老师 
加拿大



Java高级速成班

本课程针对有一定编程基础，希望在短时间内对Java企业级开发有所了解的同学。

通过本课程的学习，学员能够顺利掌握J2EE的体系结构，了解常见的Java框架并熟练使用，具体内容包括：

1. B/s体系结构，HTTP协议栈相关内容（HTML，CSS，JS）
2. 数据库访问的不同方法（Hibernate使用）
3. Spring IOC在企业开发中的应用
4. Restful API/SOAP开发及应用
5. SVN/GIT/MAVEN的应用

