1 - Servlet简介 - 文档

1. Java Web开发需要学习什么?

1.1 互联网简介

在即将开始Java Web开发之旅之前，你是否想过Web到底是什么？ 要了解Web，你需要先了解另一个你耳熟能详的概念：互联网(Internet)。

你或许听过互联网+，各种互联网思维，如果从技术视角来审视互联网，如果打开互联网的内部，其内部结构会是什么样子呢?

互联网指的是通过TCP/IP协议族相互连接在一起的计算机的网络。TCP是Transmission Control Protocol，传输控制协议；IP是Internet Protocol，网际协议。TCP/IP协议族是一个网络通讯模型，是当前互联网通讯的基础架构。

- IP用来去识别网络上的一台计算机。计算机要连接到一起相互通信，首先需要知道连接的目标计算机，而IP就能标识一台计算机。做一个类比，我们人跟人之间也需要建立连接才能交流，在一群人中说话，首先喊出一个人的名字，他就知道你在跟他说话了。IP就是计算机的名字。
- TCP是计算机之间控制传输信息的协议，同样的类比，就是人与人之间沟通的语言和方式。一个不会外语的中国人跟一个美国人交流是无效的，就跟好像一台计算机发送目标计算机无法识别的数据包。能够识别出网络上的计算机，同时也能以相互理解的方式进行通讯，这样计算机就可以连接到一起了。

互联网是如何发展起来的呢？战争往往是科技发展的原动力，互联网也不例外。互联网诞生于冷战时期，美国国防部研制的APANET是互联网的原型，那时主要用来传递战争情报。到了1982年，TCP/IP协议被标准化了，Vint Cerf和Robert Kahn将Internet的概念正式提出来了，因此他们也被誉为这个互联网之父。2004他们获得了计算机界的诺贝尔奖——图灵奖，这是计算机界能够获得的最高的终生荣誉。Vint Cerf现在依然是Google的互联网首席科学家。

随后美国自然科学基金会建立了各个大学之间的高速传播的网络NSFNET，后来转为商用。如果把互联网想成是一棵大树，NSFNET构成最重要的几个核心主枝干。后面越来越多的子网或者其他的网络，逐渐的连到这个主干上，从而形成了整个全球化的互联网。

1.2 TCP/IP协议

客户端-服务器（Client-Server）模型

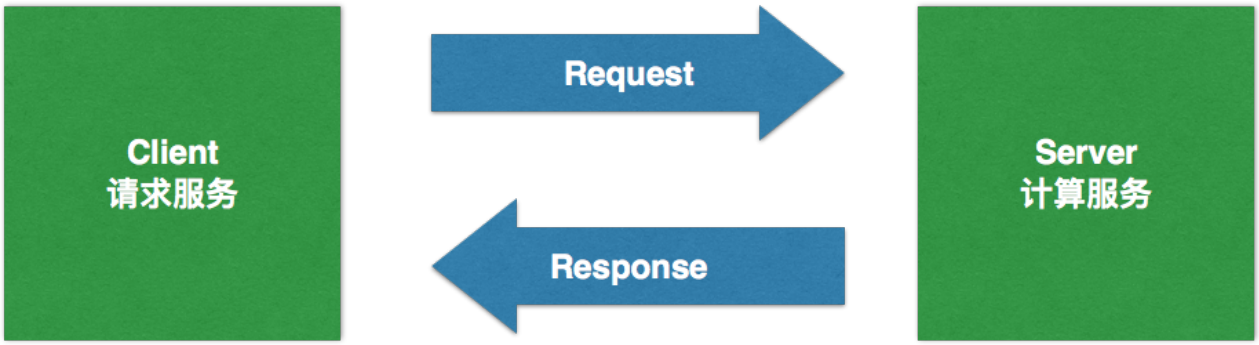
当多台计算机需要相互通信传输数据时，他们必须遵循相同的协议（Protocol），协议可以认为是通信双方遵循的一种约定——在此基础之上，双方才能够相互理解通信的内容。

类比：来自两个不同国家的人必须使用同一种语言才能够进行交流

在计算机科学里，客户端服务器模型是一种非常流行的通信模型，通信的双方被抽象为客户端（Client）和服务器（Server）：

- 客户端：向服务器请求提供服务——例如计算两个整数a, b的和a + b
- 服务器：提供计算服务——例如将客户端请求的两个整数a, b的和a + b的值经过计算后返回客户端

这个模型可以用下图表示：

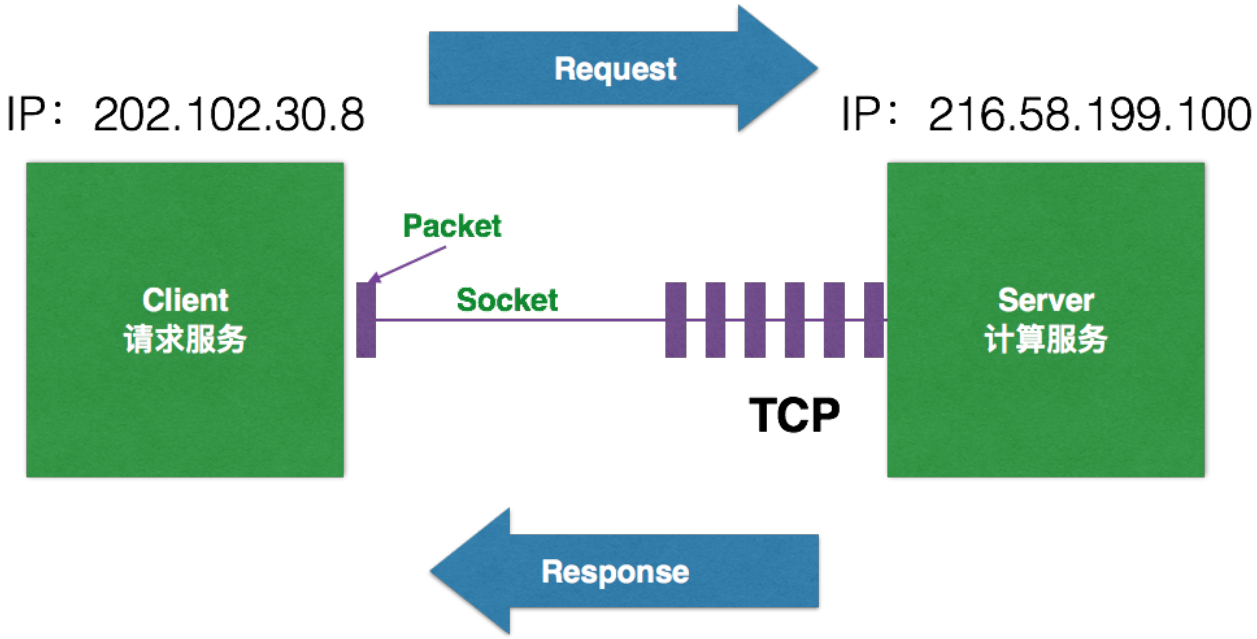


IP地址

在Client-Server模型中，第一个要解决的问题是——客户端向服务器请求服务，服务器的地址在哪里。可能使用过计算机的同学都知道，如果计算机要接入互联网，那么必须拥有一个IP地址。IP地址正是通过IP协议分配给网络中的各个设备。例如Google搜索服务器的地址是：216.58.199.100，那么当我们使用Google时，实际上计算机会在网络上寻找IP地址216.58.199.100并与之进行通信，请求Google为我们提供搜索服务。

TCP协议

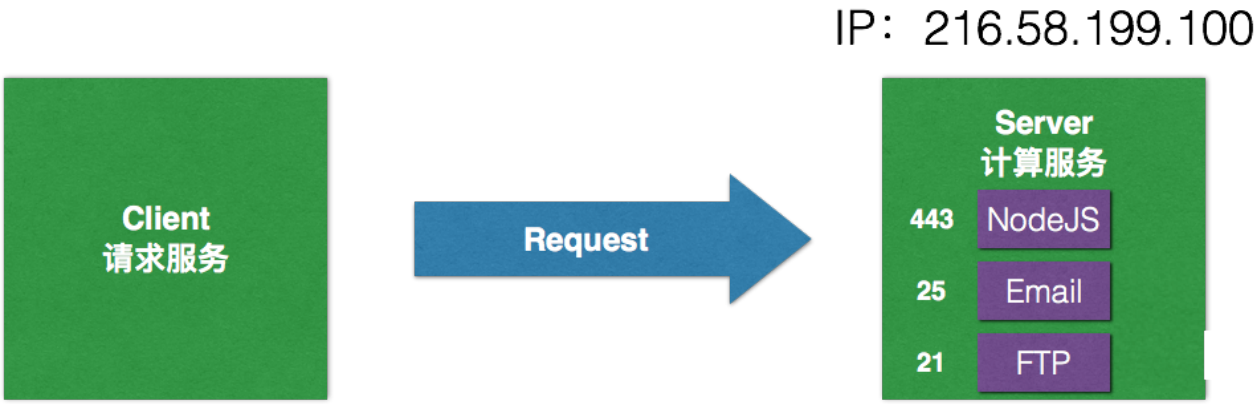
当客户端知道服务器的IP地址后，接下来就是如何与之通信。TCP协议正是解决数据传输层的通信协议，它能够在不同的计算机之间建立可靠的像管道一样的连接。如下图所示：



计算机操作系统将TCP连接抽象为套接字(Socket)，应用程序通过使用Socket接口完成TCP连接，通过TCP连接，客户端可以向服务器发送一个一个的数据包(Packet)。例如发送a=1,b=2这样的文本数据，服务器接收并计算得到结果3以后，同样通过数据包将其发送回客户端。

端口

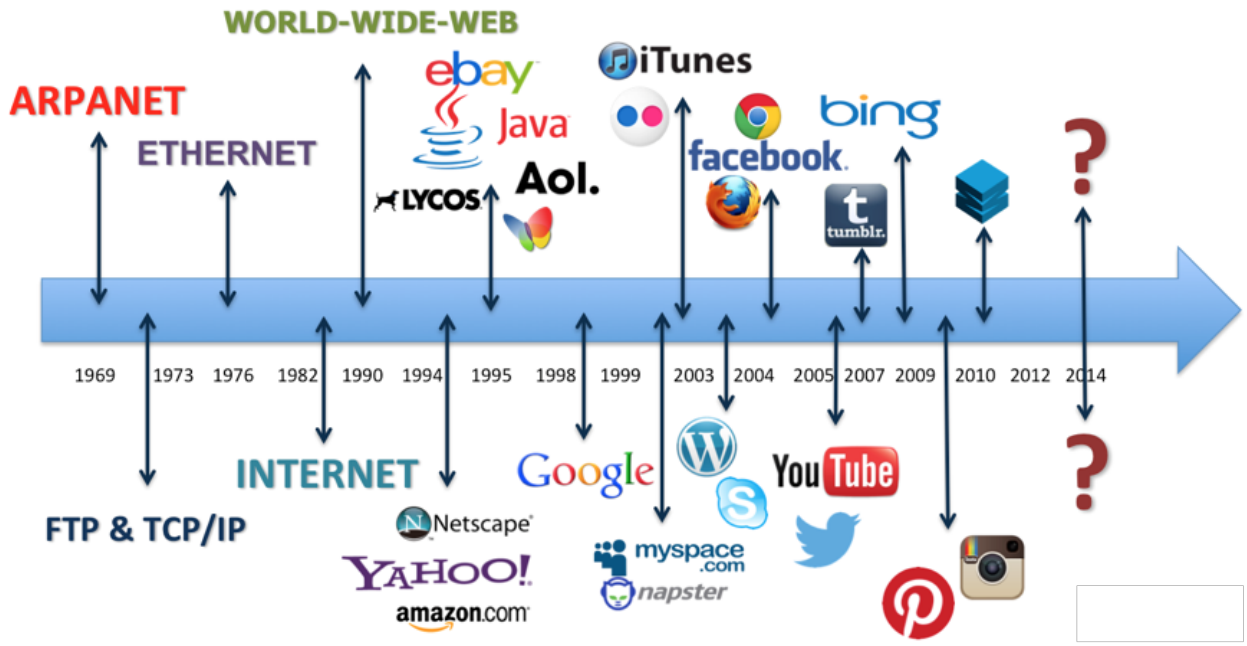
计算机的操作系统中同时运行着多个进程（例如IM工具，邮件服务器，Web服务器），当操作系统收到与其他计算机建立TCP连接后，它如何知道收到的数据包应该交给哪个进程来处理呢？答案是端口（Port），应用程序通过操作系统提供的Socket接口建立TCP连接时，必须告诉操作系统通过哪一个端口获取TCP数据包（端口的取值范围是0-65535，不同的进程不可以使用同一个端口），这个过程也叫应用程序监听(Listen)端口。如下图所示：



当客户端发起的TCP连接地址是216.58.199.100:80时（IP:Port形式的地址），服务器操作系统对于每一个访问80端口的TCP数据包，交给Web服务器进程处理。

1.3 Web的诞生

在互联网的发展过程中，World Wide Web出现于1990年。所谓“Java Web 开发”，这其中的Web便是World Wide Web的简称，中文称为万维网。



在上面这张图中，World Wide Web的位置很有趣。可以发现，这张图中在这个词之前的都是冷冰冰的技术术语，非计算机专业的同学可能都没有听过。但是在World Wide Web之后的很多词语都耳熟能详了吧。比如Yahoo、Google和Facebook，这都是互联网发展历史上里程碑式的伟大企业。为什么会有这个现象呢?

在TCP/IP协议上传输信息的方式很早就出现了，比如FTP就是通过TCP/IP传输文件的方式。为什么万维网会带来这么神奇的效应呢？因为它第一次带来了一种叫做超媒体（Hypermedia）的内容形态。超媒体可以在TCP/IP协议之上的传输，可以包含文字、音频、视频，同时可以相互链接，是一种全新的信息表达方式。

万维网是由欧洲粒子研究中心的科学家博纳斯.李发明的。他也是一个互联网发展历程中里程碑式的人物，设计了一套支撑万维网的一整套体系。其实严格来说，中国商业上的互联网报道，严格意义上指的其实都是Web，即万维网。比如大家可能听过这样的故事，说马云在1995年的时候，在美国看到了互联网，所以回国创办了阿里巴巴，那个时候他们做的是中国黄页。其实更准确地说，马云看到的其实是万维网。从技术上理解，中国黄页就是将企业信息描述成为超媒体（就是HTML文件），让所有人都能访问到。

Web最早只是一种静态信息的发布媒介，就是我们访问到的网页都是事先写好的固定信息。在发展过程中，逐渐可以用来实现动态的功能。即通过浏览器不仅仅能够看静态的信息，还是使用动态的业务功能。1994年诞生的Amazon就是一个电子商务网站，是典型动态Web应用。Web诞生之后，不管在商业还是在技术上都发展极其迅速。以至于2001年甚至出现了.com泡沫，那个时候只要企业做一个.com的网站就被认为是一个高科技企业。一个.com网站，从技术上来看，就是万维网这个超过规模的分布式系统中的一个小节点。

所以万维网的出现之后，真正的改变普通人生活的各类网络应用才开始不断涌现。

1.5 Web的技术内涵

万维网最早其实就是一帮科学家在玩，他们希望通过互联网来传送实验数据。因此发明了描述信息、定位信息和传输信息的一整套技术体系。

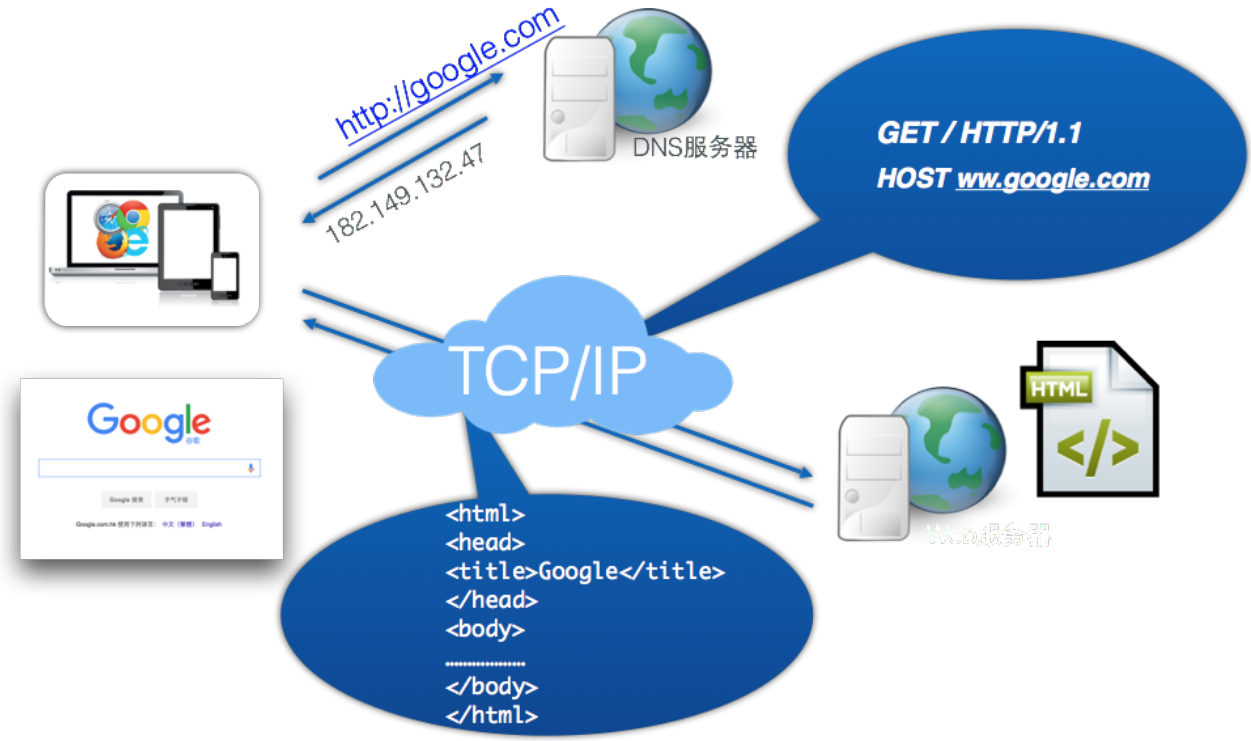
万维网从技术内涵上怎么来理解呢？首先它是运行在互联网上的，是一个TCP/IP协议之上的应用协议，它是一种超文本文档相互链接形成的一种超大规模的分布式系统。

这里需要了解一下三个术语：**HTTP超文本传输协议**、**HTML超文本语言**和**URL**统一资源定位服务。虽然是技术术语，但是在我们每天的上网过程中都能看到它们的痕迹。

- 今天你们经常听到HTML5，比如微信的HTML5页面，HTML5游戏等。HTML是从1.0版本开始的，有一个叫W3C的标准化组织，逐步在维护这样一套HTML语言的标准，一直演化到今天的HTML5。HTML的作用是描述超文本文档。比如说我们希望有一个东西展示给用户，你需要用HTML表达出来，这样才能放到万维网上让别人浏览。
- URL可以理解为网址，就是我们在浏览器中输入的一个字符串。它用来定位超文本文档，URL协议定义了Web上如何标识出一个网页，即超文本文档。这跟IP协议如何标识出一台计算机的意义是相似的。
- 我们在浏览器中输入网址，会看到网址前面都有一个“http”，这指的就是HTTP协议。HTTP协议定义了如何与超文本交互，信息通过超文本文档描述好了，也可以定位到了，接下来就需要通过HTTP协议去访问。HTTP协议定义了一套与网页交互的工作，我们暂时不去细究，我们只需知道这是获取信息的一种协议，浏览器会把我们的各种动作翻译成HTTP协议相关动作与Web服务器进行交互，并且将得到的超文本文档渲染成可读的内容让我们方便浏览。我们所谓的“连接”在技术层面都时这些协议在默默地发挥着基础设施的作用。这就是Web的基本原理。

1.6 访问一个网页时发生了什么?

基于对这三个协议的理解，我们来重新审核在浏览器中访问一个Web页面的过程。



在移动端和PC端都可以访问网页，比如我们去访问Google的页面，google.com网址就是符合URL规范的网址。浏览器看到这个网址，首先去查询DNS（Domain Naming Service）服务器，DNS服务器会将网址转换为IP地址。万维网是运行在TCP/IP协议之上的，所以首先需要知道Web服务器的IP地址，DNS帮我们做了这件事情。

有了IP地址，浏览器就可以基于HTTP协议，向远程的Web服务器发送请求了。而Google的Web服务器就能够接收到这样的请求，收到这样的请求之后，它就会调用后端的一系列功能并且最终组装出HTML页面，通过HTTP协议返回给浏览器。浏览器把返回的HTML文本渲染成为一个美观而且可读的页面，这就是在浏览器中看到Google的页面了。这整个过程是在TCP/IP协议之上完成的，Web请求和Web响应

都会安装TCP协议要求的方式进行打包和传输。

1.7 了解Java Web开发

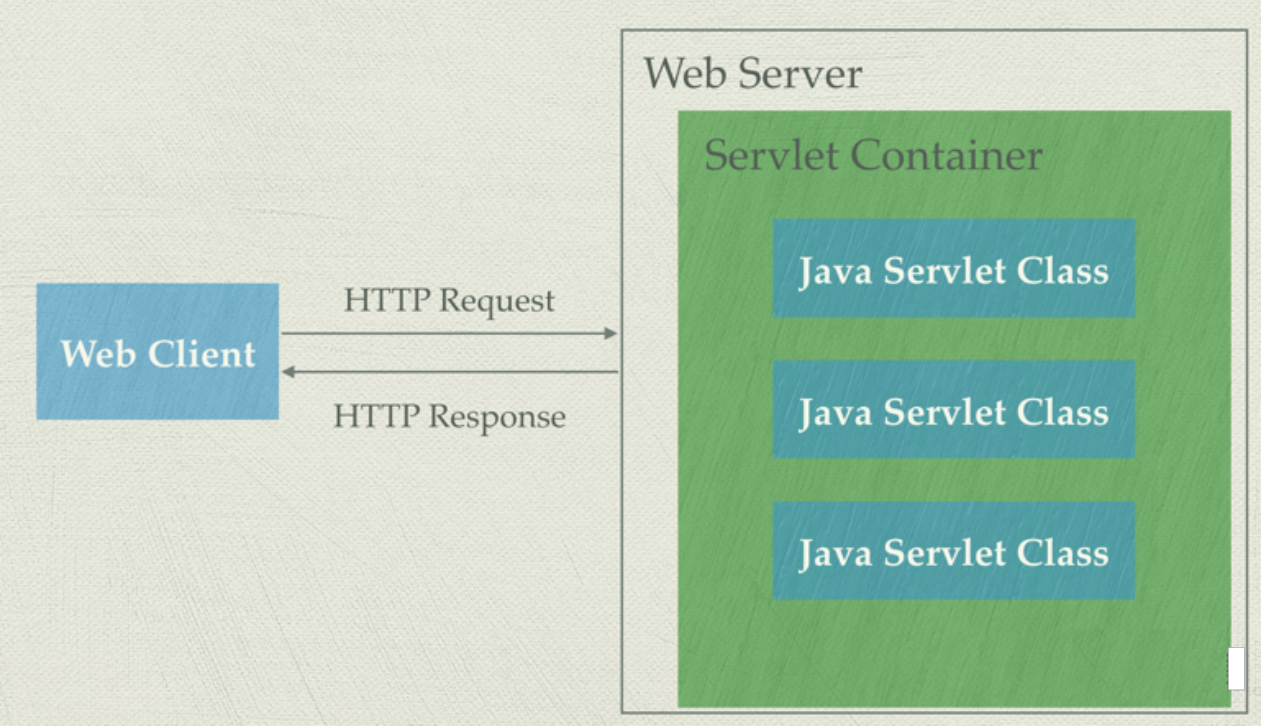
从用户的角度，我们已经很清楚访问Web应用的整个过程了。但是现在我们需要变身为一个Java Web开发者了，我们现在需要开发出Web应用，让普通用户可以与我们的应用交互。那作为开发者我们需要做什么呢?

静态的Web页面只需要使用HTML语言就可以编写，服务器会根据URL地址找到该页面以HTTP响应的格式返回给客户端。但是一个实用的Web应用肯定是动态，即页面的内容是通过程序在运行时动态生成的，而我们要做的就是编写一组HTTP请求处理程序。

从开发者的角度来看，一个请求处理的流程是这样的：

1. 监听Web服务器某网络端口，例如80；
2. 从网络端口读取HTTP请求信息；
3. 根据HTTP请求信息进行处理，生成HTTP响应内容；
4. 将响应内容通过网络I/O接口返回；
5. 返回第二步，重复。

基于Java进行开发，我们只需要专注于第三步就行了，Java提供的Web容器会帮我们处理其它的步骤，这个容器成为Servlet容器，Tomcat就是一种Servlet容器的实现，你接下来马上就会用到。所以我们的主要工作就是编写处理请求的**Java Servlet**代码。



所以简单地理解，一个Servlet就是一段将HTTP请求信息转换为一组HTML标签内容的程序，我们开发者做的工作就是实现这种转换。

一个HTTP请求信息包含哪些内容呢？ 你可以从浏览器中的网址就能看到，比如一个URL是这样的：

将到这里你现在应该清楚作为一名Java Web开发者的主要工作了：编写Java代码，将HTTP请求中的URL路径和参数等信息转换为HTML内容。所以简单来说，接下来的时间里，你需要进行以下的学习和实践：

- 建立Java Web编程环境
- 学习处理HTTP请求的相关机制和编程接口
- 掌握HTML语言以及动态生成HTML内容的技术
- 动态的功能一般都涉及数据存储，所以你需要了解数据库
- 实战项目中的一些最佳实践
- 千里之行，始于跬步，先赶紧把Java开发环境安装和配置起来吧！

2. 开发环境和第一个servlet项目

2.1 安装Tomcat

- brew install

```
brew install tomcat
```

- 推荐使用解压缩方式

```
download the Apache Tomcat first. Visit https://tomcat.apache.org/ page,
select ‘Download’ — ‘Tomcat 9’ — ‘Binary Distributions’ — ‘Core’ — ‘tar.gz’.
decompress it.
```

2.2 brew 查看Tomcat安装目录

```
 $brew list tomcat
/usr/local/Cellar/tomcat/9.0.30/bin/catalina
/usr/local/Cellar/tomcat/9.0.30/homebrew.mxcl.tomcat.plist
/usr/local/Cellar/tomcat/9.0.30/libexec/bin/ (17 files)
/usr/local/Cellar/tomcat/9.0.30/libexec/conf/ (10 files)
/usr/local/Cellar/tomcat/9.0.30/libexec/lib/ (32 files)
/usr/local/Cellar/tomcat/9.0.30/libexec/logs/ (5 files)
/usr/local/Cellar/tomcat/9.0.30/libexec/temp/safeToDelete.tmp
/usr/local/Cellar/tomcat/9.0.30/libexec/webapps/ (575 files)
/usr/local/Cellar/tomcat/9.0.30/libexec/ (2 files)
/usr/local/Cellar/tomcat/9.0.30/RELEASE-NOTES
/usr/local/Cellar/tomcat/9.0.30/RUNNING.txt
```

2.3 IDEA创建maven项目

Create a Maven Project with Servlet in IntelliJ IDEA 2018

3. Servlet简介

3.1 什么是Servlet

在Java世界里，Servlet技术用来创建Web应用程序——本质上来说，Servlet是运行于服务器端的Java程序，它能够接受客户端发起的HTTP请求并动态地生成页面内容。Servlet最初是对任意客户端-服务端通讯协议的一层抽象，但在Web技术蓬勃发展的互联网时代，它几乎已经完全和HTTP通讯协议绑定在一起使用，所以我们常用的术语Servlet——是"HTTP Servlet"的缩写。开发者可以基于Servlet在Java平台上开发动态Web应用程序，基于收到的HTTP请求生成响应内容，HTTP响应内容可以是纯文本、HTML、XML、JSON格式的数据。

Servlet API是Java EE规范的一部分，是Web开发中最常用到的部分。通常Servlet会应用在如下场景中：

- 处理从浏览器页面中提交的HTML表单数据
- 根据HTTP请求信息，动态生成HTTP响应内容。例如：根据HTTP请求数据从数据库中读取不同的内容并返回
- 使用Cookie或URL重写技术在无状态的HTTP协议之上实现对客户端状态的管理。例如：用户系统（登录一次后可以访问站点下的所有页面而无需重复登录）、电商网站的购物车功能

3.2 第一个Servlet程序

我们编写的第一个Servlet程序：

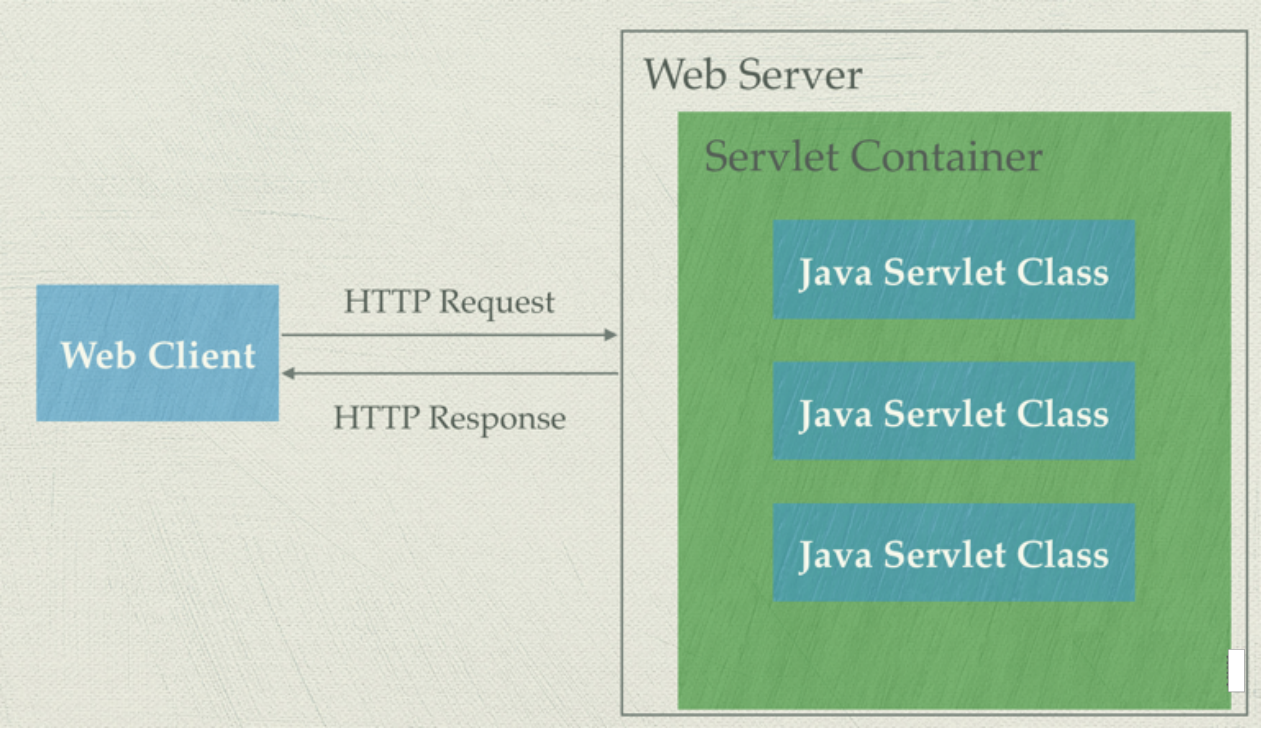
```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/HelloWorld")
public class HelloWorld extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        response.getWriter().append("Hello 半圆!");
    }
}
```

这段程序体现了Servlet的几个要素：

- 在程序实现的角度，一个Servlet是继承了javax.servlet.http.HttpServlet的子类
- doGet()方法中定义了处理HTTP GET请求的动作，在例子中是将"Hello 半圆!"作为响应内容返回
- HttpServletRequest对象中封装了HTTP请求信息，通过该对象可以访问HTTP请求数据（Header，表单，URL等等）
- HttpServletResponse可以用来生成HTTP响应内容，这里仅仅是设置了返回内容的编码，并写入了一个字符串
- @WebServlet("/HelloWorld")注解表明只有请求的路径是/HelloWorld（例如http://localhost:8080/HelloWorld）时，才会执行该Servlet生成返回内容

3.3 Servlet容器

为了让Servlet程序运行起来，我们需要将其部署在Servlet容器中。



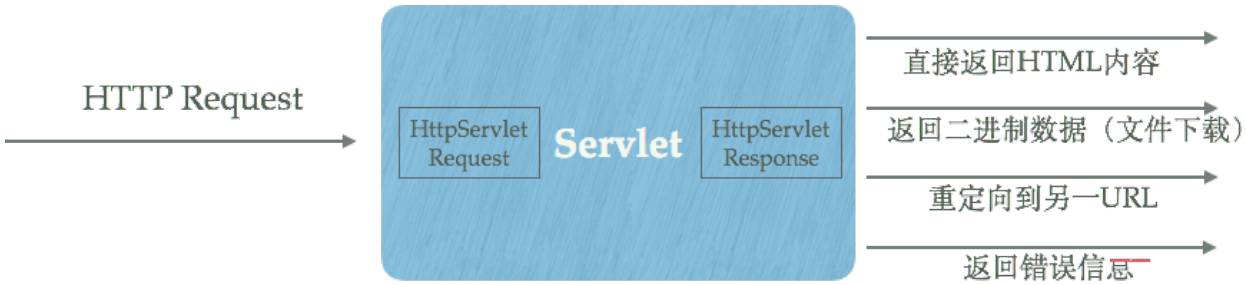
Servlet容器是Web服务器和Servlet进行通讯的主要构件，它的主要职责包括：

- 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
- 并发请求的多线程处理、线程池管理
- Session管理，HTTP缓存等

将这些公共的任务抽象到Servlet容器这个构件中，也有利于开发者专注于业务逻辑（也就是Servlet中处理请求的具体响应方法，doGet(), doPost()等）

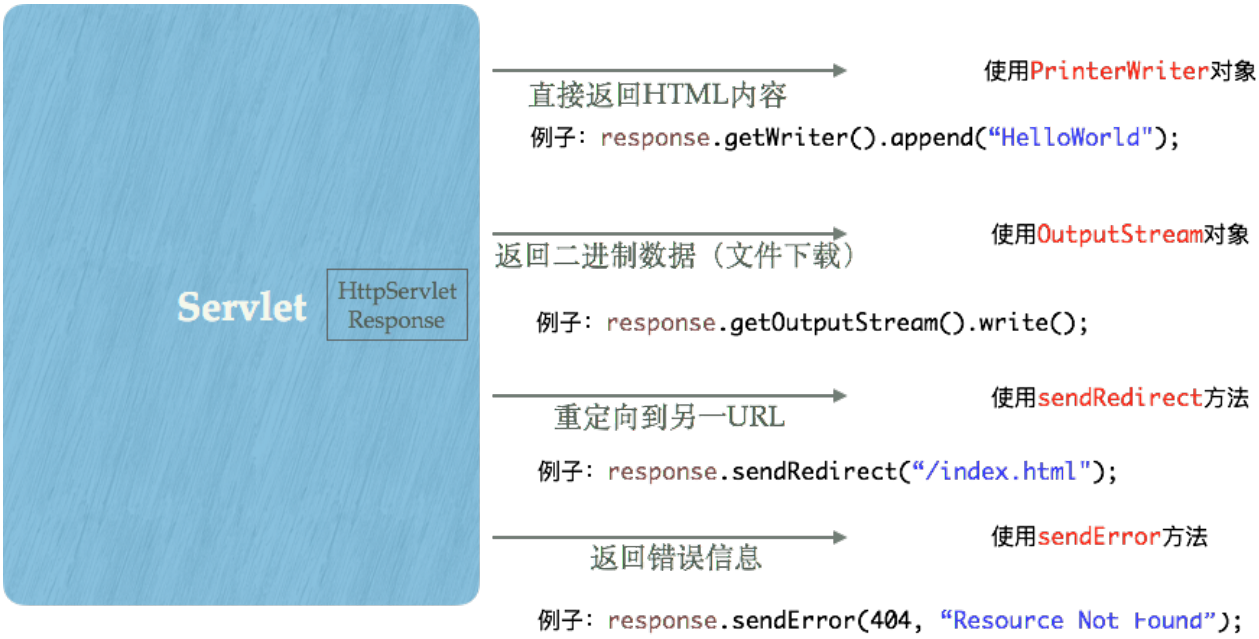
3.4 Servlet处理机制

Servlet容器监听某网络端口，接受并读取客户端发送而来的HTTP请求后，将其分发至指定的Servlet进行处理，Servlet的service()方法的两个参数HttpServletRequest和HttpServletResponse正是对HTTP请求和响应的封装，Servlet接受请求后，可以进行响应的处理：



3.5 通过HttpServletResponse生成返回内容

图中的四种处理响应的方式，都是通过HttpServletResponse对象来完成的：



这里暂时我们只需要使用到第一种方式，即通过PrintWriter对象返回字符文本。

3.6 Servlet生命周期

Servlet依托于Servlet容器运行，它包含三个方法，他们分别在特定的时机被Servlet容器所调用：

- init()，当Servlet第一次被容器加载进入内存后调用，一般用于载入一些特定的资源和配置
- service()，一旦有对应URL的HTTP请求访问即被调用，它会根据HTTP请求中的method信息将请求分发至相应的方法进行处理（doGet(), doPost()），service()方法一般不需要开发者重写。
- destroy()，Servlet被销毁时调用，一般用来释放、清理资源

开发者可以通过重写Servlet生命周期中对应的方法来实现特定的功能，例如在init()方法中从配置文件里读取配置信息、在destroy()方法中释放数据库连接资源，而Servlet自身处理HTTP请求的逻辑则需要重载service()方法。HTTP协议中规定的方法有很多，为了防止在service()中使用过多的if来处理各种方法的响应逻辑，HttpServlet添加了doGet()，doPost()用于处理相应的HTTP方法。