<http://blog.csdn.net/hejingyuan6/article/details/30273879>

**1、CGI与Servlet对比**

公共网关接口（CommonGateway Interface，CGI）脚本是生成动态内容的主要技术。虽然使用得非常广泛，但CGI脚本技术有很多的缺陷，这包括**平台相关性和缺乏可扩展性**。为了避免这些局限性，Java Servlet技术因应而生，它能够以一种可移植的方法来提供动态的、面向用户的内容。处理用户请求。

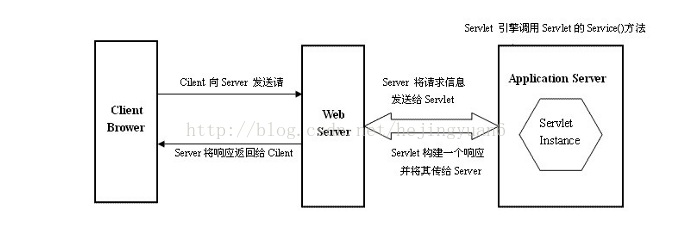
* **对比一：**当用户浏览器发出一个Http/CGI的请求，或者说调用一个CGI程序的时候，服务器端就要新启用一个进程(而且是每次都要调用)，调用CGI程序越多(特别是访问量高的时候)，就要消耗系统越多的处理时间，只剩下越来越少的系统资源，对于用户来说，只能是漫长的等待服务器端的返回页面了，这对于电子商务激烈发展的今天来说，不能不说是一种技术上的遗憾。

而Servlet充分发挥了服务器端的资源并高效的利用。每次调用Servlet时并不是新启用一个进程，而是在一个Web服务器的进程中共享和分离线程，而线程最大的好处在于可以共享一个数据源，使系统资源被有效利用。故servlet不是线程安全的，单实例多线程的

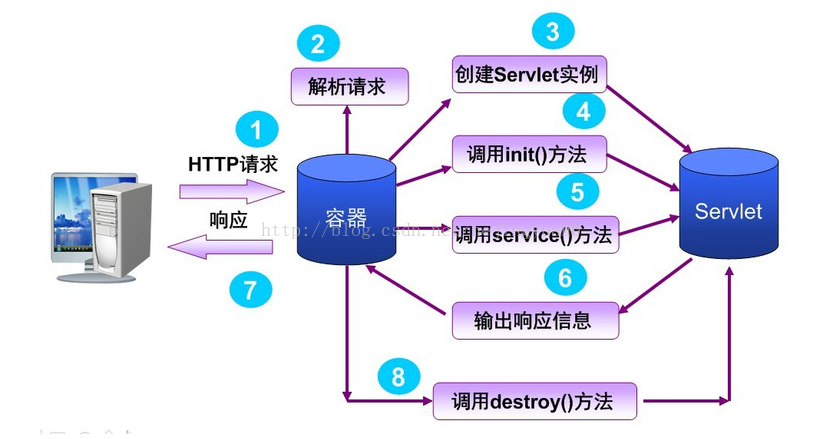
* **对比二：**传统的CGI程序，不具备平台无关性特征，系统环境发生变化，CGI程序就要瘫痪，而Servlet具备Java的平台无关性，在系统开发过程中保持了系统的可扩展性、高效性。
* **对比三：**传统技术中，一般大都为二层的系统架构，即Web服务器+数据库服务器，导致网站访问量大的时候，无法克服CGI程序与数据库建立连接时速度慢的瓶颈，从而死机、数据库死锁现象频繁发生。而我们的Servlet有连接池的概念，它可以利用多线程的优点，在系统缓存中事先建立好若干与数据库的连接，到时候若想和数据库打交道可以随时跟系统"要"一个连接即可，反应速度可想而知。

**2、Servlet的运行过程**

* 客户端发送请求至**服务器端**；
* 服务器端根据web.xml文件中的Servlet**相关配置信息**，将客户端请求转发到相应的Servlet；
* Servlet引擎调用**Service()**方法，根据**request对象**中封装的用户请求与数据库进行交互，返回数据之后，Servlet会将返回的数据封装到**response对象**中；
* Servlet生成响应内容并将其传给**服务器**。响应内容动态生成，通常取决于客户端的请求；
* 服务器将响应返回给客户端



**3、Servlet生命周期**



* **加载和实例化：**在第一次请求Servlet时，Servlet容器将会创建Servlet实例；
* **初始化：**Servlet容器加载完成Servlet之后，必须进行初始化，此时，init方法将被调用；
* Servlet初始化之后，就处于**响应请求的就绪状态**，此时如有客户端请求发送，就会调用Servlet实例的service()方法，并且根据用户的请求方式，调用doPost或者doGet方法；
* 最后，Servlet容器负责将Servlet实例进行销毁，调用destroy方法实现；

对于更多的客户端请求，Server创建新的请求和响应对象，仍然激活此Servlet的service()方法，将这两个对象作为参数传递给它。如此重复以上的循环，但**无需再次调用init()方法**。

一般**Servlet只初始化一次(只有一个对象)**，当Server不再需要Servlet时(一般当Server关闭时)，Server调用Servlet的Destroy()方法。

**4、Servlet如何同时处理多个请求？**

Servlet采用多线程来处理多个请求的同时访问。Servlet容器通过线程池来管理维护服务请求。所谓线程池，相当于数据库连接池，实际上是等待执行代码的一组线程，叫做工作者线程。Servlet容器通过一个调度线程来管理工作者线程。

* 当容器收到一个**Servlet的访问请求**，调度者线程就从线程池中选出一个工作者线程，将用户请求传递给该线程，然后由**该线程处理Servlet的service()方法**；
* 当这个线程在执行的时候，容器收到一个新的请求，调度者线程再次从线程池中选出一个新的工作者线程；
* 当容器同时收到对同一个Servlet的多个请求时，那么Servlet的service方法将在多线程中并发执行。

**注：**

* Servlet容器默认采用单实例多线程的方式来处理请求。这样减少了产生Servlet实例的开销，提升了对请求的响应时间；
* 对于Tomcat容器来讲，可以在其server.xml中通过<Connector>中设置线程池中的线程数目。

**5、如何开发线程安全的Servlet？**

Servlet容器采用多线程来处理请求，提高性能的同时也造成了**线程安全问题**。要开发线程安全的Servlet应该从一下几个方面进行：

* **变量的线程安全：**多线程并不共享局部变量，所以我们要尽可能的在Servlet中使用局部变量；
* **代码块的线程安全：**使用同步块Synchronized，防止可能调用的代码块；但是要注意的是，要**尽可能得缩小同步代码的方范围**，不要在service方法和响应方法上直接使用同步，这会严重影响性能。
* **属性的线程安全**：ServletContext，HttpSession，ServletRequest对象中属性；
* **使用同步集合：**使用**Vector**代替ArrayList，使用HashTable代替HashMap；
* 不要在Servlet中创建自己的线程来完成某个功能； Servlet本身就是多线程的，如果再创建新的线程，将会导致线程执行复杂化，出现线程安全问题；
* 在多个Servlet中，**对外部对象**，比如：文件；进行修改操作一定要加锁，做到**互斥访问**；

**6、总结**

一个servlet就是Java编程语言中的一个类，它**被用来扩展服务器的性能**，服务器上驻留着可以通过“请求-响应”编程模型来访问的应用程序。Servlet通过解析http请求，取得客户端的参数来进行下一步操作。其实简单来说，servlet就是一个控制器，取参数，调用业务逻辑。

而在.net 中HttpHandler是一个HTTP请求的真正处理中心，也正是在这个HttpHandler容器中，ASP.NET Framework才真正地对客户端请求的服务器页面做出编译和执行，并将处理过后的信息附加在HTTP请求信息流中再次返回到HttpModule中。

**1、同一用户的不同页面共享数据**

1）cookie

**服务器在客户端保存用户的信息**，比如登录名、密码等，就是cookie

cookie一般保存在**客户端**的C:\Document and Setting

功能：a. 保存用户名、密码，在一定时间不用重新登录

b. 记录用户访问网站的喜好（如有无背景音乐、网页背景颜色是什么）

c. 网站的个性化，比如定制网站的服务、内容

2）sendRedirect跳转

通过该方法将信息传给下一页面，比如sendRedirect("welcome?uname=shunping");的形式

优点：传送信息速度比较快

缺点：只能**传送字符串**，并不能传送一个对象

注意：a. welcome代表要跳转的那个servlet的url

b. servlet url 名和值间有"?"号

c. 如要传递两个以上的值，它们之间要用"&"号分开，比如：

sendRedierct("welcome?uname=shunping&upass=ok")

d. 如果传递的是中文，将得到乱码

3）隐藏表单：常见且简单的

形如：<form action=login>

<input type=hidden name=a value=b>

</form>

4）session：**同一用户不同页面共享session中的数据**(慎用)

当用户打开浏览器，访问某个网站时，服务器就会在服务器内存为该浏览器分配一个空间，该空间被这个浏览器独占

这个空间就是session空间，该空间中的数据默认存在时间为30min，也可修改。

功能：a. 网上商城的购物车

b. 保存登录用户的信息

c. 将某些数据放入到session中，供同一用户的各个页面使用

d. 防止用户非法登录到某个页面

.....

session可看作一张表，表中有两列，而行理论没有限制，每一行就是session的一个属性

**每个属性**包含两部分：一是该属性的名字（String），二是它的值（Object）

如何使用：

a. 得到session：HttpSession hs = request.getSession(true);

b. 向session添加属性：hs.setAttribute(String name, Object val)

c. 从session得到某个属性：String name = hs.getAttribute(String name)

d. 从session删除某个属性：hs.removeAttribute(String name)

e. 得到session的Id号：hs.getId();

f. 删除整个session(安全注销)：hs.setMaxInactiveInterval(0),负值时，永不超时

g. 设置session的生存时间：hs.setMaxInactiveInterval(int interval)，单位为s

**注意事项：**

a. 修改session存在时间：一是在web.xml中修改（session-timeout）;二是程序中

b. 其中30min指的是**用户发呆时间**(即一直不操作session的时间)，不是累计时间，负值时永不超时；

c. 当某个浏览器访问网站时，服务器会给浏览器分配一个唯一的session id，并以此来区分**不同的浏览器（即客户端）**

d. 因为session的各个属性要占**服务器的内存**，因此都是迫不得已才使用的

实例参考

2、servlet中操作数据库：即在数据库验证用户

注意：sql的注入漏洞

如：select \* from where uaername='asd' and password='aa' or 1='a'

3、如何在servlet中显示图片

在webApps（即在你的目录下，建立一个文件夹，比如imgs）,然后将图片拷贝至文件夹下，

在servlet中添加：<img src="./imgs/图片名"/> 即可

4、分页技术

分页算法：定义四个变量，有各自的用处

int pageSize：每页显示多少条记录

int pageNow：希望显示第几页

int pageCount：一共有多少页

int rowCount：一共有多少条记录

说明：pageSize是指定的，pageNow是用户选择的

rowCount是从表中查询得到的，pageCount是计算出来的，该计算式为：

if(rowCount%pageSize==0){

pageCount = rowCount/pageSize;

}else{

pageCount = rowCount/pageSize + 1;

}

rowCount：表中的记录条数，即语法select count(\*) from 表名

查询语句：

select 字段名列表 from 表名 where id between ? and ?

**显示第pageNow页的数据：**

sql 2000查询语句：

select top pageSize 字段名列表 from 表名 where id not in

(select top pageSize\*(pageNow-1)id from 表名)

MySQL查询语句：

select \* from 表名 order by 字段名列表 desc limit n

类似于select top n的语法：

select \* from 表名 order by 字段名列表 desc/asc limit [position,]counter

增加表记录的语句：

insert into 表名(字段1,...) select 字段1,字段2,..,字段n from 表名

5、cookie的讲解

1）服务器再客户端保存用户的信息，比如登录名、密码等...

服务器在需要时可以从客户端读取

使用：a. cookie像**一张表**，分两列，一是名字，一是值，数据类型都是String

b. 如何创建cookie（在服务器端创建的）

Cookie c = new Cookie(String name, String val);

c. 如何将一个Cookie添加到客户端

response.addCookie(c);

d. 如何读取cookie（从客户端读到服务器）

response.getCookies(); //返回cookie数组信息

e. 设置存在时间,不设置cookie将不会保存

c.setMaxAge(val); //单位为s

f. 删除cookie：c.setMaxAge(0)

2）cookie与session比较

a. **存在位置**：cookie保存在客户端，session保存在服务器端

b. **安全性**：cookie的安全性比session弱

c. **网络传输量**：cookie通过网络在客户端与服务器传输，而session保存在服务器端，不需要传输

d. **生命周期**：cookie的周期是累积的，而session是间隔的；另外关机会造成session周期的结束，而cookie不变

6、展示与业务逻辑应分装在不同文件

网站框架的改进，尽量避免重复代码，影视整个框架有清晰的层次关系

a. 进行分层（界面层、业务逻辑层）[mv模式]：m --> model模型 --> view 模型

b. 进行分装到类：常用代码（比如连接数据库）

c.

connDB：数据库连接

userBean.java：**数据操作**，一个对象 <----> 数据库表中的一条记录：添加、修改一一映射于数据库中的表

userBeanCL.java：**业务逻辑处理类**，即处理数据库表 <----> 操作 userBean

7、ServletContext讲解和使用

1）ServletContext：可以想象成是一**共用空间**，可以被所有客户访问，区别于cookie、session

a. 如何得到ServletContext实例：ServletContext sc = this.getServletContext()

b. 格式可视为一张表，相似于session，每一行就是一属性

添加属性：setAttribute(String name, Object ob)

得到值：getAttribute(String name) 返回Object

删除属性：removeAttribute(String name)

c. 生命周期：从创建开始到服务器关闭而结束

注：ServletContext中的数据长时间的保存在服务器，会占用内存，因此不建议使用

2）servlet如何操作文件

读取文件中的内容：FileReader --> BufferedReader，**读取完毕后一定要关闭文件流**

读取行数据：readline()

//创建一个FileWriter 从内存数据 -> 磁盘文件（写入，输出）

//从内存数据 <- 磁盘文件（读入，输入）

写数据：FileWriter --> BufferedWriter，写入数据完后关闭写入流

写数据：write(String str)

3）网站计数器：统计某个网页被点击（浏览）的次数

ServletContext使用功能：网站计数器、网站在线用户显示、简单的聊天系统

总之，如果涉及到不同用户共享数据，且数据流不大，又不希望写入数据库，可考虑使用ServletContext

怎么实现？怎样才算一个有效的点击？

**算法：**不同的IP访问该网页，每一次有效点击；如果是同一IP在一定时间（比如一天），不管点击该网页多少次都算一次

用户退出系统，重新登录也算一次

使用ServletContext实现：减少对文件的操作次数，获得客户端的IP地址，从而也减少了点击次数

优化算法：**结合Servlet的生命周期，减小对文件的操作次数**

**思路：**在init()方法中初始化ServletContext中属性times次数对应的**初始值**，而在destroy()中回写times到文件中，次数自加操作在ServletContext中进行完成，从而对文件操作的次数减少到两次（init()与destory()）

得到用户的IP地址：req.getRemoteAddr()

得到用户的主机名：req.getRemoteHost()

在线用户？

8、显示用户头像

数据库中保存：头像的文件名，最后再调用

9、指定跳转到某页

表单提交原理

10、修改Tomcat默认端口：~tomcat~/conf/server.xml文件中的port="8080",并将其修改为合适的端口

1-1024：有名端口

**设置虚拟目录：**在conf文件夹中的server.xml中合适位置加入以下：

<Context path="/yousite" docBase="c:\mesite" debug="0"/>

yousite：你的站点名，就是主文件夹名称

mysite：站点存放的磁盘目录

设置管理员密码：tomcat-users.xml 中修改密码

<user name="用户名" password="密码" roles="standard,manager,admin"/>

配置数据源和连接池：

问题：java程序**每次操作数据库**，需要加载驱动才能得到一根连接（数据库）

解决机制：如果连接池中有空闲的连接，tomcat就分配给请求客户端，如果没有空闲的连接，请求客户端将会在队列池中等待

两种方法配置：

a. 通过修改%Tomcat Home%/conf/server.xml文件，即在指定位置添加如下代码即可

b. 通过tomcat管理界面来配置：Tomcat administation页面中修改