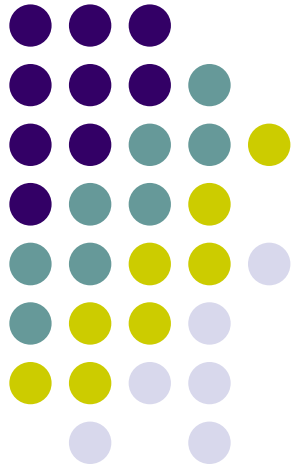
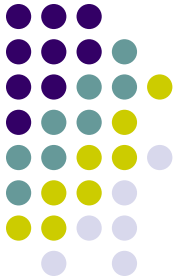
第六章 基于Java Servlet的Web程序开发

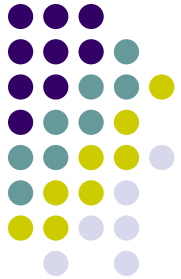


主要内容



- Java Servlet概述
- Java Servlet的编程原理
- Java Servlet所需的开发环境
- Java Servlet举例
- 开发Java Servlet的过程

Java Servlet概述



- Servlet的发展背景
- Servlet是什么?
- Servlet的特点
- Servlet与JSP

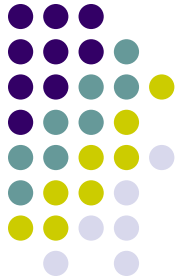
1、 Servlet的发展背景

在互连网发展的早期，人们面临着如何编写能够处理用户输入并能够产生动态内容的Web程序的问题，当时的解决办法就是公共网关接口(CGI)程序。

CGI程序可以使用多种编程语言开发，如：Perl就是一种常用的CGI语言，现在绝大多数Web服务器都支持CGI，从而使得CGI成为Web程序的通用技术。

但是，CGI有它的缺点，首先就是性能和可伸缩性问题，在CGI中，对应来自客户的每个请求都要创建一个进程来响应，以提供服务，这极大的消耗了服务器的资源。

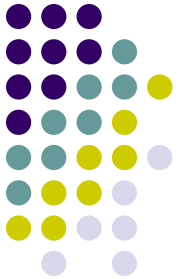
其次就是安全问题，CGI程序和Web服务器的结合本质上是一种松耦合的结合，存在着严重的安全漏洞。



因此，各个Web服务器厂商纷纷推出各自的Web服务器的API，程序员可以使用这些API来编写动态Web应用程序，它虽然克服CGI的不足，但是却带来了兼容性的问题，因为这些API都是专用于特定服务器的，要想移植在不同服务器上API编写的Web程序十分困难。

于是，Servlet出现了，它作为Java技术平台支持Web应用的突破口和基础，解决了CGI和专用API技术存在的问题，借助Java与平台无关的优点，从而实现了CGI与专用API很好的折衷，使得Web应用程序设计技术前进了一大步。

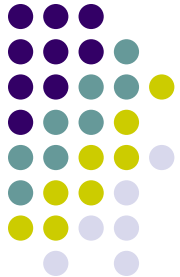
Java Servlet API很简单，几乎所有的Web服务器都支持Servlet程序，它通过多线程等技术解决了CGI的性能问题，通过Java语言提供的最强的兼容性，而且安全性也得到了极大的提高。



2、Servlet是什么?

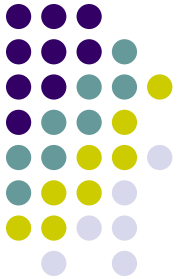
首先，Servlet是一个由Servlet容器管理的可以动态产生内容的Web构件。

再者，Servlet是一个小的、与平台无关的Java类，它实现了Java的Servlet接口，被编译成平台无关的字节码，然后加载到Servlet容器中运行，而且整个生命周期都是由Servlet容器管理。

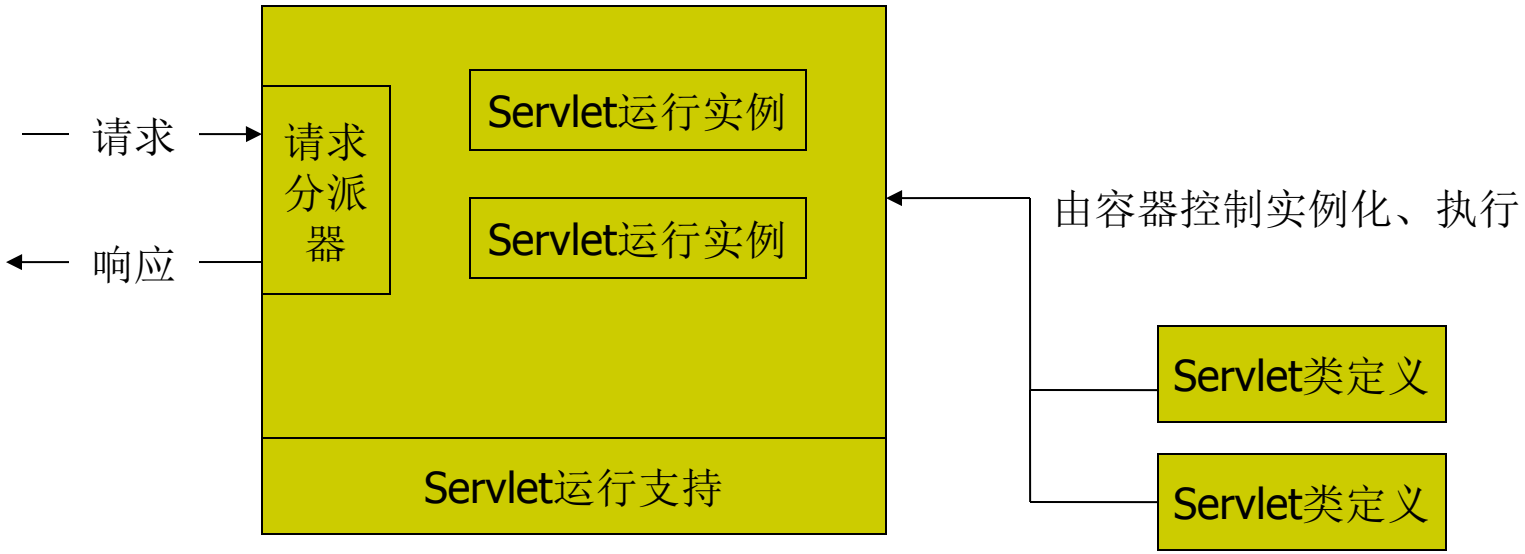


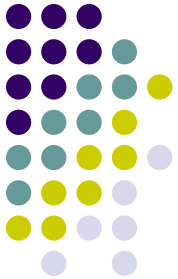
Servlet同客户端的交互遵循请求/响应模式，它是基于HTTP协议的基本机理，所以Servlet特别适合开发Web应用，那么请求/响应过程也是由Servlet容器管理。

Servlet容器，它是运行在Java虚拟机上的管理器，同Web服务器或应用服务器结合提供网络服务：译码基于MIME的请求信息，加载和管理Servlet，把请求发送到Servlet处理，并把处理结果形成基于MIME的响应回送给客户端。



Servlet容器可以在Web服务器内部实现，也可以作为Web服务器的附加构件提供。Servlet容器至少必须支持HTTP协议，也可以支持其他协议，下图描述了Servlet和Servlet容器的关系：





Servlet完成的主要功能：

(1)创建并返回一个包含基于客户请求性质的动态内容的完整的 HTML 页面。

(2)创建可嵌入到现有 HTML 页面中的一部分 HTML 页面（HTML 片段）。

(3)与其它服务器资源（包括数据库和基于 Java 的应用程序）进行通信。

(4)用多个客户机处理连接，接收多个客户机的输入，并将结果广播到多个客户机上。例如，Servlet 可以是多参与者的游戏服务器。

(5) 当允许在单连接方式下传送数据的情况下，在浏览器上打开服务器至applet的新连接，并将该连接保持在打开状态。当允许客户机和服务器简单、高效地执行会话的情况下，applet也可以启动客户浏览器和服务器之间的连接。可以通过定制协议或标准（如 IIOP）进行通信。

(6) 对特殊的处理采用 MIME 类型过滤数据，例如图像转换和服务器端包括（SSI）。

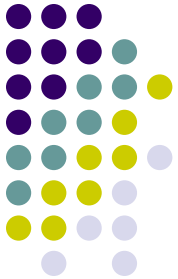
(7) 将定制的处理提供给所有服务器的标准例行程序。例如，Servlet 可以修改如何认证用户。

3、 Servlet的特点

Servlet与Applet的区别

相同之处：它们都不是独立的应用程序，没有main()方法；它们不是由用户或程序员调用，而是由另外一个应用程序(容器)调用；它们都有一个生命周期，包含init()和destroy()方法

不同之处：Applet具有很好的图形界面(AWT)，与浏览器一起，在客户端运行，Servlet则没有图形界面，运行在服务器端。



Servlet与CGI的比较

(1)高效：

在传统的CGI中，每个请求都要启动一个新的进程，如果CGI程序本身的执行时间较短，启动进程所需要的开销很可能反而超过实际执行时间。而在Servlet中，每个请求由一个轻量级的Java线程处理(而不是重量级的操作系统进程)

在传统CGI中，如果有N个并发的对同一CGI程序的请求，则该CGI程序的代码在内存中重复装载了N次；而对于Servlet，处理请求的是N个线程，只需要一份Servlet类代码。在性能优化方面，Servlet也比CGI有着更多的选择

(2)方便：

Servlet提供了大量的实用工具例程，例如自动地解析和解码HTML表单数据、读取和设置HTTP头、处理Cookie、跟踪会话状态等。

(3)功能：

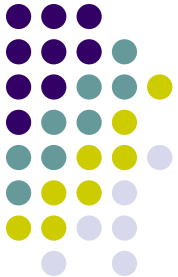
在Servlet中，许多使用传统CGI程序很难完成的任务都可以轻松地完成。例如， Servlet能够直接和Web服务器交互，而普通的CGI程序不能。 Servlet还能够在各个程序之间共享数据，使得数据库连接池之类的功能很容易实现。

(4)移植性:

Servlet用Java编写，Servlet API具有完善的标准。因此，为IPlanet Enterprise Server写的Servlet无需任何实质上的改动即可移植到Apache、Microsoft IIS或者WebStar。几乎所有的主流服务器都直接或通过插件支持Servlet。

(5)投资：

不仅有许多廉价甚至免费的Web服务器可供个人或小规模网站使用，而且对于现有的服务器，如果它不支持Servlet的话，要加上这部分功能也往往是免费的(或只需要极少的投资)



4、 Servlet与JSP

JavaServer Pages(JSP)是一种实现普通静态HTML和动态HTML混合编码的技术，JSP并没有增加任何本质上不能用Servlet实现的功能。

但是，在JSP中编写静态HTML更加方便，不必再用println语句来输出每一行HTML代码。

更重要的是，借助内容和外观的分离，页面制作中不同性质的任务可以方便地分开：比如，由页面设计者进行HTML设计，同时留出供Servlet程序员插入动态内容的空间。

Java Servlet的编程原理

- Servlet的本质
- Java Servlet API 2.2的类和接口构成
- HTTP Servlet应用编程接口
- Servlet的生命周期

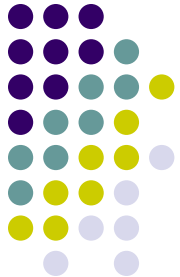
1、Servlet的本质

前面我们曾经提到Servlet就是一个实现Servlet接口的Java类。这是Servlet程序的本质。

编写Servlet程序时，我们可以直接实现Servlet接口，但是更为常用的方式是扩展一个实现了Servlet接口的子类。现在Servlet API中已经包含了两个实现了的Servlet接口的子类：GenericServlet子类和HttpServlet子类

GenericServlet类包含了对通用的协议无关的Servlet的支持，我们扩展它可以实现各种支持请求/响应模式协议的Servlet程序，如支持FTP、SMTP、POP协议的Servlet程序，GenericServlet类属于javax.servlet类包。

在编写一般的Web应用程序时，我们扩展HttpServlet类实现Servlet程序，HttpServlet类中增加了专门对应不同HTTP方法的处理方法，它属于javax.servlet.http类包。



同一般的Java程序不同，Servlet程序没有Main方法，取而代之的是一组Servlet接口或实现了Servlet接口的类的方法，这些方法作为入口，可以被Servlet容器调用，从而使得Servlet程序在Servlet容器的控制下进行，即Servlet程序不能单独运行，只能在Servlet容器中运行。

Servlet是按照请求/响应模式运行，Servlet接口中最重要的一个方法就是service方法，它用来处理客户端请求。每当Servlet容器将一个请求发送到一个Servlet的运行实例时，service方法就被容器调用，service方法可以按照多线程模式运行，可以同时处理多个并发请求。

如果采用直接实现Servlet接口的方式来编写Servlet程序，所作的主要的工作就是实现Service方法，即处理客户请求信息，产生相应响应信息

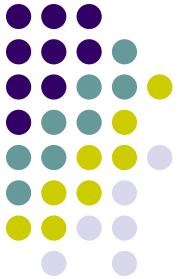
但是，我们往往都是去扩展HttpServlet子类，这个子类中定义了一些附加的方法来辅助HTTP协议的处理，它们会自动地被service方法调用，编程时就不需要在实现service方法，直接实现这些附加方法就可以了。这些方法中最常用的就是doGet方法和doPost方法，分别用于处理HTTP的GET请求和POST请求。

2、Java Servlet API 2.2的类和接口构成

Java Servlet API 2.2的类和接口组成了两个Java类包：Javax.servlet和Javax.servlet.http

Javax.servlet包提供了控制 Servlet 生命周期所必需的 Servlet 接口，是编写 Servlet 时必须要实现的。

javax.servlet.http 包提供了从Servlet 接口派生出的专门用于处理 HTTP 请求的抽象类和一般的工具类。



javax.servlet 包定义的类和接口：

interface RequestDispatcher

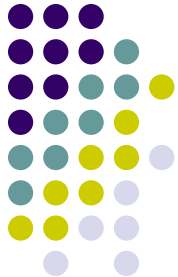
//定义一种对象，用于从客户接受请求，并将请求发送到服务器上任何指定的资源，如一个Servlet 、JSP 或 HTML 文件。

Interface Servlet

// 定 义 了 所 有 Servlet 必 须 实 现 的 方 法 。

interface ServletConfig

//定义Servlet config 对象，由Servlet 引擎用在 Servlet 初始化时，向 Servlet 传递信息。



interface ServletContext

//定义了一系列方法，以便Servlet与其运行的环境通信。

interface ServletRequest

//定义了用于向Servlet传递客户请求信息的对象。

interface ServletResponse

//定义了一个对象，由Servlet用于向客户发送响应。

interface SingleThreadModel

//用于保证Servlet在任一时刻，只处理一个请求。

class GenericServlet

//继承Servlet接口，定义了一个通用的，与协议无关的Servlet。

class ServletInputStream

//定义了一个输入流，用于由Servlet从中读取客户请求的二进制数据。

class ServletOutputStream

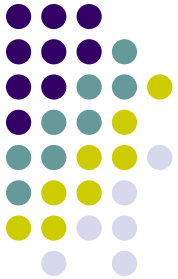
//定义了一个输出流，用于由Servlet向客户发送二进制数据。

class ServletException

//定义了一个当Servlet遇到问题时可以抛出的异常。

class UnavailableException

//定义了一种异常，用于由Servlet指明它永远或暂时不可用。



javax.servlet.http 包定义的类和接口：

interface HttpServletRequest

//继承了ServletRequest 接口，为HTTPServlet 提供请求信息。

interface HttpServletResponse

//继承了ServletResponse 接口,为HTTPServlet 输出响应信息提供支持。

interface HttpSession

//为维护 HTTP 用户的会话状态提供支持。

interface HttpSessionBindingListener

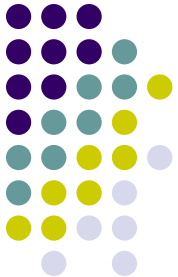
//使得某对象在加入一个会话或从会话中删除时能够得到通知。

interface HttpSessionContext

// 由 Servlet 2.1 定义，该对象在新版本已不被支持。

class Cookie

// 用在Servlet中使用Cookie技术



class HttpServlet

//定义了一个抽象类，继承 GenericServlet 抽象类，应被 HTTPServlet 继承。

class HttpSessionBindingEvent

//定义了一种对象，当某一个实现了HttpSessionBindingListener接口的对象被加入会话或从会话中删除时，会收到该类对象的一个句柄

class HttpUtils

//提供了一系列便于编写HTTPServlet 的方法

3、HTTP Servlet应用编程接口

HTTP Servlet 使用一个 HTML 表格来发送和接收数据。要创建一个 HTTP Servlet，请扩展 HttpServlet 类， 该类是用专门的方法来处理 HTML 表格的 GenericServlet 的一个子类。 HTML 表单是由 <FORM> 和 </FORM> 标记定义的。表单中典型地包含输入字段(如文本输入字段、复选框、单选按钮和选择列表)和用于提交数据的按钮。当提交信息时，它们还指定服务器应执行哪一个 Servlet(或其它的程序)。

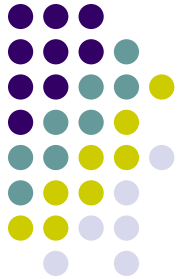
HttpServlet 类包含的主要方法：

(1) init() 方法

在 Servlet 的生命期中，仅执行一次 init() 方法。它是在服务器装入 Servlet 时执行的。 可以配置服务器，以在启动服务器或客户机首次访问 Servlet 时装入 Servlet。 无论有多少客户机访问 Servlet，都不会重复执行 init() 。

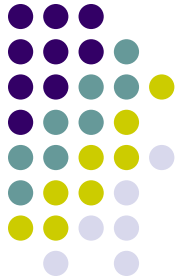
缺省的 init() 方法通常是符合要求的，但也可以用定制 init() 方法来覆盖它，典型的是管理服务器端资源。

例如，初始化数据库连接。缺省的 init() 方法设置了 Servlet 的初始化参数，并用它的 ServletConfig 对象参数来启动配置，因此所有覆盖 init() 方法的 Servlet 应调用 super.init() 以确保仍然执行这些任务。在调用 service() 方法之前，应确保已完成了 init() 方法。



(2) service() 方法

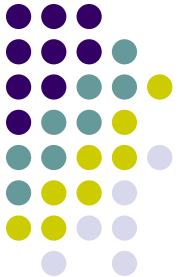
service() 方法是 Servlet 的核心。每当一个客户请求一个 HttpServlet 对象，该对象的service() 方法就要被调用，而且传递给这个方法一个“请求”(ServletRequest)对象和一个“响应"(ServletResponse)对象作为参数。



在 HttpServlet 中已存在 service() 方法。缺省的服务功能是调用与 HTTP 请求的方法相应的 do 功能。例如， 如果 HTTP 请求方法为 GET，则缺省情况下就调用 doGet() 。Servlet 应该为 Servlet 支持的 HTTP 方法覆盖 do 功能。因为 HttpServlet.service() 方法会检查请求方法是否调用了适当的处理方法，不必要覆盖 service() 方法。只需覆盖相应的 do 方法就可以了。

当一个客户通过HTML 表单发出一个HTTP POST请求时，doPost()方法被调用。与POST请求相关的参数作为一个单独的HTTP 请求从浏览器发送到服务器。当需要修改服务器端的数据时，应该使用doPost()方法。

当一个客户通过HTML 表单发出一个HTTP GET请求或直接请求一个URL时，doGet()方法被调用。与GET请求相关的参数添加到URL的后面，并与这个请求一起发送。当不会修改服务器端的数据时，应该使用doGet()方法。



Servlet 的响应可以是下列几种类型：

(1)一个输出流，浏览器根据它的内容类型(如text/HTML)进行解释。

(2)一个HTTP错误响应, 重定向到另一个URL、servlet、JSP。

(3) destroy() 方法

destroy() 方法仅执行一次，即在服务器停止且卸装Servlet时执行该方法。典型的，将 Servlet 作为服务器进程的一部分来关闭。

缺省的 destroy() 方法通常是符合要求的，但也可以覆盖它，典型的是管理服务器端资源。例如，如果 Servlet 在运行时会累计统计数据，则可以编写一个 destroy() 方法，该方法用于在未装入 Servlet 时将统计数字保存在文件中。另一个示例是关闭数据库连接。

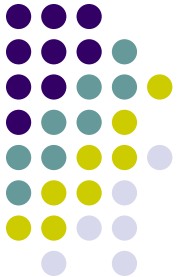
当服务器卸装 Servlet 时，将在所有 service() 方法调用完成后，或在指定的时间间隔过后调用 destroy() 方法。一个Servlet 在运行 service() 方法时可能会产生其它的线程，因此请确认在调用 destroy() 方法时，这些线程已终止或完成。

(4) GetServletConfig()方法

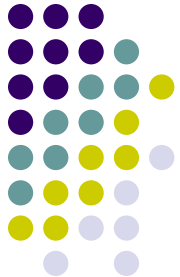
GetServletConfig()方法返回一个 ServletConfig 对象，该对象用来返回初始化参数和 ServletContext 。ServletContext 接口提供有关servlet 的环境信息。

(5) GetServletInfo()方法

GetServletInfo()方法是一个可选的方法，它提供有关servlet 的信息，如作者、版本、版权。



当服务器调用sevlet 的Service()、doGet()和doPost()这三个方法时，均需要“请求”和“响应”对象作为参数。“请求”对象提供有关请求的信息，而“响应”对象提供了一个将响应信息返回给浏览器的一个通信途径。javax.servlet 软件包中的相关类为ServletResponse 和 ServletRequest ，而javax.servlet.http 软件包中的相关类为 HttpServletRequest 和 HttpServletResponse。



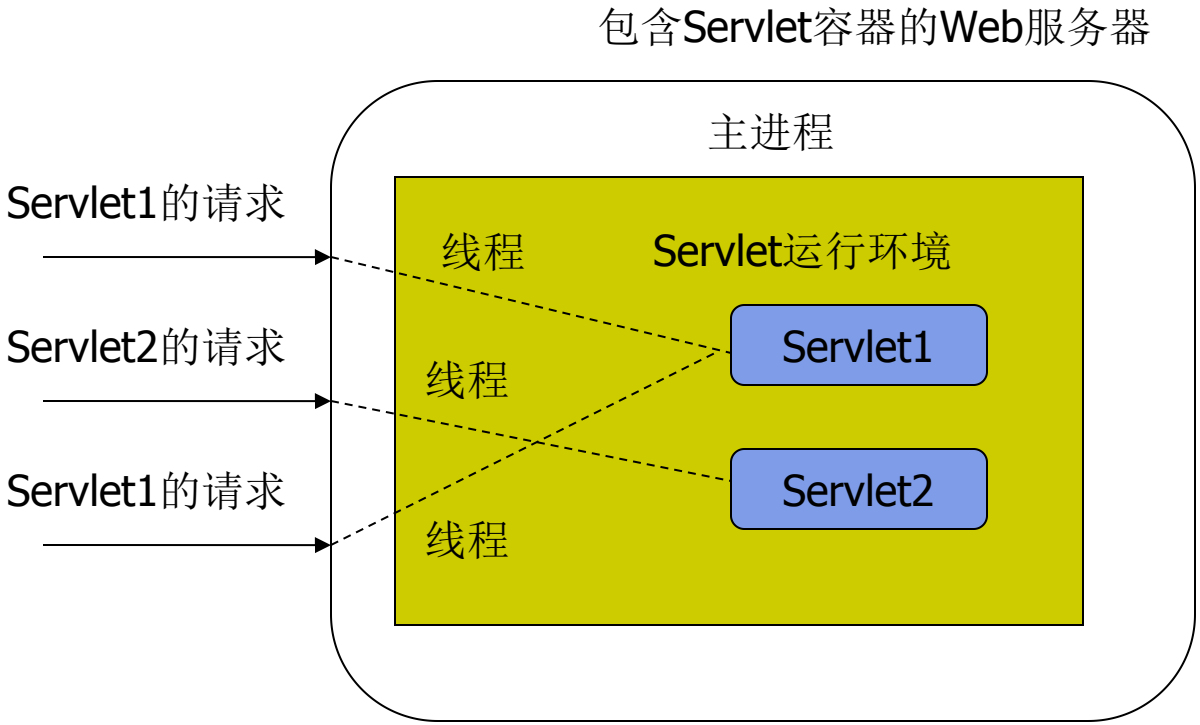
Servlet 通过这些对象与服务器通信并最终与客户机通信。Servlet 能通过调用“请求”对象的方法获知客户机环境，服务器环境的信息和所有由客户机提供的信息。Servlet 可以调用“响应”对象的方法发送响应，该响应是准备发回客户机的。

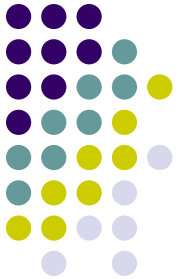
4、Servlet的生命周期

Servlet程序都是在Servlet容器的管理下运行的，Servlet容器按照一个严格定义的生命周期管理Servlet，这个生命周期定义了Servlet如何被加载、实例化、初始化、处理客户端请求、最后结束服务，这个生命周期又称为Servlet生命周期。

生命周期的每一步对应了Servlet接口的不同方法，Servlet程序实现了这些方法，Servlet容器则按照生命周期调用这些方法，管理和运行Servlet程序的运行实例。通常情况下，每个Servlet只有一个运行实例，而通过多个线程来服务多个客户端的访问，当然也不排除有多个运行实例，这取决于容器的具体实现，但是对于Servlet程序来说都是一样的.

Servlet执行模型：





Servlet的生命周期过程：

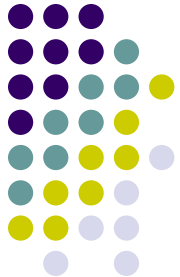
(1) 加载和实例化

Servlet容器负责Servlet的加载和实例化，Servlet容器可以配置成一启动就加载并实例化Servlet，也可以在Servlet第一次被请求时再加载

首先，Servlet容器定位定义了Servlet的Java类，然后使用Java类加载机制加载这个Servlet类，一旦成功，它就从这个Servlet类中实例化一个对象出来。

注意，在一个Servlet容器中，一个给定的Servlet的类可能会有多个实例。这些实例可能会是不同的，例如使用了不同的初始化参数进行实例化，也可能是相同的，例如Servlet是单线程模式时。

如果有多个并发访问，Servlet容器可能会实例化出一组实例，但是无论有多少实例，程序不必担心，那个实例响应请求是由Servlet容器控制的，它们可以代表不同的运行程序。



(2) 初始化

Servlet被加载并被实例化后，在能够响应客户端请求之前，必须进行初始化。容器调用Servlet实例的init方法实现Servlet的初始化，初始化时可以通过ServletConfig对象和ServletContext对象获得有关的Servlet初始化信息和容器信息。

初始化是在正式提供服务之前的一次性操作，因此init方法是实现那些大开销的一次性操作的好地方，例如初始化数据库连接等。

(3) 处理请求（主要过程）

Servlet被正确初始化后，容器可以使用Servlet来响应客户的请求。每个请求由一个ServletRequest类型的对象表示，容器同时创建一个ServletResponse类型的对象表示对应这个请求的响应，如果是HttpServlet，请求和响应对象分别是HttpServletRequest和HttpServletResponse这些对象根据HTTP请求方法分别传递给doGet方法和doPost方法等处理，并要保证线程同步。

(4) 结束服务

Servlet容器不会一直把Servlet实例保存在容器中，当它决定要把Servlet实例从容器中移走时，它就调用Servlet实例的destroy方法，这个了Servlet实例释放占用资源的时机。

destroy方法调用的前提是这个Servlet实例所有正在处理的请求都已经处理完毕了，方法调用完后，Servlet实例就从容器中移走，当然如果客户端再次请求这个Servlet服务，容器就重新实例化Servlet来提供服务。

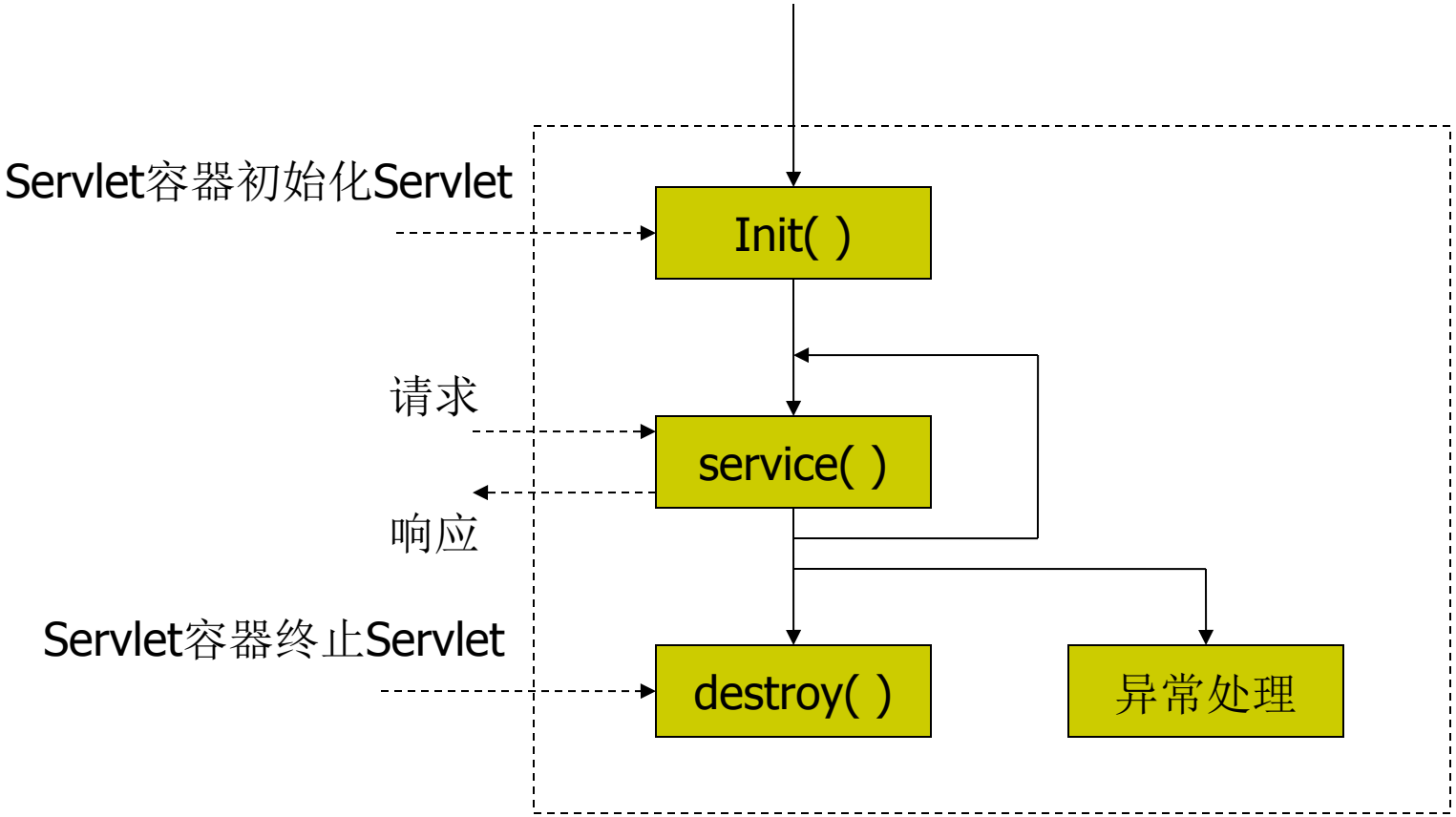
(5) 错误和异常处理

在Servlet生命周期的任何一步都由可能出现错误或者异常，Servlet 会 抛 出 ServletException 异 常 或 者 UnavailableException异常，容器会截获这些异常做出相应的反应。

(6) 保持会话状态

能够在一个客户短的连续访问之间保持会话状态是动态Web应用程序的一个重要的要求，它也是实现许多应用功能的基础，会话状态的保持和管理也是由Servlet容器来实现，Servlet程序可以通过HttpSession类访问会话信息及状态。

Servlet程序的执行过程：



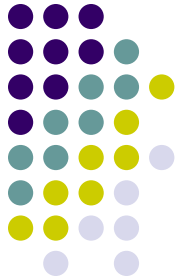
Java Servlet所需的开发环境

进行Servlet开发所需要的基本环境是JSDK以及一个支持Servlet的Web服务器。

1.JSDK(Java Servlet Development Kit)

JSDK包含了编译Servlet应用程序所需要的Java类库以及相关的文档。对于利用Java 1.1进行开发的用户，必须安装JSDK。JSDK已经被集成进Java 1.2 Beta版中，如果利用Java 1.2或以上版本进行开发，则不必安装JSDK。

JSDK可以在Sun公司的站点免费下载，地址是：http://www.sun.com/software/jwebserver/redirect.html

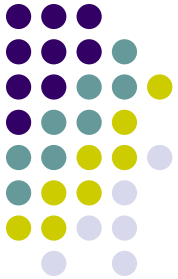


2.支持Servlet的Web服务器

Servlet需要运行在支持Servlet的Web服务器上。目前支持Servlet的Web服务器SUN公司的JSWDK1.0.1。

如果现有的Web服务器不支持Servlet，则可以利用一些第三方厂商的服务器增加件(add - ons)来使Web服务器支持Servlet，例如Live Software公司(http://www.livesoftware.com)提供了一种称为JRun的产品，通过安装JRun的相应版本，可以使Microsoft IIS和Netscape Web Server支持Servlet。

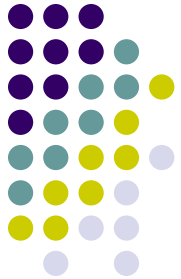
Java Servlet举例



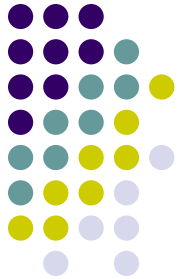
这部分我们通过两个Java Servlet的例子来具体的认识一下Servlet：

1、HelloWorld Servlet(helloWorld.java)

```
Import  java.io.*;
Import  javax.servlet.*;
Import  javax.servlet.http.*;
```

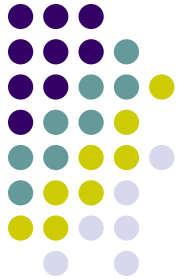


```
/**
*A simple HTTP Servlet that responds to the
*HTTP GET method with a dynamically
*generated HTML Web page.   **/
Public class HelloWorld extends HttpServlet{
    public void doGet (HttpServletRequest request,
                                HttpServletResponse response)
                               throws ServletException,IOException
```



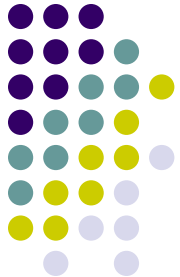
```
{
    response.setContentType(“text/html”);//Set MIME type
//write response data(the Html Web page to return);
    PrintWriter out=response.getWriter();
    out.println(“<html>”)+
 “<head><title>HelloWorld Servlet</title></head>
 “<body><h1>HelloWorld!</h1></body></html>”)
    out.close();
}
}
```

这个Servlet继承HttpServlet抽象类，通过HttpServlet抽象类来获取对HttpServlet定义的标准HTTP功能的访问。该Servlet的doGet方法以上述的请求对象和响应对象作为参数(HttpServletRequest 和 HttpServletResponse)，它通过response对象引用来设置MIME类型，然后，它再使用response对象来构造一个PrintWriter对象。接着，该PrintWriter对象动态产生一个简单的Web页面，此页面再Servlet执行之后返回给客户。

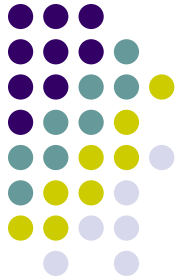


2、CurrentTime Servlet(CurrnetTime.java)

```
import java.io.*;
import javax.Servlet.*;
import javax.Servlet.http.*;
public class CurrentTime extends HttpServlet{
pulbic void doGet(HttpServletRequest request,
HttpServletResponse response) throws IOException,
ServletException
{
```

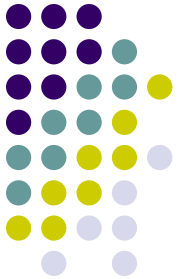


```
response.setContextType(      “text/html;charset=gb231
2”);
PrintWriter out=response.getWriter();
out.println(“<html><body><head>”);
out.println(“<title>显示当前时间</title>”);
out.println(“</head>”);
out.println(“<body><h1>显示当前时间的Servlet</h1>”)
                        java.util.Date dt=new
java.util.Date(System.currentTimeMillis());
```



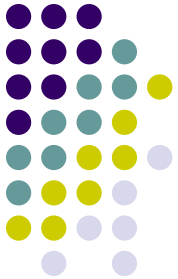
```
    out.print(dt.getHours());
   out.print( ":" );
    out.print(dt.getMinutes());
   out.print( ":" );
    out.println(dt.getSeconds());
   out.println( "</body></html>" );
    }
}
```

开发Java Servlet的过程



开发Java Servlet和开发Java应用程序没什么本质区别，也可以划分为以下三个步骤：

- 创建Servlet（编写）
- 编译Servlet（编译）
- 调用Servlet（测试）



1、创建Servlet：(以HttpServlet为例)

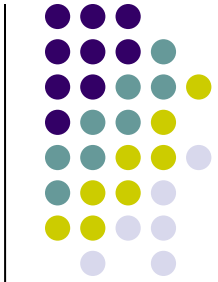
创建一个 HTTP Servlet，通常涉及下列四个步骤

(1)扩展 HttpServlet 抽象类。

(2)重载适当的方法。如覆盖（或称为重写）doGet()或doPost()方法。

(3)如果有 HTTP 请求信息的话，获取该信息。用 HttpServletRequest 对象来检索 HTML 表格所提交的数据或 URL 上的查询字符串。“请求”对象含有特定的方法以检索客户机提供的信息，有3个可用的方法：getParameterNames()、getParameter()和getParameterValues()

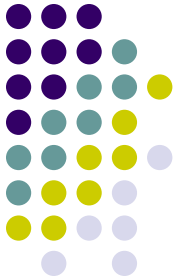
4. **生成** HTTP **响应。**HttpServletResponse **对象生成响应，并将它返回到发出请求的客户机上。它的方法允许设置“请求” 标题和“响应”主体。“响应”对象还含有** getWriter() **方法以返回一个** PrintWriter **对象。使用** PrintWriter **的** print() **和** println() **方法以编写** Servlet **响应来返回给客户机。或者，直接使用**out**对象输出有关**HTML**文档内容。**



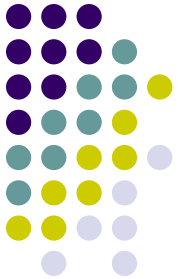
一个简单Servlet的创建过程：

ServletSample.java

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ServletSample extends HttpServlet { // 第一步： 扩展 HttpServlet 抽象类。
```



```
public void doGet (HttpServletRequest request,
HttpServletResponse response)
        throws ServletException, IOException
{
    // 第二步：重写doGet()方法
```



```
String myName = ""; // 第三步：获取HTTP 请求信息
      java.util.Enumeration keys =
request.getParameterNames();
      while (keys.hasMoreElements());
      {
      key = (String) keys.nextElement();
      if (key.equalsIgnoreCase("myName"))
      myName = request.getParameter(key);
      }
      if (myName == "")
      myName = "Hello";
```

```
// 第四步：生成 HTTP 响应。
    response.setContentType("text/html");
    response.setHeader("Pragma", "No-cache");
    response.setDateHeader("Expires", 0);
    response.setHeader("Cache-Control", "no-cache");
   out.println("<head><title>Just a basic
    servlet</title></head> <body>");
    out.println("<h1>Just a basic servlet</h1>");
    out.println ("<p>" + myName + ", this is a very basic servlet
    that writes an HTML page.");
```

```
    out.println ("<p>For instructions on running those samples
    on your WebSphere应用服器,"+
          "open the page:");
    out.println("<pre>http://<em>your.server.name</em>/IB
    MWebAs/samples/index.aspl</pre>");
    out.println("where <em>your.server.name</em> is the
    hostname of your WebSphere应用服务器.");
    out.println("</body></html>");
    out.flush();
    }
}
```

上述ServletSample类扩展 HttpServlet 抽象类、重写doGet()方法。在重写的doGet()方法中，获取HTTP 请求中的一个任选的参数（myName），该参数可作为调用的 URL 上的查询参数传递到 Servlet。可以如下使用示例：

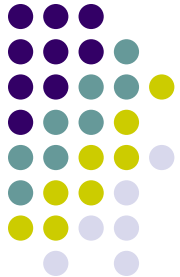
http://your.server.name/servlet/ServletSample?myname=Michael

2、编译Servlet：

我们要用JDK当中的Javac来编译编写好的Servlet，进行语法检查最终生成对应的Servlet的Java类。

3、调用Servlet：

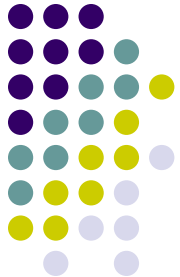
要调用 Servlet 或 Web 应用程序，请使用下列任一种方法：由 URL 调用、在 <FORM> 标记中调用、在 <SERVLET>标记中调用、在 JSP 文件中调用、在 ASP 文件中调用。



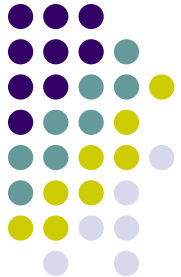
1. 由 URL 调用 Servlet

两种使用 Servlet 的 URL 从浏览器中调用该 Servlet 的方法：

(1) 指定 Servlet 名称：当用 WebSphere应用服务器 管理器来将一个 Servlet 实例添加（注册）到服务器配置中时，必须指定“Servlet 名称”参数的值。



例如，可以指定将hi作为HelloWorldServlet 的 Servlet 名称。要调用该 Servlet，需打开 http://your.server.name/servlet/hi。也可以指定 Servlet 和类使用同一名称（HelloWorldServlet）。在这种情况下，将由 http://your.server.name/servlet/HelloWorldServlet 来调用 Servlet 的实例。

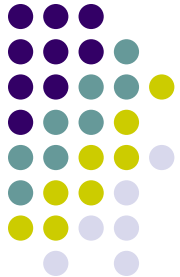


(2) 指定 Servlet 别名：用 WebSphere应用服务器 管理器来配置 Servlet 别名，该别名是用于调用 Servlet 的快捷 URL。快捷 URL 中不包括 Servlet 名称。

2. 在 <FORM> 标记中指定 Servlet

可以在 <FORM> 标记中调用 Servlet。HTML 格式使用户能在 Web 页面（即从浏览器）上输入数据，并向 Servlet 提交数据。例如：

```
<FORM METHOD="GET"
      ACTION="/servlet/myservlet">
<OL>
<INPUT                    TYPE="radio" NAME="broadcast"
VALUE="am">AM<BR>
<INPUT                    TYPE="radio" NAME="broadcast"
VALUE="fm">FM<BR>
</OL>
</FORM>
```

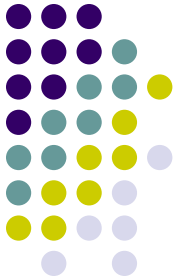


ACTION 特性表明了用于调用 Servlet 的 URL。关于 METHOD 的特性，如果用户输入的信息是通过 GET 方法向 Servlet 提交的，则 Servlet 必须优先使用 doGet() 方法。反之，如果用户输入的信息是通过 POST 方法向 Servlet 提交的，则 Servlet 必须优先使用 doPost() 方法。

3．在 <SERVLET> 标记中指定 Servlet

当使用 <SERVLET> 标记来调用 Servlet 时，如同使用 <FORM> 标记一样，无需创建一个完整的 HTML 页面。作为替代，Servlet 的输出仅是 HTML 页面的一部分，且被动态嵌入到原始 HTML 页面中的其它静态文本中。所有这些都发生在服务器上，且发送给用户的仅是结果 HTML 页面。建议在 Java 服务器页面（JSP）文件中使用 <SERVLET> 标记。请参阅有关 JSP 技术。

原始 HTML 页面中包含 <SERVLET> 和 </SERVLET> 标记。 Servlet 将在这两个标记中被调用，且 Servlet 的响应将覆盖这两个标记间的所有东西和标记本身。如果用户的浏览器可以看到 HTML 源文件，则用户将看不到 <SERVLET> 和 </SERVLET> 标记。



4．在 JSP 文件中调用 Servlet

可以从 JavaServer 页面（JSP）文件中调用 Servlet。请参阅JSP技术部分。

5．在 ASP 文件中调用 Servlet

如果在 Microsoft Internet Information Server（IIS）上有遗留的 ASP 文件，并且无法将 ASP 文件移植成 JSP 文件时，可用 ASP 文件来调用 Servlet。在 WebSphere应用服务器 中的 ASP 支持包括一个用于嵌入 Servlet 的 ActiveX 控制。