Servlet API 中文版--基本类和接口

[一、javax.servlet.Servlet接口 1](#_Toc257637476)

[二、javax.servlet.GenericServlet类（协议无关版本） 1](#_Toc257637477)

[三、javax.servlet.http.HttpServlet类（HTTP版本） 3](#_Toc257637478)

[四、javax.servlet.ServletRequest类（协议无关版本） 4](#_Toc257637479)

[五、javax.servlet.http.HttpServletRequest接口（HTTP版本） 5](#_Toc257637480)

[六、javax.servlet.ServletResponse接口（协议无关版本） 7](#_Toc257637481)

[七、javax.servlet.http.HttpServletResponse接口（HTTP版本） 8](#_Toc257637482)

[八、javax.servlet.ServletContext接口 8](#_Toc257637483)

[九、javax.servlet.http.HttpSession接口 10](#_Toc257637484)

## 一、javax.servlet.Servlet接口

servlet抽象集是javax.servlet.Servlet接口，它规定了必须由Servlet类实现由servlet引擎识别和管理的方法集。

Servlet接口的基本目标是提供生命期方法init()、service()和destroy()方法。

servlet接口中的方法

|  |  |  |
| --- | --- | --- |
| void | init(ServletConfit config)throws ServletException | 在servlet被载入后和实施服务前由servlet引擎进行?次性调用。如果init()产生溢出UnavailableException，则 servle退出服务。 |
| ServletConfig | getServletConfig() | 返回传递到servlet的init()方法的ServletConfig对象 |
| void | service(ServletRequest request, ServletResponse response)throws ServletException,IOException | 处理request对象中描述的请求，使用response对象返回请求结果 |
| String | getServletInfo() | 返回描述servlet的一个字符串 |
| void | destory() | 当servlet将要卸载时由servlet引擎调用 |

## 二、javax.servlet.GenericServlet类（协议无关版本）

GenericServlet是一种与协议无关的servlet，是一种跟本不对请求提供服务的servlet，而是简单地从init()方法启动后台线程并在destory()中杀死。它可以用于模拟操作系统的端口监控进程。

servlet API提供了Servlet接口的直接实现，称为GenericServlet。此类提供除了service()方法外所有接口中方法的缺省实现。这意味着通过简单地扩展GenericServlte可以编写一个基本的servlet。

除了Servlet接口外，GenericServlet也实现了ServletConfig接口，处理初始化参数和servlet上下文，提供对授权传递到init()方法中的ServletConfig对象的方法。

GenericServlet类中的方法

|  |  |  |
| --- | --- | --- |
| void | destory() | 编飞组成单词“destory”的一个注册入口 |
| String | getInitParameter(String name) | 返回具有指定名称的初始化参数值。通过凋用config.getInitParameter(name)实现。 |
| Enumeration | getInitParameterNames() | 返回此serv]et已编码的所有初始化参数的?个枚举类型值。调用config.getInitParameterNames()获得列表。如果未提供初始化参数，则返回?个空的枚举类型值（但不是null） |
| ServletConfig | getServletConfig() | 返回传递到init()方法的ServletConfig对象 |
| ServletContext | getServletContext() | 返回在config对象中引用的ServletContext |
| String | getServletInfo() | 返回空字符串 |
| void | init(ServletConfig config)throws ServletException | 在一实例变量中保存config对象。编写组成单词“init”的注册入口,然后调用方法init() |
| void | init()throws ServletException | 可以被跳过以处理servlet初始化.在config对象被保存后init(ServletConfig config)的结尾处自动被调用.servlet作者经常会忘记调用super.init(config) |
| void | log(String msg) | 编写注册servlet的入口.为此调用servlet上下文的log()方法.servlet的名字被加到消息文本的开头。 |
| void | log(String msg,Throwable t) | 编写一个入口和servlet注册的栈轨迹。此方法也是ServletContext中相应方法的一个副本。 |
| abstract | void service(Request request,Response response)throws ServletException,IOException | 由servlet引擎调用为请求对象描述的请求提供服务。这是GenericServlet中唯一的抽象方法。因此它也是唯一必须被子类所覆盖的方法。 |
| String | getServletName() | 返回在Web应用发布描述器（web.xml）中指定的servlet的名字。 |

## 三、javax.servlet.http.HttpServlet类（HTTP版本）

虽然servlet API允许扩展到其它协议，但最终所有的servlet均在Web环境下实施操作，只有几种servlet直接扩展了GenericServlet。对servlet更一般的是扩展其HTTP子类HttpServlet。

HttpServlet 类通过调用指定到HTTP请求方法的方法实现service()，亦即对DELETE、HEAD、GET、OPTIONS、POST、PUT和 TRACE，分别调用doDelete()、doHead()、doGet()、doOptions()、doPost()、doPut()和 doTrace()方法，将请求和响应对象置入其HTTP指定子类。

HttpServlet类中的方法

|  |  |  |
| --- | --- | --- |
| Void | doGet(HttpServletRequest request HttpServletResponse response)throws ServletException IOException | 由servlet引擎调用用处理一个HTTP GET请求。输入参数、HTTP头标和输入流可从request对象、response头标和response对象的输出流中获得。 |
| Void | doPost(HttpServletRequest request HttpServletResponse response)throws ServletException IOException | 由servlet引擎调用用处理一个HTTP POST请求。输入参数、HTTP头标和输入流可从request对象、response头标和response对象的输出流中获得。 |
| Void | doPut(HttpServletRequest request HttpServletResponse response)throws ServletException IOException | 由servlet引擎调用用处理一个HTTP PUT请求。本方法中请求URI指出被载入的文件位置。 |
| Void | doDelete(HttpServletRequest request HttpServletResponse response)throws ServletException IOException | 由servlet引擎调用用处理一个HTTP DELETE请求。请求URI指出资源被删除。 |
| Void | doOptions(HttpServletRequest request HttpServletResponse response)throws ServletException IOException | 由servlet引擎调用用处理一个HTTP OPTIONS请求。返回一个Allow响应头标表明此servlet支持的HTTP方法。一个servlet不需要覆盖此方法，因为 HttpServlet方法已经实现规范所需的功能。 |
| Void | doTrace(HttpServletRequest request HttpServletResponse response)throws ServletException, IOException | 由servlet引擎调用用处理一个HTTP TRACE请求。使得请求头标被反馈成响应关标。一个servlet不需要覆盖此方法，因为HttpServlet方法已经实现HTTP规范所需的功能。 |
| Void | service(HttpServletRequest request HttpServletResponse response)throws ServletException, IOException | Service(Request request Response response)调用的一个立即方法，带有指定HTTP请求和响应。此方法实际上将请求导向doGet()、doPost()等等。不应该覆盖此方法。 |
| Void | service(Request request Response response)throws ServletException IOException | 将请求和响应对象置入其指定的HTTP子类，并调用指定HTTP的service()方法。 |

## 四、javax.servlet.ServletRequest类（协议无关版本）

ServletRequest接口封装了客户端请求的细节。它与协议无关，并有一个指定HTTP的子接口。

ServletRequest主要处理：

1.找到客户端的主机名和IP地址

2.检索请求参数

3.取得和设置属性

4.取得输入和输出流

ServletRequest类中的方法

|  |  |  |
| --- | --- | --- |
| Object | getAttribute(String name) | 返回具有指定名字的请求属性，如果不存在则返回null。属性可由servlet引擎设置或使用setAttribute()显式加入。 |
| Enumeration | getAttributeName() | 返回请求中所有属性名的枚举。如果不存在属性，则返回一个空的枚举。 |
| String | getCharacteEncoding() | 返回请求所用的字符编码。 |
| Int | getContentLength() | 指定输入流的长度，如果未知则返回-1。 |
| ServletInputStream | getInputStream()throws IOException | 返回与请求相关的（二进制）输入流。可以调用getInputStream()或getReader()方法之一。 |
| String | getParameter(String name) | 返回指定输入参数，如果不存在，返回null。 |
| Enumeration | getParameterName() | 返回请求中所有参数名的一个可能为空的枚举。 |
| String[] | getParameterValues(String name) | 返回指定输入参数名的取值数组，如果取值不存在则返回null。它在参数具有多个取值的情况下十分有用。 |
| String | get Protocol() | 返回请求使用协议的名称和版本。 |
| String | getScheme() | 返回请求URI的子串，但不包含第一个冒号前的内容。 |
| String | getServerName() | 返回处理请求的服务器的主机名。 |
| String | getServerPort() | 返回接收主机正在侦听的端口号。 |
| BufferedReader | getReader()throws IOException | 返回与请求相关输入数据的一个字符解读器。此方法与getInputStream()只可分别调用，不能同时使用。 |
| String | getRemoteAddr() | 返回客户端主机的数字型IP地址。 |
| String | getRemoteHost() | 如果知道，返回客户端主机名。 |
| void | setAttribute(String name,Object obj) | 以指定名称保存请求中指定对象的引用。 |
| void | removeAttribute(String name) | 从请求中删除指定属性 |
| Locale | getLocale() | 如果已知，返回客户端的第一现场或者为null。 |
| Enumeration | getLocales() | 如果已知，返回客户端的第一现场的一个枚举，否则返回服务器第一现场。 |
| boolean | isSecure() | 如果请求使用了如HTTPS安全隧道，返回true |
| RequestDispatcher | getRequestDispatcher(String name) | 返回指定源名称的RequsetDispatcher对象。 |

## 五、javax.servlet.http.HttpServletRequest接口（HTTP版本）

HttpServletRequest类主要处理：

1.读取和写入HTTP头标

2.取得和设置cookies

3.取得路径信息

4.标识HTTP会话

HttpServletRequest接口中的方法

|  |  |  |
| --- | --- | --- |
| String | getAuthType() | 如果servlet由一个鉴定方案所保护，如HTTP基本鉴定，则返回方案名称。 |
| String | getContextPath() | 返回指定servlet上下文（web应用）的URL的前缀。 |
| Cookie[] | getCookies() | 返回与请求相关cookie的一个数组。 |
| Long | getDateHeader(String name) | 将输出转换成适合构建Date对象的long类型取值的getHeader()的简化版。 |
| String | getHeader(String name) | 返回指定的HTTP头标指。如果其由请求给出，则名字应为大小写不敏感。 |
| Enumeration | getHeaderNames() | 返回请求给出的所有HTTP头标名称的权举值。 |
| Enumeration | getHeaders(String name) | 返回请求给出的指定类型的所有HTTP头标的名称的枚举值，它对具有多取值的头标非常有用。 |
| int | getIntHeader(String name) | 将输出转换为int取值的getHeader()的简化版。 |
| String | getMethod() | 返回HTTP请求方法（例如GET、POST等等） |
| String | getPathInfo() | 返回在URL中指定的任意附加路径信息。 |
| String | getPathTranslated() | 返回在URL中指定的任意附加路径信息，被子转换成一个实际路径。 |
| String | getQueryString() | 返回查询字符串，即URL中?后面的部份。 |
| String | getRemoteUser() | 如果用户通过鉴定，返回远程用户名，否则为null。 |
| String | getRequestedSessionId() | 返回客户端的会话ID |
| String | getRequestURI() | 返回URL中一部分，从“/”开始，包括上下文，但不包括任意查询字符串。 |
| String | getServletPath() | 返回请求URI上下文后的子串 |
| HttpSession | getSession() | 调用getSession(true)的简化版。 |
| HttpSession | getSession(boolean create) | 返回当前HTTP会话，如果不存在，则创建一个新的会话，create参数为true。 |
| Principal | getPrincipal() | 如果用户通过鉴定，返回代表当前用户的java.security.Principal对象，否则为null。 |
| boolean | isRequestedSessionIdFromCookie() | 如果请求的会话ID由一个Cookie对象提供，则返回true，否则为false。 |
| boolean | isRequestedSessionIdFromURL() | 如果请求的会话ID在请求URL中解码，返回true，否则为false |
| boolean | isRequestedSessionIdValid() | 如果客户端返回的会话ID仍然有效，则返回true。 |
| Boolean | isUserInRole(String role) | 如果当前已通过鉴定用户与指定角色相关，则返回true，如果不是或用户未通过鉴定，则返回false。 |

## 六、javax.servlet.ServletResponse接口（协议无关版本）

ServletResponse对象将一个servlet生成的结果传到发出请求的客户端。ServletResponse操作主要是作为输出流及其内容类型和长度的包容器，它由servlet引擎创建.

ServletResponse接口中的方法

|  |  |  |
| --- | --- | --- |
| void | flushBuffer()throws IOException | 发送缓存到客户端的输出内容。因为HTTP需要头标在内容前被发送，调用此方法发送状态行和响应头标，以确认请求。 |
| int | getBufferSize() | 返回响应使用的缓存大小。如果缓存无效则返加0。 |
| String | getCharacterEncoding() | 返回响应使用字符解码的名字。除非显式设置，否则为ISO-8859-1 |
| Locale | getLocale() | 返回响应使用的现场。除非用setLocale()修改，否则缺省为服务器现场。 |
| OutputStream | getOutputStream()throws IOException | 返回用于将返回的二进制输出写入客户端的流，此方法和getWrite()方法二者只能调用其一。 |
| Writer | getWriter()throws IOException | 返回用于将返回的文本输出写入客户端的一个字符写入器，此方法和getOutputStream()二者只能调用其一。 |
| boolean | isCommitted() | 如果状态和响应头标已经被发回客户端，则返回true，在响应被确认后发送响应头标毫无作用。 |
| void | reset() | 清除输出缓存及任何响应头标。如果响应已得到确认，则引发事件IllegalStateException。 |
| void | setBufferSize(int nBytes) | 设置响应的最小缓存大小。实际缓存大小可以更大，可以通过调用getBufferSize()得到。如果输出已被写入，则产生IllegalStateException。 |
| void | setContentLength(int length) | 设置内容体的长度。 |
| void | setContentType(String type) | 设置内容类型。在HTTP servlet中即设置Content-Type头标。 |
| void | setLocale(Locale locale) | 设置响应使用的现场。在HTTP servlet中，将对Content-Type头标取值产生影响。 |

## 七、javax.servlet.http.HttpServletResponse接口（HTTP版本）

HttpServletResponse加入表示状态码、状态信息和响应头标的方法，它还负责对URL中写入一Web页面的HTTP会话ID进行解码。

HttpServletResponse接口中的方法

|  |  |  |
| --- | --- | --- |
| void | addCookie(Cookie cookie) | 将一个Set-Cookie头标加入到响应。 |
| void | addDateHeader(String name,long date) | 使用指定日期值加入带有指定名字（或代换所有此名字头标）的响应头标的方法。 |
| void | setHeader(String name,String value) | 设置具有指定名字和取值的一个响应头标。 |
| void | addIntHeader(String name,int value) | 使用指定整型值加入带有指定名字的响应头标（或代换此名字的所有头标）。 |
| boolean | containsHeader(String name) | 如果响应已包含此名字的头标，则返回true。 |
| String | encodeRedirectURL(String url) | 如果客户端不知道接受cookid，则向URL加入会话ID。第一种形式只对在sendRedirect()中使用的URL进行调用。其他被编码的URLs应被传递到encodeURL() |
| String | encodeURL(String url) |  |
| void | sendError(int status) | 设置响应状态码为指定值（可选的状态信息）。HttpServleetResponse定义了一个完整的整数常量集合表示有效状态值。 |
| void | sendError(int status,String msg) |  |
| void | setStatus(int status) | 设置响应状态码为指定指。只应用于不产生错误的响应，而错误响应使用sendError()。 |

## 八、javax.servlet.ServletContext接口

一个servlet上下文是servlet引擎提供用来服务于Web应用的接口。Servlet上下文具有名字（它属于Web应用的名字）唯一映射到文件系统的一个目录。

一个servlet可以通过ServletConfig对象的getServletContext()方法得到servlet上下文的引用，如果servlet直接或间接调用子类GenericServlet，则可以使用getServletContext()方法。

Web应用中servlet可以使用servlet上下文得到：

1.在调用期间保存和检索属性的功能，并与其他servlet共享这些属性。

2.读取Web应用中文件内容和其他静态资源的功能。

3.互相发送请求的方式。

4.记录错误和信息化消息的功能。

ServletContext接口中的方法

|  |  |  |
| --- | --- | --- |
| Object | getAttribute(String name) | 返回servlet上下文中具有指定名字的对象，或使用已指定名捆绑一个对象。从Web应用的标准观点看，这样的对象是全局对象，因为它们可以被同一servlet在另一时刻访问。或上下文中任意其他servlet访问。 |
| void | setAttribute(String name,Object obj) | 设置servlet上下文中具有指定名字的对象。 |
| Enumeration | getAttributeNames() | 返回保存在servlet上下文中所有属性名字的枚举。 |
| ServletContext | getContext(String uripath) | 返回映射到另一URL的servlet上下文。在同一服务器中URL必须是以“/”开头的绝对路径。 |
| String | getInitParameter(String name) | 返回指定上下文范围的初始化参数值。此方法与ServletConfig方法名称不一样，后者只应用于已编码的指定servlet。此方法应用于上下文中所有的参数。 |
| Enumeration | getInitParameterNames() | 返回（可能为空）指定上下文范围的初始化参数值名字的枚举值。 |
| int | getMajorVersion() | 返回此上下文中支持servlet API级别的最大和最小版本号。 |
| int | getMinorVersion() |  |
| String | getMimeType(String fileName) | 返回指定文件名的MIME类型。典型情况是基于文件扩展名，而不是文件本身的内容（它可以不必存在）。如果MIME类型未知，可以返回null。 |
| RequestDispatcher | getNameDispatcher(String name) | 返回具有指定名字或路径的servlet或JSP的RequestDispatcher。如果不能创建RequestDispatch，返回null。如果指定路径，必须心“/”开头，并且是相对于servlet上下文的顶部。 |
| RequestDispatcher | getNameDispatcher(String path) |  |
| String | getRealPath(String path) | 给定一个URI，返回文件系统中URI对应的绝对路径。如果不能进行映射，返回null。 |
| URL | getResource(String path) | 返回相对于servlet上下文或读取URL的输入流的指定绝对路径相对应的URL，如果资源不存在则返回null。 |
| InputStream | getResourceAsStream(String path) |  |
| String | getServerInfo() | 返顺servlet引擎的名称和版本号。 |
| void | log(String message) |  |
| void | log(String message,Throwablet) | 将一个消息写入servlet注册，如果给出Throwable参数，则包含栈轨迹。 |
| void | removeAttribute(String name) | 从servlet上下文中删除指定属性。 |

## 九、javax.servlet.http.HttpSession接口

HttpSession类似于哈希表的接口，它提供了setAttribute ()和getAttribute()方法存储和检索对象。HttpSession提供了一个会话ID关键字，一个参与会话行为的客户端在同一会话的请求中存储和返回它。servlet引擎查找适当的会话对象，并使之对当前请求可用。

HttpSession接口中的方法

|  |  |  |
| --- | --- | --- |
| Object | getAttribute(String name) | 将会话中一个对象保存为指定名字，返回或删除前面保存的此名称对象。 |
| void | setAttribute(String name,Object value) |  |
| void | removeAttribute(String name) |  |
| Enumeration | getAttributeName() | 返回捆绑到当前会话的所有属性名的枚举值。 |
| long | getCreationTime() | 返回表示会话创建和最后访问日期和时间的一个长整型，该整型形式为java.util.Date()构造器中使用的形式。 |
| long | getLastAccessedTime() |  |
| String | getId() | 返回会话ID，servlet引擎设置的一个唯一关键字。 |
| ing | getMaxInactiveInterval() | 如果没有与客户端发生交互，设置和返回会话存活的最大秒数。 |
| void | setMasInactiveInterval(int seconds) |  |
| void | invalidate() | 使得会话被终止，释放其中任意对象。 |
| boolean | isNew() | 如果客户端仍未加入到会话，返回true。当会话首次被创建，会话ID被传入客户端，但客户端仍未进行包含此会话ID的第二次请示时，返回true。 |

API对象的说明

这一部分包含了对Java Servlet API的全部类和接口的详细说明。这个说明与Javadoc API差不多，但是这份文档提供了更多的信息。

API包含了两个软件包，十二个接口和九个类。

软件包：javax.servlet

所包含的接口：RequestDispatcher；Servlet；ServletConfig；ServletContext；ServletRequest；ServletResponse；SingleThreadModel。

所包含的类：GenericServlet；ServletInputStream；ServletOutputStream；ServletException；UnavailableException。

一、RequestDispatcher接口：

定义：

public interface RequestDispatcher;

定义一个对象，从客户端接收请求，然后将它发给服务器的可用资源（例如Servlet、CGI、HTML文件、JSP文件）。Servlet引擎创建request dispatcher对象，用于封装由一个特定的URL定义的服务器资源。

这个接口是专用于封装Servlet的，但是一个Servlet引擎可以创建request dispatcher对象用于封装任何类型的资源。

request dispatcher对象是由Servlet引擎建立的，而不是由Servlet开发者建立的。

方法

1、forward

public void forward(ServletRequest request, ServletReponse response)

throws ServletException, IOException;

被用来从这个Servlet向其它服务器资源传递请求。当一个Servlet对响应作了初步的处理，并要求其它的对象对此作出响应时，可以使用这个方法。

当request对象被传递到目标对象时，请求的URL路径和其他路径参数会被调整为反映目标对象的目标URL路径。

如果已经通过响应返回了一个ServletOutputStream对象或PrintWriter对象，这个方法将不能使用，否则，这个方法会抛出一个IllegalStateException。

2、include

public void include(ServletRequest request, ServletResponse response)

throws ServletException, IOException

用来包括发送给其他服务器资源的响应的内容。本质上来说，这个方法反映了服务器端的内容。

请求对象传到目标对象后会反映调用请求的请求URL路径和路径信息。这个响应对象只能调用这个Servlet的ServletOutputStream对象和PrintWriter对象。

一个调用include的Servlet不能设置头域，如果这个Servlet调用了必须设置头域的方法（例如cookie），这个方法将不能保证正常使用。作为一个Servlet开发者，你必须妥善地解决那些可能直接存储头域的方法。例如，即使你使用会话跟踪，为了保证session的正常工作，你必须在一个调用include的Servlet之外开始你的session

二、Servlet接口。

定义

public interface Servlet

这个接口定义了一个Servlet：一个在Web服务器上继承了这个功能的Java类。

方法

1、init

public void init(ServletConfig config) throws ServletException;

Servlet引擎会在Servlet实例化之后，置入服务之前精确地调用init方法。在调用service方法之前，init方法必须成功退出。

如果init方法抛出一个ServletException，你不能将这个Servlet置入服务中，如果init方法在超时范围内没完成，我们也可以假定这个Servlet是不具备功能的，也不能置入服务中。

2、service

public void service(ServletRequest request, ServletResponse response)

throws ServletException, IOException;

Servlet引擎调用这个方法以允许Servlet响应请求。这个方法在Servlet未成功初始化之前无法调用。在Servlet被初始化之前，Servlet引擎能够封锁未决的请求。

在一个Servlet对象被卸载后，直到一个新的Servelt被初始化，Servlet引擎不能调用这个方法

3、destroy

public void destroy();

当一个Servlet被从服务中去除时，Servlet引擎调用这个方法。在这个对象的service方法所有线程未全部退出或者没被引擎认为发生超时操作时，destroy方法不能被调用。

4、getServletConfig

public ServletConfig getServletConfig();

返回一个ServletConfig对象，作为一个Servlet的开发者，你应该通过init方法存储ServletConfig对象以便这个方法能返回这个对象。为了你的便利，GenericServlet在执行这个接口时，已经这样做了。

5、getServletInfo

public String getServletInfo();

允许Servlet向主机的Servlet运行者提供有关它本身的信息。返回的字符串应该是纯文本格式而不应有任何标志（例如HTML，XML等）。

三、ServletConfig接口

定义

public interface ServletConfig

这个接口定义了一个对象，通过这个对象，Servlet引擎配置一个Servlet并且允许Servlet获得一个有关它的ServletContext接口的说明。每一个ServletConfig对象对应着一个唯一的Servlet。

方法

1、getInitParameter

public String getInitParameter(String name);

这个方法返回一个包含Servlet指定的初始化参数的String。如果这个参数不存在，返加空值。

2、getInitParameterNames

public Enumeration getInitParameterNames();

这个方法返回一个列表String对象，该对象包括Servlet的所有初始化参数名。如果Servlet没有初始化参数，getInitParameterNames返回一个空的列表。

3、getServletContext

public ServletContext getServletContext();

返回这个Servlet的ServletContext对象。

四、ServletContext接口

定义

public interface ServletContext

定义了一个Servlet的环境对象，通过这个对象，Servlet引擎向Servlet提供环境信息。

一个Servlet的环境对象必须至少与它所驻留的主机是一一对应的。在一个处理多个虚拟主机的Servlet引擎中（例如，使用了HTTP1.1的主机头域），每一个虚拟主机必须被视为一个单独的环境。此外，Servlet引擎还可以创建对应于一组Servlet的环境对象。

方法

1、getAttribute

public Object getAttribute(String name);

返回Servlet环境对象中指定的属性对象。如果该属性对象不存在，返回空值。这个方法允许访问有关这个Servlet引擎的在该接口的其他方法中尚未提供的附加信息。

2、getAttributeNames

public Enumeration getAttributeNames();

返回一个Servlet环境对象中可用的属性名的列表。

3、getContext

public ServletContext getContext(String uripath);

返回一个Servlet环境对象，这个对象包括了特定URI路径的Servlets和资源，如果该路径不存在，则返回一个空值。URI路径格式是/dir/dir/filename.ext。

为了安全，如果通过这个方法访问一个受限制的Servlet的环境对象，会返回一个空值。

4、getMajorVersion

public int getMajorVersion();

返回Servlet引擎支持的Servlet API的主版本号。例如对于2.1版，这个方法会返回一个整数2。

5、getMinorVersion

public int getMinorVersion();

返回Servlet引擎支持的Servlet API的次版本号。例如对于2.1版，这个方法会返回一个整数2。

6、getMimeType

public String getMimeType(String file);

返回指定文件的MIME类型，如果这种MIME类型未知，则返回一个空值。MIME类型是由Servlet引擎的配置决定的。

7、getRealPath

public String getRealPath(String path);

一个符合URL路径格式的指定的虚拟路径的格式是：/dir/dir/filename.ext。用这个方法，可以返回与一个符合该格式的虚拟路径相对应的真实路径的String。这个真实路径的格式应该适合于运行这个Servlet引擎的计算机（包括其相应的路径解析器）。

不管是什么原因，如果这一从虚拟路径转换成实际路径的过程不能执行，该方法将会返回一个空值。

8、getResource

public URL getResource(String uripath);

返回一个URL对象，该对象反映位于给定的URL地址（格式：/dir/dir/filename.ext）的Servlet环境对象已知的资源。无论URLStreamHandlers对于访问给定的环境是不是必须的，Servlet引擎都必须执行。如果给定的路径的Servlet环境没有已知的资源，该方法会返回一个空值。

这个方法和java.lang.Class的getResource方法不完全相同。java.lang.Class的getResource方法通过装载类来寻找资源。而这个方法允许服务器产生环境变量给任何资源的任何Servlet，而不必依赖于装载类、特定区域等等。

9、getResourceAsStream

public InputStream getResourceAsStream(String uripath);

返回一个InputStream对象，该对象引用指定的URL的Servlet环境对象的内容。如果没找到Servlet环境变量，就会返回空值，URL路径应该具有这种格式：/dir/dir/filename.ext。

这个方法是一个通过getResource方法获得URL对象的方便的途径。请注意，当你使用这个方法时，meta-information（例如内容长度、内容类型）会丢失。

10、getRequestDispatcher

public RequestDispatcher getRequestDispatcher(String uripath);

如果这个指定的路径下能够找到活动的资源(例如一个Servlet，JSP页面，CGI等等)就返回一个特定URL的RequestDispatcher对象，否则，就返回一个空值，Servlet引擎负责用一个request dispatcher对象封装目标路径。这个request dispatcher对象可以用来完全请求的传送。

11、getServerInfo

public String getServerInfo();

返回一个String对象，该对象至少包括Servlet引擎的名字和版本号。

12、log

public void log(String msg);

public void log(String msg, Throwable t);

public void log(Exception exception, String msg); // 这种用法将被取消

写指定的信息到一个Servlet环境对象的log文件中。被写入的log文件由Servlet引擎指定，但是通常这是一个事件log。当这个方法被一个异常调用时，log中将包括堆栈跟踪。

13、setAttribute

public void setAttribute(String name, Object o);

给予Servlet环境对象中你所指定的对象一个名称。

14、removeAttribute

public void removeAttribute(String name);

从指定的Servlet环境对象中删除一个属性。

注：以下几个方法将被取消

15、getServlet

public Servlet getServlet(String name) throws ServletException;

最初用来返回一个指定名称的Servlet，如果没找到就返回一个空值。如果这个Servlet能够返回，这就意味着它已经被初始化，而且已经可以接受service请求。这是一个危险的方法。当调用这个方法时，可能并不知道Servlet的状态，这就可能导致有关服务器状态的问题。而允许一个Servlet访问其他Servlet的这个方法也同样的危险。

现在这个方法返回一个空值，为了保持和以前版本的兼容性，现在这个方法还没有被取消。在以后的API版本中，该方法将被取消。

16、getServletNames

public Enumeration getServletNames();

最初用来返回一个String对象的列表，该列表表示了在这个Servlet环境下所有已知的Servlet对象名。这个列表总是包含这个Servlet自身。

基于与上一个方法同样的理由，这也是一个危险的方法。

现在这个方法返回一个空的列表。为了保持和以前版本的兼容性，现在这个方法还没有被取消。在以后的API版本中，该方法将被取消。

17、getServlets

public Enumeration getServlets();

最初用来返回在这个Servelet环境下所有已知的Servlet对象的列表。这个列表总是包含这个Servlet自身。

基于与getServlet方法同样的理由，这也是一个危险的方法。

现在这个方法返回一个空的列表。为了保持和以前版本的兼容性，现在这个方法还没有被取消。在以后的API版本中，该方法将被取消。

五、ServletRequest接口

定义

public interface ServletRequest

定义一个Servlet引擎产生的对象，通过这个对象，Servlet可以获得客户端请求的数据。这个对象通过读取请求体的数据提供包括参数的名称、值和属性以及输入流的所有数据。

方法

1、getAttribute

public Object getAttribute(String name);

返回请求中指定属性的值，如果这个属性不存在，就返回一个空值。这个方法允许访问一些不提供给这个接口中其他方法的请求信息以及其他Servlet放置在这个请求对象内的数据。

2、getAttributeNames

public Enumeration getAttributeNames();

返回包含在这个请求中的所有属性名的列表。

3、getCharacterEncoding

public String getCharacterEncoding();

返回请求中输入内容的字符编码类型，如果没有定义字符编码类型就返回空值。

4、getContentLength

public int getContentLength();

请求内容的长度，如果长度未知就返回-1。

5、getContentType

public String getContentType();

返回请求数据体的MIME类型，如果类型未知返回空值。

6、getInputStream

public ServletInputStream getInputStream() throws IOException;

返回一个输入流用来从请求体读取二进制数据。如果在此之前已经通过getReader方法获得了要读取的结果，这个方法会抛出一个IllegalStateException。

7、getParameter

public String getParameter(String name);

以一个String返回指定的参数的值，如果这个参数不存在返回空值。例如，在一个HTTP Servlet中，这个方法会返回一个指定的查询语句产生的参数的值或一个被提交的表单中的参数值。如果一个参数名对应着几个参数值，这个方法只能返回通过getParameterValues方法返回的数组中的第一个值。因此，如果这个参数有（或者可能有）多个值，你只能使用getParameterValues方法。

8、getParameterNames

public Enumeration getParameterNames();

返回所有参数名的String对象列表，如果没有输入参数，该方法返回一个空值。

9、getParameterValues

public String[] getParameterValues(String name);

通过一个String对象的数组返回指定参数的值，如果这个参数不存在，该方法返回一个空值。

10、getProtocol

public String getProtocol();

返回这个请求所用的协议，其形式是协议/主版本号.次版本号。例如对于一个HTTP1.0的请求，该方法返回HTTP/1.0。

11、getReader

public BufferedReader getReader() throws IOException;

这个方法返回一个buffered reader用来读取请求体的实体，其编码方式依照请求数据的编码方式。如果这个请求的输入流已经被getInputStream调用获得，这个方法会抛出一个IllegalStateException。

12、getRemoteAddr

public String getRemoteAddr();

返回发送请求者的IP地址。

13、getRemoteHost

public String getRemoteHost();

返回发送请求者的主机名称。如果引擎不能或者选择不解析主机名（为了改善性能），这个方法会直接返回IP地址。

14、getScheme

public String getScheme();

返回请求所使用的URL的模式。例如，对于一个HTTP请求，这个模式就是http。

15、getServerName

public String getServerName();

返回接收请求的服务器的主机名。

16、getServerPort

public int getServerPort();

返回接收请求的端口号。

17、setAttribute

public void setAttribute(String name, Object object);

这个方法在请求中添加一个属性，这个属性可以被其他可以访问这个请求对象的对象（例如一个嵌套的Servlet）使用。

注：以下方法将被取消

getRealPath

public String getRealPath(String path);

返回与虚拟路径相对应的真实路径，如果因为某种原因，这一过程不能进行，该方法将返回一个空值。

这个方法和ServletContext接口中的getRealPath方法重复。在2.1版中，ServletContext接口将阐明一个Servlet所能用的所有的路径的映射。该方法执行的结果将会与ServletContext中getRealPath方法的结果完全一样。

六、ServletResponse接口

定义

public interface ServletResponse

定义一个Servlet引擎产生的对象，通过这个对象，Servlet对客户端的请求作出响应。这个响应应该是一个MIME实体，可能是一个HTML页、图象数据或其他MIME的格式。

方法

1、getCharacterEncoding

public String getCharacterEncoding();

返回MIME实体的字符编码。这个字符编码可以是指定的类型，也可以是与请求头域所反映的客户端所能接受的字符编码最匹配的类型。在HTTP协议中，这个信息被通过Accept-Charset传送到Servlet引擎。

有关字符编码和MIME的更多信息请参看RFC 2047。

2、getOutputStream

public ServletOutputStream getOutputStream() throws IOException;

返回一个记录二进制的响应数据的输出流。

如果这个响应对象已经调用getWriter，将会抛出IllegalStateException。

3、getWriter

public PrintWriter getWriter throws IOException;

这个方法返回一个PringWriter对象用来记录格式化的响应实体。如果要反映使用的字符编码，必须修改响应的MIME类型。在调用这个方法之前，必须设定响应的content类型。

如果没有提供这样的编码类型，会抛出一个UnsupportedEncodingException，如果这个响应对象已调用getOutputStream，会抛出一个getOutputStream。

4、setContentLength

public void setContentLength(int length);

设置响应的内容的长度，这个方法会覆盖以前对内容长度的设定。为了保证成功地设定响应头的内容长度，在响应被提交到输出流之前必须调用这个方法。

5、setContentType

public void setContentType(String type);

这个方法用来设定响应的content类型。这个类型以后可能会在另外的一些情况下被隐式地修改，这里所说的另外的情况可能当服务器发现有必要的情况下对MIME的字符设置。

为了保证成功地设定响应头的content类型，在响应被提交到输出流之前必须调用这个方法。

七、SingleThreadModel接口

定义

public interface SingleThreadModel;

这是一个空接口，它指定了系统如何处理对同一个Servlet的调用。如果一个Servlet被这个接口指定，那么在这个Servlet中的service方法中将不会有两个线程被同时执行。

Servlet可以通过维持一个各自独立的Servlet实例池，或者通过只让Servlet的service中只有一个线程的方法来实现这个保证。

八、GenericServlet类

public abstract class GenericServlet implements Servlet,

ServletConfig, Serializable;

这个类的存在使得编写Servlet更加方便。它提供了一个简单的方案，这个方案用来执行有关Servlet生命周期的方法以及在初始化时对ServletConfig对象和ServletContext对象进行说明。

方法

1、destroy

public void destroy();

在这里destroy方法不做任何其他的工作。

2、getInitParameter

public String getInitParameter(String name);

这是一个简便的途径，它将会调用ServletConfig对象的同名的方法。

3、getInitParameterNames

public Enumeration getInitParameterNames();

这是一个简便的途径，它将会调用ServletConfig对象的同名的方法。

4、getServletConfig

public ServletConfig getServletConfig();

返回一个通过这个类的init方法产生的ServletConfig对象的说明。

5、getServletContext

public ServletContext getServletContext();

这是一个简便的途径，它将会调用ServletConfig对象的同名的方法。

6、getServletInfo

public String getServletInfo();

返回一个反映Servlet版本的String。

7、init

public void init() throws ServletException;

public void init(ServletConfig config) throws ServletException;

init(ServletConfig config)方法是一个对这个Servlet的生命周期进行初始化的简便的途径。

init()方法是用来让你对GenericServlet类进行扩充的，使用这个方法时，你不需要存储config对象，也不需要调用super.init(config)。

init(ServletConfig config)方法会存储config对象然后调用init()。如果你重载了这个方法，你必须调用super.init(config)，这样GenericServlet类的其他方法才能正常工作。

8、log

public void log(String msg);

public void log(String msg, Throwable cause);

通过Servlet content对象将Servlet的类名和给定的信息写入log文件中。

9、service

public abstract void service(ServletRequest request, ServletResponse

response) throws ServletException, IOException;

这是一个抽象的方法，当你扩展这个类时，为了执行网络请求，你必须执行它。

九、ServletInputStream类

定义

public abstract class ServletInputStream extends InputStream

这个类定义了一个用来读取客户端的请求信息的输入流。这是一个Servlet引擎提供的抽象类。一个Servlet通过使用ServletRequest接口获得了对一个ServletInputStream对象的说明。

这个类的子类必须提供一个从InputStream接口读取有关信息的方法。

方法

1、readLine

public int readLine(byte[] b, int off, int len) throws IOException;

从输入流的指定的偏移量开始将指定长度的字节读入到指定的数组中。如果该行所有请求的内容都已被读取，这个读取的过程将结束。如果是遇到了新的一行，新的一行的首个字符也将被读入到数组中。

十、ServletOutputStream类

定义

public abstract class ServletOutputStream extends OutputStream

这是一个由Servlet引擎使用的抽象类。Servlet通过使用ServletResponse接口的使用获得了对一个这种类型的对象的说明。利用这个输出流可以将数据返回到客户端。

这个类的子类必须提供一个向OutputStream接口写入有关信息的方法。

在这个接口中，当一个刷新或关闭的方法被调用时。所有数据缓冲区的信息将会被发送到客户端，也就是说响应被提交了。请注意，关闭这种类型的对象时不一定要关闭隐含的socket流。

方法

1、print

public void print(String s) throws IOException;

public void print(boolean b) throws IOException;

public void print(char c) throws IOException;

public void print(int i) throws IOException;

public void print(long l) throws IOException;

public void print(float f) throws IOException;

public void print(double d) throws IOException;

输出变量到输出流中

2、println

public void println() throws IOException;

public void println(String s) throws IOException;

public void println(boolean b) throws IOException;

public void println(char c) throws IOException;

public void println(int i) throws IOException;

public void println(long l) throws IOException;

public void println(float f) throws IOException;

public void println(double d) throws IOException;

输出变量到输出流中，并增加一个回车换行符

十一、ServletException类

定义

public class ServletException extends Exception

当Servlet遇到问题时抛出的一个异常。

构造函数

public ServletException();

public ServletException(String message);

public ServletException(String message, Throwable cause);

public ServletException(Throwable cause);

构造一个新的ServletException，如果这个构造函数包括一个Throwable参数，这个Throwable对象将被作为可能抛出这个异常的原因。

方法

1、getRootCause

public Throwable getRootCause();

如果配置了抛出这个异常的原因，这个方法将返回这个原因，否则返回一个空值。

十二、UnavailableException类

定义

public class UnavailableException extends ServletException

不论一个Servlet是永久地还是临时地无效，都会抛出这个异常。Servlet会记录这个异常以及Servlet引擎所要采取的相应措施。

临时的无效是指Servlet在某一时间由于一个临时的问题而不能处理请求。例如，在另一个不同的应用层的服务（可能是数据库）无法使用。这个问题可能会自行纠正或者需要采取其他的纠正措施。

永久的无效是指除非管理员采取措施，这个Servlet将不能处理客户端的请求。例如，这个Servlet配置信息丢失或Servlet的状态被破坏。

Servlet引擎可以安全地处理包括永久无效在内的这两种异常，但是对临时无效的正常处理可以使得Servlet引擎更健壮。特别的，这时对Servlet的请求只是被阻止（或者是被延期）一段时间，这显然要比在service自己重新启动前完全拒绝请求更为科学。

构造函数

public UnavailableException(Servlet servlet, String message);

public UnavailableException(int seconds, Servlet servlet,

String message);

构造一个包含指定的描述信息的新的异常。如果这个构造函数有一个关于秒数的参数，这将给出Servlet发生临时无效后，能够重新处理请求的估计时间。如果不包含这个参数，这意味着这个Servlet永久无效。

方法

1、getServlet

public Servlet getServlet();

返回报告无效的Servlet。这被Servlet引擎用来识别受到影响的Servlet。

2、getUnavailableSeconds

public int getUnavailableSeconds();

返回Servlet预期的无效时间，如果这个Servlet是永久无效，返回-1。

3、isPermanent

public boolean isPermanent();

如果这个Servlet永久无效，返回布尔值true，指示必须采取一些管理行动以使得这个Servlet可用。