一、HttpServletRequest接口

定义\

public interface HttpServletRequest extends ServletRequest;

用来处理一个对Servlet的HTTP格式的请求信息。

方法

1、getAuthType

public String getAuthType();

返回这个请求的身份验证模式。

2、getCookies

public Cookie[] getCookies();

返回一个数组，该数组包含这个请求中当前的所有cookie。如果这个请求中没有cookie，返回一个空数组。

3、getDateHeader

public long getDateHeader(String name);

返回指定的请求头域的值，这个值被转换成一个反映自1970-1-1日（GMT）以来的精确到毫秒的长整数。

如果头域不能转换，抛出一个IllegalArgumentException。如果这个请求头域不存在，这个方法返回-1。

4、getHeader

public String getHeader(String name);

返回一个请求头域的值。（译者注：与上一个方法不同的是，该方法返回一个字符串）

如果这个请求头域不存在，这个方法返回-1。

5、getHeaderNames

public Enumeration getHeaderNames();

该方法返回一个String对象的列表，该列表反映请求的所有头域名。

有的引擎可能不允许通过这种方法访问头域，在这种情况下，这个方法返回一个空的列表。

6、getIntHeader

public int getIntHeader(String name);

返回指定的请求头域的值，这个值被转换成一个整数。

如果头域不能转换，抛出一个IllegalArgumentException。如果这个请求头域不存在，这个方法返回-1。

7、getMethod

public String getMethod();

返回这个请求使用的HTTP方法（例如：GET、POST、PUT）

8、getPathInfo

public String getPathInfo();

这个方法返回在这个请求的URL的Servlet路径之后的请求URL的额外的路径信息。如果这个请求URL包括一个查询字符串，在返回值内将不包括这个查询字符串。这个路径在返回之前必须经过URL解码。如果在这个请求的URL的Servlet路径之后没有路径信息。这个方法返回空值。

9、getPathTranslated

public String getPathTranslated();

这个方法获得这个请求的URL的Servlet路径之后的额外的路径信息，并将它转换成一个真实的路径。在进行转换前，这个请求的URL必须经过URL解码。如果在这个URL的Servlet路径之后没有附加路径信息。这个方法返回空值。

10、getQueryString

public String getQueryString();

返回这个请求URL所包含的查询字符串。一个查询字串符在一个URL中由一个“？”引出。如果没有查询字符串，这个方法返回空值。

11、getRemoteUser

public String getRemoteUser

返回作了请求的用户名，这个信息用来作HTTP用户论证。

如果在请求中没有用户名信息，这个方法返回空值。

12、getRequestedSessionId

public String getRequestedSessionId();

返回这个请求相应的session id。如果由于某种原因客户端提供的session id是无效的，这个session id将与在当前session中的session id不同，与此同时，将建立一个新的session。

如果这个请求没与一个session关联，这个方法返回空值。

13、getRequestURI

public String getRequestURI();

从HTTP请求的第一行返回请求的URL中定义被请求的资源的部分。如果有一个查询字符串存在，这个查询字符串将不包括在返回值当中。例如，一个请求通过 /catalog/books?id=1这样的URL路径访问，这个方法将返回/catalog/books。这个方法的返回值包括了Servlet路径和路径信息。

如果这个URL路径中的的一部分经过了URL编码，这个方法的返回值在返回之前必须经过解码。

14、getServletPath

public String getServletPath();

这个方法返回请求URL反映调用Servlet的部分。例如，一个Servlet被映射到/catalog/summer这个URL路径，而一个请求使用了/catalog/summer/casual这样的路径。所谓的反映调用Servlet的部分就是指/catalog/summer。

如果这个Servlet不是通过路径匹配来调用。这个方法将返回一个空值。

15、getSession

public HttpSession getSession();

public HttpSession getSession(boolean create);

返回与这个请求关联的当前的有效的session。如果调用这个方法时没带参数，那么在没有session与这个请求关联的情况下，将会新建一个session。如果调用这个方法时带入了一个布尔型的参数，只有当这个参数为真时，session才会被建立。

为了确保session能够被完全维持。Servlet开发者必须在响应被提交之前调用该方法。

如果带入的参数为假，而且没有session与这个请求关联。这个方法会返回空值。

16、isRequestedSessionIdValid

public boolean isRequestedSessionIdValid();

这个方法检查与此请求关联的session当前是不是有效。如果当前请求中使用的session无效，它将不能通过getSession方法返回。

17、isRequestedSessionIdFromCookie

public boolean isRequestedSessionIdFromCookie();

如果这个请求的session id是通过客户端的一个cookie提供的，该方法返回真，否则返回假。

18、isRequestedSessionIdFromURL

public boolean isRequestedSessionIdFromURL();

如果这个请求的session id是通过客户端的URL的一部分提供的，该方法返回真，否则返回假。请注意此方法与isRequestedSessionIdFromUrl在URL的拼写上不同。

以下方法将被取消\

19、isRequestedSessionIdFromUrl

public boolean isRequestedSessionIdFromUrl();

该方法被isRequestedSessionIdFromURL代替。

二、HttpServletResponse接口

定义\

public interface HttpServletResponse extends ServletResponse

描述一个返回到客户端的HTTP回应。这个接口允许Servlet程序员利用HTTP协议规定的头信息。

成员变量

public static final int SC\_CONTINUE = 100;

public static final int SC\_SWITCHING\_PROTOCOLS = 101;

public static final int SC\_OK = 200;

public static final int SC\_CREATED = 201;

public static final int SC\_ACCEPTED = 202;

public static final int SC\_NON\_AUTHORITATIVE\_INFORMATION = 203;

public static final int SC\_NO\_CONTENT = 204;

public static final int SC\_RESET\_CONTENT = 205;

public static final int SC\_PARTIAL\_CONTENT = 206;

public static final int SC\_MULTIPLE\_CHOICES = 300;

public static final int SC\_MOVED\_PERMANENTLY = 301;

public static final int SC\_MOVED\_TEMPORARILY = 302;

public static final int SC\_SEE\_OTHER = 303;

public static final int SC\_NOT\_MODIFIED = 304;

public static final int SC\_USE\_PROXY = 305;

public static final int SC\_BAD\_REQUEST = 400;

public static final int SC\_UNAUTHORIZED = 401;

public static final int SC\_PAYMENT\_REQUIRED = 402;

public static final int SC\_FORBIDDEN = 403;

public static final int SC\_NOT\_FOUND = 404;

public static final int SC\_METHOD\_NOT\_ALLOWED = 405;

public static final int SC\_NOT\_ACCEPTABLE = 406;

public static final int SC\_PROXY\_AUTHENTICATION\_REQUIRED = 407;

public static final int SC\_REQUEST\_TIMEOUT = 408;

public static final int SC\_CONFLICT = 409;

public static final int SC\_GONE = 410;

public static final int SC\_LENGTH\_REQUIRED = 411;

public static final int SC\_PRECONDITION\_FAILED = 412;

public static final int SC\_REQUEST\_ENTITY\_TOO\_LARGE = 413;

public static final int SC\_REQUEST\_URI\_TOO\_LONG = 414;

public static final int SC\_UNSUPPORTED\_MEDIA\_TYPE = 415;

public static final int SC\_INTERNAL\_SERVER\_ERROR = 500;

public static final int SC\_NOT\_IMPLEMENTED = 501;

public static final int SC\_BAD\_GATEWAY = 502;

public static final int SC\_SERVICE\_UNAVAILABLE = 503;

public static final int SC\_GATEWAY\_TIMEOUT = 504;

public static final int SC\_HTTP\_VERSION\_NOT\_SUPPORTED = 505;

以上HTTP产状态码是由HTTP/1.1定义的。

方法

1、addCookie

public void addCookie(Cookie cookie);

在响应中增加一个指定的cookie。可多次调用该方法以定义多个cookie。为了设置适当的头域，该方法应该在响应被提交之前调用。

2、containsHeader

public boolean containsHeader(String name);

检查是否设置了指定的响应头。

3、encodeRedirectURL

public String encodeRedirectURL(String url);

对sendRedirect方法使用的指定URL进行编码。如果不需要编码，就直接返回这个URL。之所以提供这个附加的编码方法，是因为在 redirect的情况下，决定是否对URL进行编码的规则和一般情况有所不同。所给的URL必须是一个绝对URL。相对URL不能被接收，会抛出一个 IllegalArgumentException。

所有提供给sendRedirect方法的URL都应通过这个方法运行，这样才能确保会话跟踪能够在所有浏览器中正常运行。

4、encodeURL

public String encodeURL(String url);

对包含session ID的URL进行编码。如果不需要编码，就直接返回这个URL。Servlet引擎必须提供URL编码方法，因为在有些情况下，我们将不得不重写URL，例如，在响应对应的请求中包含一个有效的session，但是这个session不能被非URL的（例如cookie）的手段来维持。

所有提供给Servlet的URL都应通过这个方法运行，这样才能确保会话跟踪能够在所有浏览器中正常运行。

5、sendError

public void sendError(int statusCode) throws IOException;

public void sendError(int statusCode, String message) throws

IOException;

用给定的状态码发给客户端一个错误响应。如果提供了一个message参数，这将作为响应体的一部分被发出，否则，服务器会返回错误代码所对应的标准信息。

调用这个方法后，响应立即被提交。在调用这个方法后，Servlet不会再有更多的输出。

6、sendRedirect

public void sendRedirect(String location) throws IOException;

使用给定的路径，给客户端发出一个临时转向的响应（SC\_MOVED\_TEMPORARILY）。给定的路径必须是绝对URL。相对URL将不能被接收，会抛出一个IllegalArgumentException。

这个方法必须在响应被提交之前调用。调用这个方法后，响应立即被提交。在调用这个方法后，Servlet不会再有更多的输出。

7、setDateHeader

public void setDateHeader(String name, long date);

用一个给定的名称和日期值设置响应头，这里的日期值应该是反映自1970-1-1日（GMT）以来的精确到毫秒的长整数。如果响应头已经被设置，新的值将覆盖当前的值。

8、setHeader

public void setHeader(String name, String value);

用一个给定的名称和域设置响应头。如果响应头已经被设置，新的值将覆盖当前的值。

9、setIntHeader

public void setIntHeader(String name, int value);

用一个给定的名称和整形值设置响应头。如果响应头已经被设置，新的值将覆盖当前的值。

10、setStatus

public void setStatus(int statusCode);

这个方法设置了响应的状态码，如果状态码已经被设置，新的值将覆盖当前的值。

以下的几个方法将被取消\

11、encodeRedirectUrl

public String encodeRedirectUrl(String url);

该方法被encodeRedirectURL取代。

12、encodeUrl

public String encodeUrl(String url);

该方法被encodeURL取代。

13、setStatus

public void setStatus(int statusCode, String message);

这个方法设置了响应的状态码，如果状态码已经被设置，新的值将覆盖当前的值。如果提供了一个message，它也将会被作为响应体的一部分被发送。

三、HttpSession接口

定义\

public interface HttpSession

这个接口被Servlet引擎用来实现在HTTP客户端和HTTP会话两者的关联。这种关联可能在多外连接和请求中持续一段给定的时间。session用来在无状态的HTTP协议下越过多个请求页面来维持状态和识别用户。

一个session可以通过cookie或重写URL来维持。

方法

1、getCreationTime

public long getCreationTime();

返回建立session的时间，这个时间表示为自1970-1-1日（GMT）以来的毫秒数。

2、getId

public String getId();

返回分配给这个session的标识符。一个HTTP session的标识符是一个由服务器来建立和维持的唯一的字符串。

3、getLastAccessedTime

public long getLastAccessedTime();

返回客户端最后一次发出与这个session有关的请求的时间，如果这个session是新建立的，返回-1。这个时间表示为自1970-1-1日（GMT）以来的毫秒数。

4、getMaxInactiveInterval

public int getMaxInactiveInterval();

返加一个秒数，这个秒数表示客户端在不发出请求时，session被Servlet引擎维持的最长时间。在这个时间之后，Servlet引擎可能被Servlet引擎终止。如果这个session不会被终止，这个方法返回-1。

当session无效后再调用这个方法会抛出一个IllegalStateException。

5、getValue

public Object getValue(String name);

返回一个以给定的名字绑定到session上的对象。如果不存在这样的绑定，返回空值。

当session无效后再调用这个方法会抛出一个IllegalStateException。

6、getValueNames

public String[] getValueNames();

以一个数组返回绑定到session上的所有数据的名称。

当session无效后再调用这个方法会抛出一个IllegalStateException。

7、invalidate

public void invalidate();

这个方法会终止这个session。所有绑定在这个session上的数据都会被清除。并通过HttpSessionBindingListener接口的valueUnbound方法发出通告。

8、isNew

public boolean isNew();

返回一个布尔值以判断这个session是不是新的。如果一个session已经被服务器建立但是还没有收到相应的客户端的请求，这个session将被认为是新的。这意味着，这个客户端还没有加入会话或没有被会话公认。在他发出下一个请求时还不能返回适当的session认证信息。

当session无效后再调用这个方法会抛出一个IllegalStateException。

9、putValue

public void putValue(String name, Object value);

以给定的名字，绑定给定的对象到session中。已存在的同名的绑定会被重置。这时会调用HttpSessionBindingListener接口的valueBound方法。

当session无效后再调用这个方法会抛出一个IllegalStateException。

10、removeValue

public void removeValue(String name);

取消给定名字的对象在session上的绑定。如果未找到给定名字的绑定的对象，这个方法什么出不做。 这时会调用HttpSessionBindingListener接口的valueUnbound方法。

当session无效后再调用这个方法会抛出一个IllegalStateException。

11、setMaxInactiveInterval

public int setMaxInactiveInterval(int interval);

设置一个秒数，这个秒数表示客户端在不发出请求时，session被Servlet引擎维持的最长时间。

以下这个方法将被取消\

12、getSessionContext

public HttpSessionContext getSessionContext();

返回session在其中得以保持的环境变量。这个方法和其他所有HttpSessionContext的方法一样被取消了。

四、HttpSessionBindingListener接口

定义\

public interface HttpSessionBindingListener

这个对象被加入到HTTP的session中，执行这个接口会通告有没有什么对象被绑定到这个HTTP session中或被从这个HTTP session中取消绑定。

方法

1、valueBound

public void valueBound(HttpSessionBindingEvent event);

当一个对象被绑定到session中，调用此方法。HttpSession.putValue方法被调用时，Servlet引擎应该调用此方法。

2、valueUnbound

public void valueUnbound(HttpSessionBindingEvent event);

当一个对象被从session中取消绑定，调用此方法。HttpSession.removeValue方法被调用时，Servlet引擎应该调用此方法。

五、HttpSessionContext接口

定义\

此接口将被取消\

public interface HttpSessionContext

这个对象是与一组HTTP session关联的单一的实体。

这个接口由于安全的原因被取消，它出现在目前的版本中仅仅是为了兼容性的原因。这个接口的方法将模拟以前的版本的定义返回相应的值。

方法

1、getSession

public HttpSession getSession(String sessionId);

当初用来返回与这个session id相关的session。现在返回空值。

2、getIds

public Enumeration getIds();

当初用来返回这个环境下所有session id的列表。现在返回空的列表。

六、Cookie类\

定义\

public class Cookie implements Cloneable

这个类描述了一个cookie，有关cookie的定义你可以参照Netscape Communications Corporation的说明，也可以参照RFC 2109。

构造函数

public Cookie(String name, String value);

用一个name-value对定义一个cookie。这个name必须能被HTTP/1.1所接受。

以字符$开头的name被RFC 2109保留。

给定的name如果不能被HTTP/1.1所接受，该方法抛出一个IllegalArgumentException。

方法

1、getComment

public String getComment();

返回描述这个cookie目的的说明，如果未定义这个说明，返回空值。

2、getDomain

public String getDomain();

返回这个cookie可以出现的区域，如果未定义区域，返回空值。

3、getMaxAge

public int getMaxAge();

这个方法返回这个cookie指定的最长存活时期。如果未定义这个最长存活时期，该方法返回-1。

4、getName

public String getName();

该方法返回cookie名。

5、getPath

public String getPath();

返回这个cookie有效的所有URL路径的前缀，如果未定义，返回空值。

6、getSecure

public boolean getSecure();

如果这个cookie只通过安全通道传输返回真，否则返回假。

7、getValue

public String getValue();

该方法返回cookie的值。

8、getVersion

public int getVersion();

返回cookie的版本。版本1由RFC 2109解释。版本0由Netscape Communications Corporation的说明解释。新构造的cookie默认使用版本0。

9、setComment

public void setComment(String purpose);

如果一个用户将这个cookie提交给另一个用户，必须通过这个说明描述这个cookie的目的。版本0不支持这个属性。

10、setDomain

public void setDomain(String pattern);

这个方法设置cookie的有效域的属性。这个属性指定了cookie可以出现的区域。一个有效域以一个点开头（.foo.com），这意味着在指定的域名解析系统的区域中（可能是www.foo.com但不是a.b.foo.com）的主机可以看到这个cookie。默认情况是，cookie只能返回保存它的主机。

11、setMaxAge

public void setMaxAge(int expiry);

这个方法设定这个cookie的最长存活时期。在该存活时期之后，cookie会被终目。负数表示这个cookie不会生效，0将从客户端删除这个cookie。

12、setPath

public void setPath(String uri);

这个方法设置cookie的路径属性。客户端只能向以这个给定的路径String开头的路径返回cookie。

13、setSecure

public void setSecure(boolean flag);

指出这个cookie只能通过安全通道（例如HTTPS）发送。只有当产生这个cookie的服务器使用安全协议发送这个cookie值时才能这样设置。

14、setValue

public void setValue(String newValue);

设置这个cookie的值，对于二进制数据采用BASE64编码。

版本0不能使用空格、{}、()、=、，、“”、/、?、@、：以及；。

15、setVersion

public void setVersion(int v);

设置cookie的版本号

七、HttpServlet类\

定义\

public class HttpServlet extends GenericServlet implements

Serializable

这是一个抽象类，用来简化HTTP Servlet写作的过程。它是GenericServlet类的扩充，提供了一个处理HTTP协议的框架。

在这个类中的service方法支持例如GET、POST这样的标准的HTTP方法。这一支持过程是通过分配他们到适当的方法（例如doGet、doPost）来实现的。

方法

1、doDelete

protected void doDelete(HttpServletRequest request,

HttpServletResponse response) throws ServletException,

IOException;

被这个类的service方法调用，用来处理一个HTTP DELETE操作。这个操作允许客户端请求从服务器上删除URL。这一操作可能有负面影响，对此用户就负起责任。

这一方法的默认执行结果是返回一个HTTP BAD\_REQUEST错误。当你要处理DELETE请求时，你必须重载这一方法。

2、doGet

protected void doGet(HttpServletRequest request,

HttpServletResponse response) throws ServletException,

IOException;

被这个类的service方法调用，用来处理一个HTTP GET操作。这个操作允许客户端简单地从一个HTTP服务器“获得”资源。对这个方法的重载将自动地支持HEAD方法。

GET操作应该是安全而且没有负面影响的。这个操作也应该可以安全地重复。

这一方法的默认执行结果是返回一个HTTP BAD\_REQUEST错误。

3、doHead

protected void doHead(HttpServletRequest request,

HttpServletResponse response) throws ServletException,

IOException;

被这个类的service方法调用，用来处理一个HTTP HEAD操作。默认的情况是，这个操作会按照一个无条件的GET方法来执行，该操作不向客户端返回任何数据，而仅仅是返回包含内容长度的头信息。

与GET操作一样，这个操作应该是安全而且没有负面影响的。这个操作也应该可以安全地重复。

这个方法的默认执行结果是自动处理HTTP HEAD操作，这个方法不需要被一个子类执行。

4、doOptions

protected void doOptions(HttpServletRequest request,

HttpServletResponse response) throws ServletException,

IOException;

被这个类的service方法调用，用来处理一个HTTP OPTION操作。这个操作自动地决定支持哪一种HTTP方法。例如，一个Servlet写了一个HttpServlet的子类并重载了doGet方法，doOption会返回下面的头：

Allow: GET,HEAD,TRACE,OPTIONS

你一般不需要重载这个方法。

5、doPost

protected void doPost(HttpServletRequest request,

HttpServletResponse response) throws ServletException,

IOException;

被这个类的service方法调用，用来处理一个HTTP POST操作。这个操作包含请求体的数据，Servlet应该按照他行事。

这个操作可能有负面影响。例如更新存储的数据或在线购物。

这一方法的默认执行结果是返回一个HTTP BAD\_REQUEST错误。当你要处理POST操作时，你必须在HttpServlet的子类中重载这一方法。

6、doPut

protected void doPut(HttpServletRequest request,

HttpServletResponse response) throws ServletException,

IOException;

被这个类的service方法调用，用来处理一个HTTP PUT操作。这个操作类似于通过FTP发送文件。

这个操作可能有负面影响。例如更新存储的数据或在线购物。

这一方法的默认执行结果是返回一个HTTP BAD\_REQUEST错误。当你要处理PUT操作时，你必须在HttpServlet的子类中重载这一方法。

7、doTrace

protected void doTrace(HttpServletRequest request,

HttpServletResponse response) throws ServletException,

IOException;

被这个类的service方法调用，用来处理一个HTTP TRACE操作。这个操作的默认执行结果是产生一个响应，这个响应包含一个反映trace请求中发送的所有头域的信息。

当你开发Servlet时，在多数情况下你需要重载这个方法。

8、getLastModified

protected long getLastModified(HttpServletRequest request);

返回这个请求实体的最后修改时间。为了支持GET操作，你必须重载这一方法，以精确地反映最后修改的时间。这将有助于浏览器和代理服务器减少装载服务器和网络资源，从而更加有效地工作。返回的数值是自1970-1-1日（GMT）以来的毫秒数。

默认的执行结果是返回一个负数，这标志着最后修改时间未知，它也不能被一个有条件的GET操作使用。

9、service

protected void service(HttpServletRequest request,

HttpServletResponse response) throws ServletException,

IOException;

public void service(ServletRequest request, ServletResponse response)

throws ServletException, IOException;

这是一个Servlet的HTTP-specific方案，它分配请求到这个类的支持这个请求的其他方法。

当你开发Servlet时，在多数情况下你不必重载这个方法。

八、HttpSessionBindingEvent类\

定义\

public class HttpSessionBindingEvent extends EventObject

这个事件是在监听到HttpSession发生绑定和取消绑定的情况时连通HttpSessionBindingListener的。这可能是一个session被终止或被认定无效的结果。

事件源是HttpSession.putValue或HttpSession.removeValue。

构造函数

public HttpSessionBindingEvent(HttpSession session, String name);

通过引起这个事件的Session和发生绑定或取消绑定的对象名构造一个新的HttpSessionBindingEvent。

方法

1、getName

public String getName();

返回发生绑定和取消绑定的对象的名字。

2、getSession

public HttpSession getSession();

返回发生绑定和取消绑定的session的名字。

九、HttpUtils类\

定义\

public class HttpUtils

收集HTTP Servlet使用的静态的有效的方法。

方法

1、getRequestURL

public static StringBuffer getRequestURL(HttpServletRequest

request);

在服务器上重建客户端用来建立请求的URL。这个方法反映了不同的协议（例如http和https）和端口，但不包含查询字符串。

这个方法返回一个StringBuffer而不是一个String，这样URL可以被Servlet开发者有效地修改。

2、parsePostData

public static Hashtable parsePostData(int len,

ServletInputstream in);

解析一个包含MIME类型application/x-www-form-urlencoded的数据的流，并创建一个具有关键值-数据对的 hash table。这里的关键值是字符串，数据是该字符串所对应的值的列表。一个关键值可以在POST的数据中出现一次或多次。这个关键值每出现一次，它的相应的值就被加入到hash table中的字符串所对应的值的列表中。

从POST数据读出的数据将经过URL解码，+将被转换为空格以十六进制传送的数据（例如%xx）将被转换成字符。

当POST数据无效时，该方法抛出一个IllegalArgumentException。

3、parseQueryString

public static Hashtable parseQueryString(String s);

解析一个查询字符串，并创建一个具有关键值-数据对的hash table。这里的数据是该字符串所对应的值的列表。一个关键值可以出现一次或多次。这个关键值每出现一次，它的相应的值就被加入到hash table中的字符串所对应的值的列表中。

从查询字符串读出的数据将经过URL解码，+将被转换为空格以十六进制传送的数据（例如%xx）将被转换成字符。

当查询字符串无效时，该方法抛出一个IllegalArgumentException。

术语表

bytecode

字节码：由Java编译器和Java解释程序生成的机器代码。

cookie

由Web服务器建立的数据，该数据存储在用户的计算机上，提供了一个Web站点跟踪用户的参数并存储在用户自己硬盘上的方法。

HTTP

超文本传输协议。一个请求响应协议用来连接WWW服务器向客户端浏览器传输HTML页面。

输入流对象

一个对象，由ServletInputStream类定义，被Servlet用来从客户端读取请求。

映射

由Servlet实例和Servlet返回数据的URL组成的一对，例如，HelloServlet和/hello/index.html。

输出流对象

一个对象，由ServletOutputStream class类定义，被Servlet用来向客户端返回数据。

request dispatcher object

由RequestDispatcher接口定义的一个对象，用来从客户端接收请求，并将其发送到Web服务器上可用的其他资源（例如Servlet、CGI、HTML文件或JSP文件）。

sandboxed servlet

在一个安全性约束下运行的Servlet。

servlet

一个小的，具有平台无关性的，没有图形用户界面的Java程序。它可以在许多方面扩充Web服务的功能。

servlet configuration object

ServletConfig接口定义的一个对象，用来配置一个Servlet。

servlet context object

ServletContext接口定义的一个对象。给予Servlet有关Servlet引擎的信息。

servlet引擎

由Web服务器提供商制作的一个环境，可以允许Servlet在具体的Web服务器上运行。

servlet请求对象

由ServletRequest接口定义的一个对象，允许Servlet获得用关客户端请求的数据。

servlet response object

由ServletResponse接口定义的一个对象，允许Servlet作出响应。

servlet runner

Java Servlet Developer’s Kit (JSDK)中的sun.servlet.http.HttpServer过程，它使得Servlet得以运行。

会话跟踪

在一个Web应用程序中，识别一个从同一个客户端发出的连续的唯一的请求的能力。

SSL

加密套接字协议层。一个安全协议，用来在Iternet上的客户端浏览器和服务器交换密钥和加密数据。

URI

统一资源标识。定义一个Internet地址，它是一个URL的超集。

URL

统一资源路径。这个地址定义了到达一个WWW上的文件的路线，通常由协议前缀、域名、目录名和文件名组成。