Java servlet

From Wikipedia, the free encyclopedia

是一个扩展了服务器功能的Java程序

A **Java servlet** is a Java program that extends the capabilities of a server. Although servlets can respond to any types of requests, they most commonly implement applications hosted on Web servers.[1] Such Web servlets are the Java counterpart to other dynamic Web content technologies such as PHP and ASP.NET.

Contents [hide]

使用servlet给web服务器添加动态内容

简介

Introduction [edit]

处理或存储符合Java Servlet API的Java类

HTTP servlet：servlet通常使用HTTP协议

Servlets are most often used to process or store a Java class in Java EE that conforms to the Java Servlet API,[2] a standard for implementing Java classes which respond to requests. Servlets could in principle communicate over any client–server protocol, but they are most often used with the HTTP protocol. Thus "servlet" is often used as shorthand for "HTTP servlet".[3] Thus, a software developer may use a servlet to add dynamic content to a web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlets can maintain state in session variables across many server transactions by using HTTP cookies, or rewriting URLs.

使用cookies来维护跨多个服务器事务的会话状态变量

To deploy and run a servlet, a web container must be used. A web container (also known as a servlet container) is essentially the component of a web server that interacts with the servlets. The web container is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet and ensuring that the URL requester has the correct access rights.

Web容器(也称servlet容器)本质上是web服务器的组件，与servlets交互
职责：管理servlets的生命周期、将URL映射到特定的servlet、确保URL请求具有正确的访问权限

The Servlet API, contained in the Java package hierarchy `javax.servlet`, defines the expected interactions of the web container and a servlet.[3]

Servlet API定义了web容器和servlet的期望交互
Servlet是一个接受请求并生成响应内容的对象

A `Servlet` is an object that receives a request and generates a response based on that request. The basic Servlet package defines Java objects to represent servlet requests and responses, as well as objects to reflect the servlet's configuration parameters and execution environment. The package `javax.servlet.http` defines HTTP-specific subclasses of the generic servlet elements, including session management objects that track multiple requests and responses between the web server and a client. Servlets may be packaged in a WAR file as a web application.

Servlets can be generated automatically from Java Server Pages (JSP) by the JavaServer Pages

Servlet与JSP的关系：Servlets可以从JSP自动生成（JSP编译器）

Servlet与JSP的区别：servlets将HTML嵌入到Java代码中，JSPs将Java代码嵌入到HTML中

compiler. The difference between servlets and JSP is that servlets typically embed HTML inside Java code, while JSPs embed Java code in HTML. While the direct usage of servlets to generate HTML (as shown in the example below) has become rare, the higher level MVC web framework in Java EE (JSF) still explicitly uses the servlet technology for the low level request/response handling via the FacesServlet. A somewhat older usage is to use servlets in conjunction with JSPs in a pattern called "Model 2", which is a flavor of the model–view–controller.

The current version of Servlet is 3.1.

上层的MVC Web框架仍然使用servlet技术来处理底层的请求/响应

历史

History [edit]

The Servlet1 specification was created by Sun Microsystems, with version 1.0 finalized in June 1997. Starting with version 2.3, the specification was developed under the Java Community Process. JSR 53 defined both the Servlet 2.3 and JavaServer Page 1.2 specifications. JSR 154 specifies the Servlet 2.4 and 2.5 specifications. As of June 9, 2015, the current version of the Servlet specification is 3.1.

In his blog on java.net, Sun veteran and GlassFish lead Jim Driscoll details the history of servlet technology.[4] James Gosling first thought of servlets in the early days of Java, but the concept did not become a product until Sun shipped the *Java Web Server* [*clarify*] product. This was before what is now the Java Platform, Enterprise Edition was made into a specification.

Servlet API history

Servlet API version	Released	Platform	Important Changes
Servlet 4.0	Under development	Java EE 8	HTTP/2
Servlet 3.1	May 2013	Java EE 7	Non-blocking I/O, HTTP protocol upgrade mechanism (WebSocket)[5] 非阻塞I/O、HTTP协议升级机制(WebSocket)
Servlet 3.0	December 2009	Java EE 6, Java SE 6	Pluggability, Ease of development, Async Servlet, Security, File Uploading 可插拔性、异步Servlet、安全、文件上传
Servlet 2.5	September 2005	Java EE 5, Java SE 5	Requires Java SE 5, supports annotation 支持注解
Servlet 2.4	November 2003	J2EE 1.4, J2SE 1.3	web.xml uses XML Schema web.xml 使用XML模式
Servlet 2.3	August 2001	J2EE 1.3, J2SE 1.2	Addition of `Filter` 增加 过滤器
Servlet 2.2	August 1999	J2EE 1.2, J2SE 1.2	Becomes part of J2EE, introduced independent web applications in .war files
Servlet 2.1	November 1998	Unspecified	First official specification, added `RequestDispatcher`, `ServletContext` 请求分发器、Servlet上下文
Servlet 2.0		JDK 1.1	Part of Java Servlet Development Kit 2.0

Servlet 1.0	June 1997		

Compared with other web application models [edit]

This section **relies largely or entirely upon a single source**. Relevant discussion may be found on the talk page. Please help improve this article by introducing citations to additional sources.

(December 2013)

The advantages of using servlets are their fast performance and ease of use combined with more power over traditional CGI (Common Gateway Interface). Traditional CGI scripts written in Java have a number of performance disadvantages:

传统CGI脚本的性能劣势：

每个请求都会创建一个新的进程

- When an HTTP request is made, a new process is created each time the CGI script is called. The overhead associated with process creation can dominate the workload especially when the script does relatively fast operations. Thus, process creation will take more time for CGI script execution. In contrast, for servlets, each request is handled by a separate Java thread *within* the web server process, thereby avoiding the overhead associated with forking processes within the HTTP daemon.

servlets：每个请求都是由web服务器进程中的一个独立的Java线程处理

多副本：每个请求会复制CGI脚本到内存中

- Simultaneous CGI requests will load the CGI script to be copied into memory once per request. With servlets, there is only one copy that persists across requests and is shared between threads.

servlets：仅一个副本，跨多个请求存在，并在线程之间共享

- Only a single instance answers all requests concurrently. This reduces memory usage and eases the management of persistent data.

仅一个实例并发地回答所有的请求。这减少了内存使用，并简化了持久化数据的管理。

- A servlet can be run by a servlet container in a restrictive environment, called a sandbox. This is similar to an applet that runs in the sandbox of the web browser. This enables restricted use of potentially harmful servlets.[3] CGI programs can of course also sandbox themselves, since they are simply OS processes.

Technologies like FastCGI and its derivatives (including SCGI, WSGI) do not exhibit the performance disadvantages of CGI incurred by the constant process spawning. They are, however, roughly as simple as CGI. They are therefore also in contrast with servlets which are substantially more complex.

Servlet的生命周期

Life cycle of a servlet [edit]

3个方法对servlet的生命周期至关重要

Three methods are central to the life cycle of a servlet. These are `init()`, `service()`, and `destroy()`. They are implemented by every servlet and are invoked at specific times by the server.

- During initialization stage of the servlet life cycle, the web container initializes the servlet instance by calling the `init()` method, passing an object implementing the `javax.servlet.ServletConfig` interface. This configuration object allows the servlet to access name-value initialization parameters from the web application.

初始化阶段：web容器通过调用init()方法来初始化servlet实例

配置对象允许servlet访问web应用的键值对初始化参数

- After initialization, the servlet instance can service client requests. Each request is serviced in its own separate thread. The web container calls the `service()` method of the servlet for every request. The `service()` method determines the kind of request being made and dispatches it to an appropriate method to handle the request. The developer of the servlet must provide an

在初始化之后，servlet实例可以服务客户端请求。每个请求都在其单独的线程中提供服务。

web容器为每个请求调用servlet的service()方法，service()方法确定请求的类型并将其分派到一个适当的方法来处理请求。

implementation for these methods. If a request is made for a method that is not implemented by the servlet, the method of the parent class is called, typically resulting in an error being returned to the requester.

最后，web容器调用destroy()方法使servlet停止服务。

- Finally, the web container calls the `destroy()` method that takes the servlet out of service. The `destroy()` method, like `init()`, is called only once in the lifecycle of a servlet.

The following is a typical user scenario of these methods. 这些方法的典型用户场景

1. Assume that a user requests to visit a URL. 一个用户请求访问 URL
   - The browser then generates an HTTP request for this URL.
   - This request is then sent to the appropriate server. HTTP请求由web服务器接收到并提交到servlet容器
2. The HTTP request is received by the web server and forwarded to the servlet container.
   - The container maps this request to a particular servlet. 容器将此请求映射到一个特定的servlet
   - The servlet is dynamically retrieved and loaded into the address space of the container. servlet被动态地检索并加载到该容器的地址空间
3. The container invokes the `init()` method of the servlet. 容器调用servlet的init()方法（仅在首次被加载到内存时才会调用）
   - This method is invoked only when the servlet is first loaded into memory.
   - It is possible to pass initialization parameters to the servlet so that it may configure itself.
4. The container invokes the `service()` method of the servlet.
   - This method is called to process the HTTP request. 调用此方法来处理HTTP请求
   - The servlet may read data that has been provided in the HTTP request.
   - The servlet may also formulate an HTTP response for the client.
5. The servlet remains in the container's address space and is available to process any other HTTP requests received from clients.
   - The `service()` method is called for each HTTP request. 为每个HTTP请求调用service()方法
6. The container may, at some point, decide to unload the servlet from its memory.
   - The algorithms by which this decision is made are specific to each container.
7. The container calls the servlet's `destroy()` method to relinquish any resources such as file handles that are allocated for the servlet; important data may be saved to a persistent store. 容器调用servlet的destroy()方法来释放分配给它的任何资源
8. The memory allocated for the servlet and its objects can then be garbage collected. 分配给此servlet的内存及其对象可以被垃圾收集

示例

Example [edit]

The following example servlet prints how many times its `service()` method was called.

Note that `HttpServlet` is a subclass of `GenericServlet`, an implementation of the `Servlet` interface.

The `service()` method of `HttpServlet` class dispatches requests to the methods `doGet()`, `doPost()`, `doPut()`, `doDelete()`, and so on; according to the HTTP request. In the example below `service()` is overridden and does not distinguish which HTTP request method it serves.

```
import java.io.IOException;
```

```
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ServletLifeCycleExample extends HttpServlet {

    private int count;

    @Override
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        getServletContext().log("init() called");
        count = 0;
    }

    @Override
    protected void service(HttpServletRequest request,
HttpServletResponse response)
            throws ServletException, IOException {
        getServletContext().log("service() called");
        count++;
        response.getWriter().write("Incrementing the count: count = " +
count);
    }

    @Override
    public void destroy() {
        getServletContext().log("destroy() called");
    }

}
```

Container servers [edit]

The specification for Servlet technology has been implemented in many products. See a list of implementations on the Web container page.

引用列表

References [edit]

1. ^ "servlet" . http://www.webopedia.com/ : WEBOPEDIA. Retrieved 2011-04-27. "A small java program that runs on a server. The term usually refers to a Java applet which runs within a web server environment. This is analogous to a Java applet that runs within a web browser environment."
2. ^ Java Servlet API
3. ^ a b c [1] 1.1 What is a servlet?
4. ^ "Servlet History | Java.net" . Weblogs.java.net. 2005-12-10. Retrieved 2013-06-14.
5. ^ [2] What's new in Servlet 3.1? - Java EE 7 moving forward

外部链接

External links [edit]

- JSR 369 - Java servlet 4.0 documentation
- JSR 340 - Java servlet 3.1 documentation JSR 340: Java Servlet 3.1 Specification
- JSR 315 - Java servlet 3.0 documentation
- JSR 154 - Java servlet 2.4 documentation
- JSR 53 - Java servlet 2.3 documentation