## Intro

Servlet is java class/component used in request-response programming model.

*javax.servlet* and *javax.servlet.http****Servlet***interface defines life-cycle methods.

The lifecycle of servlet is controlled by a servlet-container. When a request is mapped to servlet, the container performs the following steps:

1. If an instance of servlet does not exist, the web container
   1. Loads the servlet class
   2. Creates an instance of the servlet classs
   3. Initializes the servlet instance by calling the ***init*** method(override this method to customize persistent conf data, resources…).
2. Invokes the ***service*** method
3. If the container needs to remove the servlet(reclaim memory or is shutdown), it finalizes the servlet by calling the ***destroy*** method.

#### Handling Servlet Life-Cycle Events

|  |  |  |
| --- | --- | --- |
| **bject** | **Event** | **Listener Interface and Event Class** |
| Web context (see [Accessing the Web Context](http://docs.oracle.com/javaee/5/tutorial/doc/bnagl.html)) | Initialization and destruction | javax.servlet.ServletContextListener and  ServletContextEvent |
| Attribute added, removed, or replaced | javax.servlet.ServletContextAttributeListener and  ServletContextAttributeEvent |
| Session (See [Maintaining Client State](http://docs.oracle.com/javaee/5/tutorial/doc/bnagm.html)) | Creation, invalidation, activation, passivation, and timeout | javax.servlet.http.HttpSessionListener, javax.servlet.http.HttpSessionActivationListener, and  HttpSessionEvent |
| Attribute added, removed, or replaced | javax.servlet.http.HttpSessionAttributeListener and  HttpSessionBindingEvent |
| Request | A servlet request has started being processed by web components | javax.servlet.ServletRequestListener and  ServletRequestEvent |
| Attribute added, removed, or replaced | javax.servlet.ServletRequestAttributeListener and  ServletRequestAttributeEvent |

## Scope objects

|  |  |  |
| --- | --- | --- |
| **Scope Object** | **Class** | **Accessible From** |
| Web context | javax.servlet.ServletContext | Web components within a web context. See [Accessing the Web Context](http://docs.oracle.com/javaee/5/tutorial/doc/bnagl.html). |
| Session | javax.servlet.http.HttpSession | Web components handling a request that belongs to the session. See [Maintaining Client State](http://docs.oracle.com/javaee/5/tutorial/doc/bnagm.html). |
| Request | Subtype of javax.servlet.ServletRequest | Web components handling the request. |
| Page | javax.servlet.jsp.JspContext | The JSP page that creates the object. See [Using Implicit Objects](http://docs.oracle.com/javaee/5/tutorial/doc/bnahl.html#bnahn). |

Concurrent access can arise in several situations:

* Multiple web components accessing objects stored in the web context.
* Multiple web components accessing objects stored in a session.
* Multiple threads within a web component accessing instance variables. A web container will typically create a thread to handle each request. If you want to ensure that a servlet instance handles only one request at a time, a servlet can implement the ***SingleThreadModel*** interface. If a servlet implements this interface, you are guaranteed that no two threads will execute concurrently in the servlet’s service method. A web container can implement this guarantee by synchronizing access to a single instance of the servlet, or by maintaining a pool of web component instances and dispatching each new request to a free instance. This interface does not prevent synchronization problems that result from web components accessing shared resources such as static class variables or external objects. In addition, the Servlet 2.4 specification deprecates the SingleThreadModel interface.

## Versions

|  |  |  |  |
| --- | --- | --- | --- |
| **Servlet API версия** | **Релиз** | **Платформа** | **Важнейшие изменения** |
| Servlet 3.1 | [Май 2013](http://jcp.org/en/jsr/detail?id=340) | JavaEE 7, JavaSE 7 | Неблокирующий ввод-вывод, поддержка нестандартных протоколов поверх HTTP |
| Servlet 3.0 | [Декабрь 2009](http://www.javaworld.com/javaworld/jw-02-2009/jw-02-servlet3.html) | JavaEE 6, JavaSE 6 | Pluggability, простота разработки, асинхронные сервлеты, безопасность, загрузка файлов |
| Servlet 2.5 | [Сентябрь 2005](http://www.javaworld.com/javaworld/jw-01-2006/jw-0102-servlet.html) | JavaEE 5 , J2SE 5.0 | Требует J2SE 5.0, поддержка [annotations](http://en.wikipedia.org/wiki/Java_annotation) |
| Servlet 2.4 | [Ноябрь 2003](http://www.javaworld.com/jw-03-2003/jw-0328-servlet.html) | J2EE 1.4, J2SE 1.3 | web.xml использует XML Schema |
| Servlet 2.3 | [Август 2001](http://www.javaworld.com/jw-01-2001/jw-0126-servletapi.html) | J2EE 1.3, J2SE 1.2 | Появление Filter |
| Servlet 2.2 | [Август 1999](http://www.javaworld.com/jw-10-1999/jw-10-servletapi.html) | J2EE 1.2, J2SE 1.2 | Становится частью [J2EE](http://ru.wikipedia.org/wiki/J2EE), предлагает независимые веб-приложения в [.war](http://ru.wikipedia.org/wiki/WAR_%28%D1%82%D0%B8%D0%BF_%D1%84%D0%B0%D0%B9%D0%BB%D0%B0%29) файлах |
| Servlet 2.1 | [Ноябрь 1998](http://www.javaworld.com/jw-12-1998/jw-12-servletapi.html) | не оговорено | Первая официальная спецификация, добавлены RequestDispatcher, ServletContext |
| Servlet 2.0 |  | JDK 1.1 | Часть Java Servlet Development Kit 2.0 |
| Servlet 1.0 | Июнь 1997 |  |  |

## Sharing Data Between Servlets

Version 2.1 of the Servlet API offers a new way of sharing named objects between all the Servlets in a Servlet context (and also other contexts, as you'll see below) by binding the objects to the ServletContext object which is shared by several Servlets.

The ServletContext class has several methods for accessing the shared objects:

* public void setAttribute(String name, Object object) adds a new object or replaces an old object by the specified name. The attribute name should follow the same naming convention as a package name (e.g. a Servlet com.foo.fooservlet.FooServlet could have an attribute com.foo.fooservlet.bar).

Just like a custom ServletRequest attribute, an object which is stored as a ServletContext attribute should also be serializable to allow attributes to be shared by Servlets which are running in different [JVMs](http://www.novocode.com/doc/servlet-essentials/appendix.html#a_d_JVM) on different machines in a load-balancing server environment.

* public Object getAttribute(String name) returns the named object or null if the attribute does not exist.

In addition to the user-defined attributes there may also be predefined attributes which are specific to the Servlet engine and provide additional information about a Servlet(Context)'s environment.

* public Enumeration getAttributeNames() returns an Enumeration of the names of all available attributes.
* public void removeAttribute(String name) removes the attribute with the specified name if it exists.

## FILTERS

The filter declaration order corresponds to the order filters are applied by servlet-container.

<filter>

<filter-name>LogFilter</filter-name>

<filter-class>LogFilter</filter-class>

<init-param>

<param-name>test-param</param-name>

<param-value>Initialization Paramter</param-value>

</init-param>

</filter>

<filter>

<filter-name>AuthenFilter</filter-name>

<filter-class>AuthenFilter</filter-class>

<init-param>

<param-name>test-param</param-name>

<param-value>Initialization Paramter</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>LogFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>AuthenFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

## SERVLET 3.0

1. Простота разработки:
   * благодаря аннотациям программирование приобретает декларативный стиль (@WebServlet, @WebFilter и @WebListener)
   * metadata-complete для элемента web-app. Атрибут определяем, является ли дескриптор полным или требуется пройтись по классам на предмет аннотаций.
2. Компонентность (pluggability) и расширяемость:

Причиной такой архитектуры является то, что сервлет-контейнеры поддерживали только единый монолитный файл дескриптора, в котором определены все сведения о развертывании. С ростом приложения, растет зависимость от внешних структур, в результате дескриптор развертывания тоже растет и усложняется. Как Вы, наверное, знаете, поддержка такого дескриптора может доставить немало хлопот.

Эту проблему решает, концепция Web-фрагментов или модульного web.xml, одна из самых значительных концепций Servlet 3.0. Web-фрагмент - это логический раздел web-приложения, состоящий из таких элементов, как сервлет, отображение сервлета, фильтр, отображение фильтра, слушатели, и их дочерние элементы, упакованные в JAR-файл. Разработчики фрэймворов могут использовать эту возможность для определения своих собственных веб-фрагментов, а разработчики приложений могут подключать фрэймворки, простым копированием JAR-файла в classpath, без редактирования дескриптора развертывания. Проще говоря, эта функция направлена на нулевое конфигурирование при подключении фрэймворка или библиотеки.

Дескриптор развертывания Web-фрагмента находится в файле web-fragment.xml, в каталог META-INF JAR-файла, в целом он похож на дескриптор обычного WEB-приложения. Во время развертывания контейнер сканирует classpath, обнаруживает все веб-фрагменты и обрабатывает их. Атрибут metadata-complete обсуждался ранее в этой статье, он контролирует сканирование web-фрагмента на предмет аннотаций. Пример веб-фрагмента приведен ниже:

1. Асинхронная обработка;
2. Программная регистрация сервлетов
3. Улучшение модели безопасности;
4. прочее.