Parul® University

NAAC A++

Enterprise Programming using JAVA

Chapter-2: Servlets

Prof. ARNIKA PATEL
Assistant Professor
Department of CSE

Parul® University

NAAC A++

Content

1. Basics of Web.....3
2. Servlet Lifecycle.....5
3. Servlets API.....26

INDEX

Basics of Web

A Web application is sometimes called a Web app, Thus, it is an application that is accessed using a Web browser over the network such as the Internet or an Intranet.

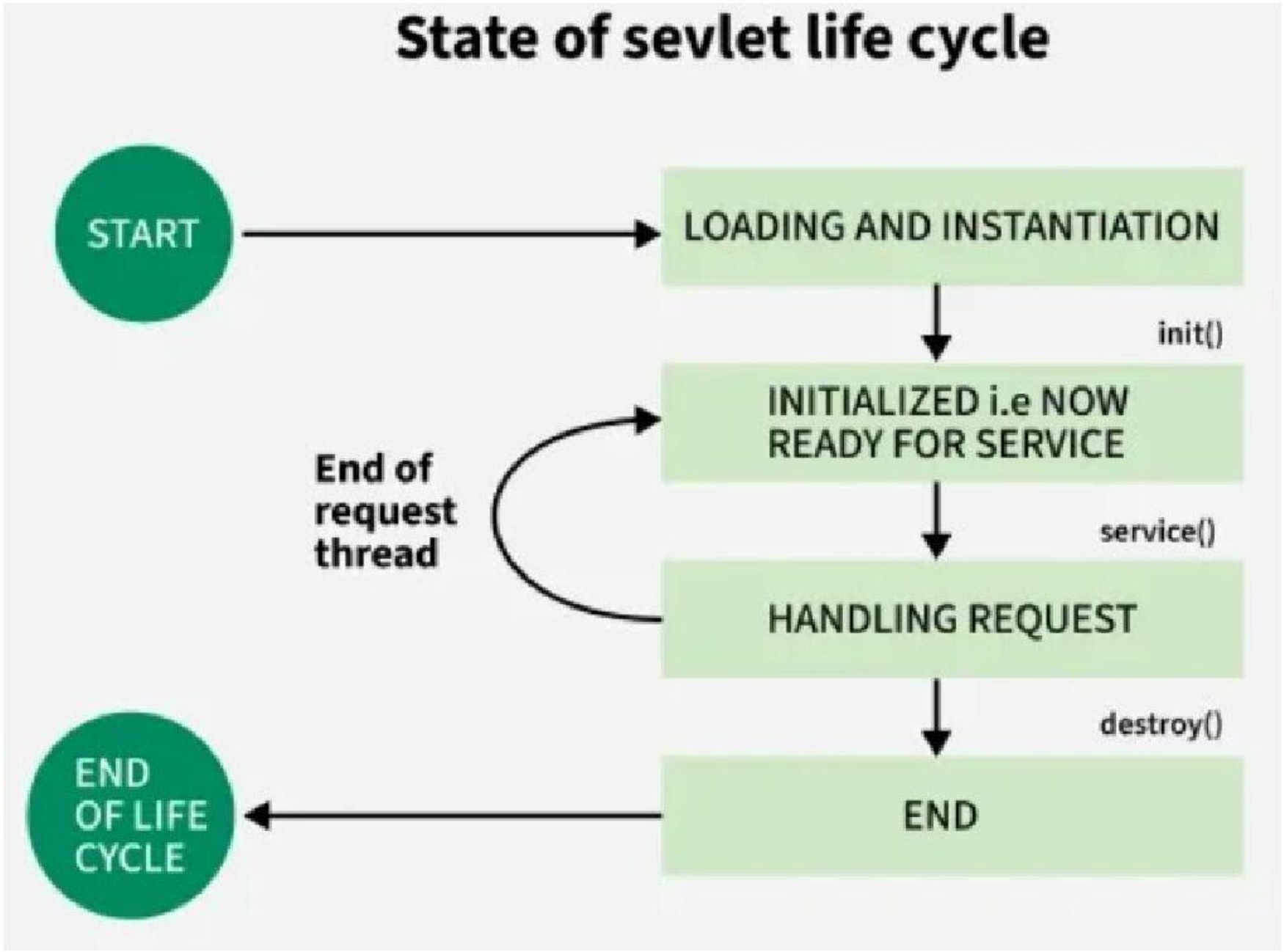
While many of the Web applications are written directly in PHP or Perl, Java remains a commonly used programming language for writing Web applications.

Servlets

Servlets are the Java programs that run on the Java-enabled web server or application server.

They are used to handle the request obtained from the webserver, process the request, produce the response, then send a response back to the webserver.

Servlet LifeCycle



Servlet LifeCycle

The entire life cycle of a Servlet is managed by the Servlet container, which uses the **jakarta.servlet.Servlet** interface to understand the Servlet object and manage it. So, before creating a Servlet object, let's first understand the life cycle of the Servlet object, which is actually understanding how the Servlet container manages the Servlet object.

Servlet LifeCycle

1. Loading a Servlet

Loading: The Servlet container loads the Servlet class into memory.

Instantiation: The container creates an instance of the Servlet using the no-argument constructor.

2. Initializing a Servlet

After the Servlet is instantiated, the Servlet container initializes it by calling the init(ServletConfig config) method. This method is called only once during the Servlet's life cycle.

Servlet LifeCycle

3. Handling request

Once the Servlet is initialized, it is ready to handle client requests. The Servlet container performs the following steps for each request:

- Create Request and Response Objects
- Invoke the service() Method

Servlet LifeCycle

4. Destroying a Servlet

When the Servlet container decides to remove the Servlet, it follows these steps which are listed below

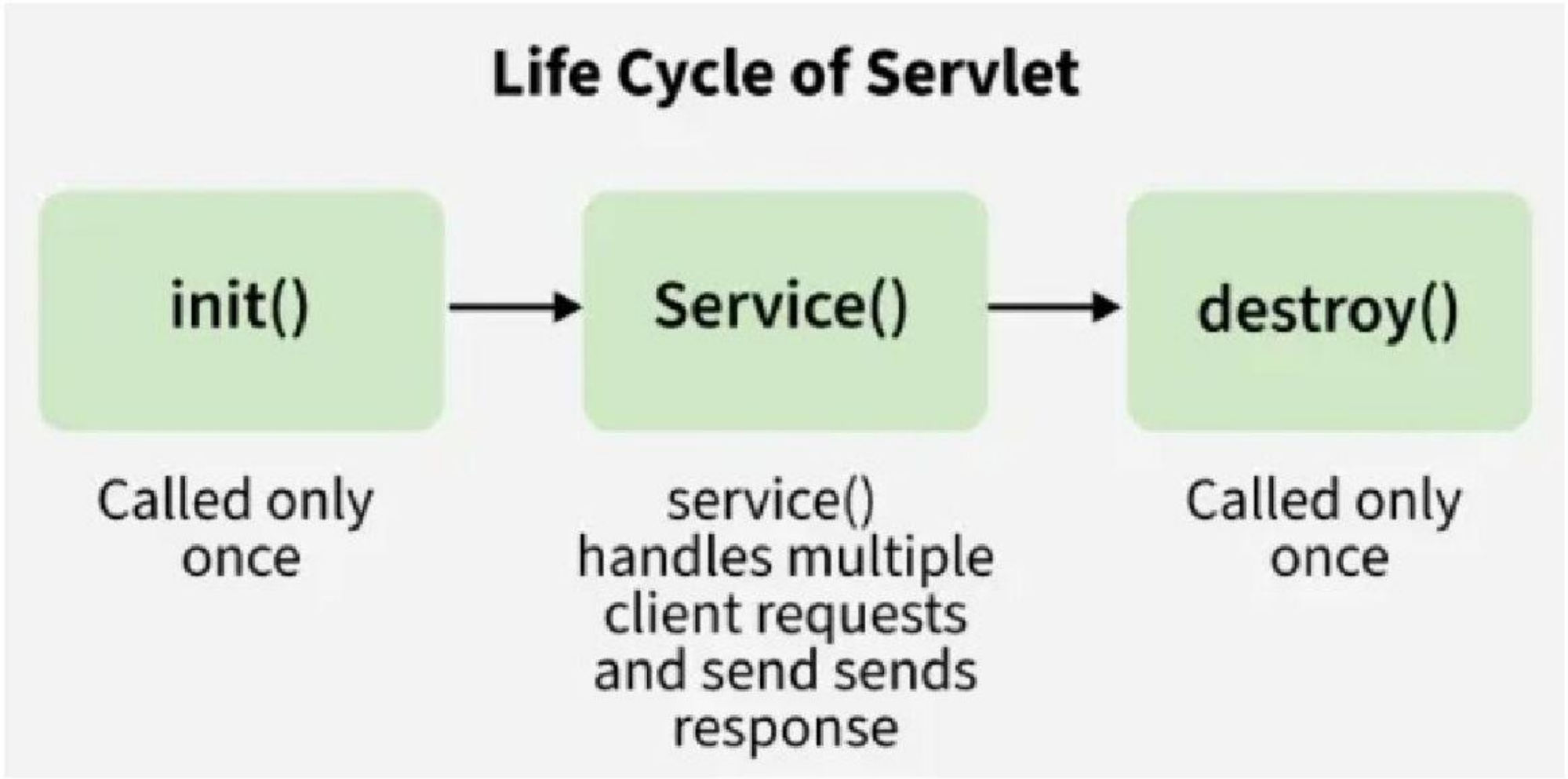
- Allow Active Threads to Complete
- Invoke the destroy() Method
- Release Servlet Instance

Servlet LifeCycle Methods

There are three life cycle methods of a Servlet:

init()
service()
destroy()

Servlet LifeCycle Methods



Servlet LifeCycle Methods

The servlet life cycle consists of these stages:

- Servlet is created
- Servlet is initialized
- Servlet is ready to service
- Servlet is servicing
- Servlet is not ready to service
- Servlet is destroyed

Servlet LifeCycle Methods

1. init() Method

This method is called by the Servlet container to indicate that this Servlet instance is instantiated successfully and is about to put into the service.

```
// init() method
public class MyServlet implements Servlet {
        public void init(ServletConfig config) throws
ServletException
        {
                // initialization code
        }
        // rest of code
}
```

Servlet LifeCycle Methods

1. init() Method

A Servlet life begins from this method. This method is called only once to load the servlet.

Question:

Why can't we write connected architecture code inside the constructor, since constructor also run only once in it's entire life?

Servlet LifeCycle Methods

1. init() Method

Answer:

Suppose if the connection doesn't get established, then we can throw an exception from init() and the rest of the steps stop executing. But in the constructor we can't use, throw in it's prototype otherwise it is an error

Servlet LifeCycle Methods

1. init() Method

Question:

Why it is recommended to use the non parameterized version of init() instead of parameterized version?

Servlet LifeCycle Methods

1. **init() Method: Using the Parameterized init(ServletConfig.con)**

Answer:

When the parameterized init(ServletConfig con) is called during the servlet life cycle, the following steps occur:

- First the overridden init(ServletConfig con) in our servlet class is executed.
- Then the init(ServletConfig con) method in the parent HttpServlet class is called using super.init(con).
- Finally the init() (non-parameterized version) in the parent HttpServlet class is invoked. However, this method has an empty body, making the call redundant.

Servlet LifeCycle Methods

1. init() Method: Using the Parameterized init(ServletConfig.con)

```
public void init(ServletConfig con) throws ServletException
{
        super.init(con); // Calls HttpServlet's
init(ServletConfig)
        // Custom database connection code here
}
```

Servlet LifeCycle Methods

1. **init() Method: Using the Parameterized init(ServletConfig.con)**

Disadvantages:

More Calls: Total 3 init() calls

- Parameterized init(ServletConfig con) in our servlet class.
- Parameterized init(ServletConfig con) in HttpServlet.
- Non-parameterized init() in HttpServlet(empty body, hence useless)

Performace Overhead: Extra calls increase stack usage and execution time.

Servlet LifeCycle Methods

1. init() Method: Using the Non-Parameterized init()

- Instead of overriding the parameterized init(servletConfig con), override the non-parameterized init().
- During the servlet life cycle
  - The parameterized init(servletConfig con) in theHttpServlet class is called automatically.
  - This method then calls the non-parameterizedinit() method of our servlet directly.

```
public void init() throws ServletException
{
        // Custom database connection code here
}
```

Servlet LifeCycle Methods

Advantages:

Fewer Calls: Total of 2 init() calls

- Parameterized init(ServletConfig con) in HttpServlet.
- Non-parameterized init() in our servlet class.

Servlet LifeCycle Methods

2. service() Method

This method is used to inform the Servlet about the client requests

- This method uses ServletRequest object to collect the data requested by the client
- This method uses ServletResponse object to generate the output content

Servlet LifeCycle Methods

2. service() Method

```
// service() method
public class MyServlet implements Servlet {
        public void service(ServletRequest req,
        ServletResponse res) throws ServletException,
        IOException
        {
                // request handling code
        }
        // rest of code
}
```

Servlet LifeCycle Methods

3. destroy() Method

This method runs only once during the lifetime of a signals the end of the Servlet instance.

```
// destroy() method
public void destroy()
{
        // code
}
```

Servlet LifeCycle Methods

3. destroy() Method

- The destroy() method is called only once.
- It is called at the end of the life cycle of the servlet.
- This method performs various tasks such as closing connection with the database, releasing memory allocated to the servlet, releasing resources that are allocated to the servlet and other cleanup activities.
- After destroy() method the Servlet instance becomes eligible for garbage collection.

Servlet API

- Servlets are the Java programs that run on the Java-enabled web server or application server.
- They are used to handle the request obtained from the webserver, process the request, produce the response, then send a response back to the webserver.
- In Java, to create web applications we use Servlets.
- To create Java Servlets, we need to use Servlet API which contains all the necessary interfaces and classes. Servlet API has 2 packages namely,

    1. javax.servlet
    2. javax.servlet.http

Servlet API

- **javax.servlet**
  - This package provides the number of interfaces and classes to support Generic servlet which is protocol independent.
  - These interfaces and classes describe and define the contracts between a servlet class and the runtime environment provided by a servlet container.

Servlet API

javax.servlet: Classes available in javax.servlet package:

1. **GenericServlet:** To define a generic and protocol-independent servlet.
2. **ServletContextAttributeEvent:** To generate notifications about changes to the attributes of the servlet context of a web application.
3. **ServletContextEvent:** To generate notifications about changes to the servlet context of a web application.
4. **ServletInputStream:** This class provides an input stream to read binary data from a client request.

Servlet API

javax.servlet: Classes available in javax.servlet package:

5. **ServletOutputStream:** This class provides an output stream for sending binary data to the client.
6. **ServletRequestAttributeEvent:** To generate notifications about changes to the attributes of the servlet request in an application.
7. **ServletRequestEvent:** To indicate lifecycle events for a ServletRequest.

Servlet API

javax.servlet: Classes available in javax.servlet package:

8. **ServletRequestWrapper :** This class provides the implementation of the ServletRequest interface that can be subclassed by developers to adapt the request to a Servlet.
9. **ServletResponseWrapper:** This class provides the implementation of the ServletResponse interface that can be subclassed by developers to adapt the response from a Servlet.

Servlet API

javax.servlet: Interfaces available in javax.servlet package:

1. **Filter:** To perform filtering tasks on either the request to a resource, or on the response from a resource, or both.
2. **FilterChain:** To provide a view into the invocation chain of a filtered request for a resource to the developer by the servlet container.
3. **FilterConfig:** To pass information to a filter during initialization used by a servlet container.
4. **RequestDispatcher:** It defines an object to dispatch the request and response to any other resource, means it receives requests from the client and sends them to a servlet/HTML file/JSP file on the server.

Servlet API

javax.servlet: Interfaces available in javax.servlet package:

5. **Servlet:** This is the main interface that defines the methods in which all the servlets must implement. To implement this interface, write a generic servlet that extends javax.servlet.GenericServlet or an HTTP servlet that extends javax.servlet.http.HttpServlet.
6. **ServletConfig:** It defines an object created by a servlet container at the time of servlet instantiation and to pass information to the servlet during initialization.

Servlet API

javax.servlet: Interfaces available in javax.servlet package:

7. **ServletContext:** It defines a set of methods that a servlet uses to communicate with its servlet container. The information related to the web application available in web.xml file is stored in ServletContext object created by container.
8. **ServletContextAttributeListener:** The classes that implement this interface receive notifications of changes to the attribute list on the servlet context of a web application.

Servlet API

javax.servlet: Interfaces available in javax.servlet package:

9. **ServletRequest:** It defines an object that is created by servlet container to pass client request information to a servlet.
10. **ServletRequestAttributeListener:** To generate the notifications of request attribute changes while the request is within the scope of the web application in which the listener is registered.
11. **ServletRequestListener:** To generate the notifications of requests coming in and out of scope in a web component.

Servlet API

javax.servlet: Interfaces available in javax.servlet package:

12. **ServletResponse:** It defines an object created by servlet container to assist a servlet in sending a response to the client.

Servlet API

javax.servlet: Exceptions in javax.servlet package:

1. **ServletException:** A general exception thrown by a servlet when it encounters difficulty.
2. **UnavailableException:** Thrown by a servlet or filter to indicate that it is permanently or temporarily unavailable.

Servlet API

javax.servlet.http

- This package provides the number of interfaces and classes to support HTTP servlet which is HTTP protocol dependent.
- These interfaces and classes describe and define the contracts between a servlet class running under HTTP protocol and the runtime environment provided by a servlet container.

Servlet API

javax.servlet.http: Classes available in javax.servlet.http package:

1. **Cookie:** Creates a cookie object. It is a small amount of information sent by a servlet to a Web browser, saved by the browser, and later sent back to the server used for session management.
2. **HttpServlet:** Provides an abstract class that defines methods to create an HTTP suitable servlet for a web application.
3. **HttpServletRequestWrapper:** This class provides implementation of the HttpServletRequest interface that can be subclassed to adapt the request to a Servlet.

Servlet API

javax.servlet.http: Classes available in javax.servlet.http package:

4. **HttpServletResponseWrapper:** This class provides implementation of the HttpServletResponse interface that can be subclassed to adapt the response from a Servlet.
5. **HttpSessionBindingEvent:** This events are either sent to an object that implements HttpSessionBindingListener when it is bound or unbound from a session, or to a HttpSessionAttributeListener that has been configured in the deployment descriptor when any attribute is bound, unbound or replaced in a session.

Servlet API

javax.servlet.http: Classes available in javax.servlet.http package:

6. **HttpSessionEvent:** To represent event notifications for changes to sessions within a web application.

Servlet API

javax.servlet.http: Interfaces available in javax.servlet.http package:

1. **HttpServletRequest:** To provide client HTTP request information for servlets. It extends the ServletRequest interface.
2. **HttpServletResponse:** To provide HTTP-specific functionality in sending a response to client. It extends the ServletResponse interface.
3. **HttpSession:** It provides a way to identify a user across web application/web site pages and to store information about that user.

Servlet API

javax.servlet.http: Interfaces available in javax.servlet.http package:

4. **HttpSessionActivationListener:** Container to notify all the objects that are bound to a session that sessions will be passivated and that session will be activated.
5. **HttpSessionAttributeListener:** To get notifications of changes to the attribute lists of sessions within this web application, this listener interface can be implemented.
6. **HttpSessionBindingListener:** It causes an object to be notified by an HttpSessionBindingEvent object, when it is bound to or unbound from a session.

Servlet API

javax.servlet.http: Interfaces available in javax.servlet.http package:

7. **HttpSessionListener:** To receive notification events related to the changes to the list of active sessions in a web application.

Servlet API

GenericServlet Class

Servlet API provide ***GenericServlet*** class in javax.servlet package.

public abstract class GenericServlet
extends java.lang.Object
implements Servlet, ServletConfig, java.io.Serializable

.

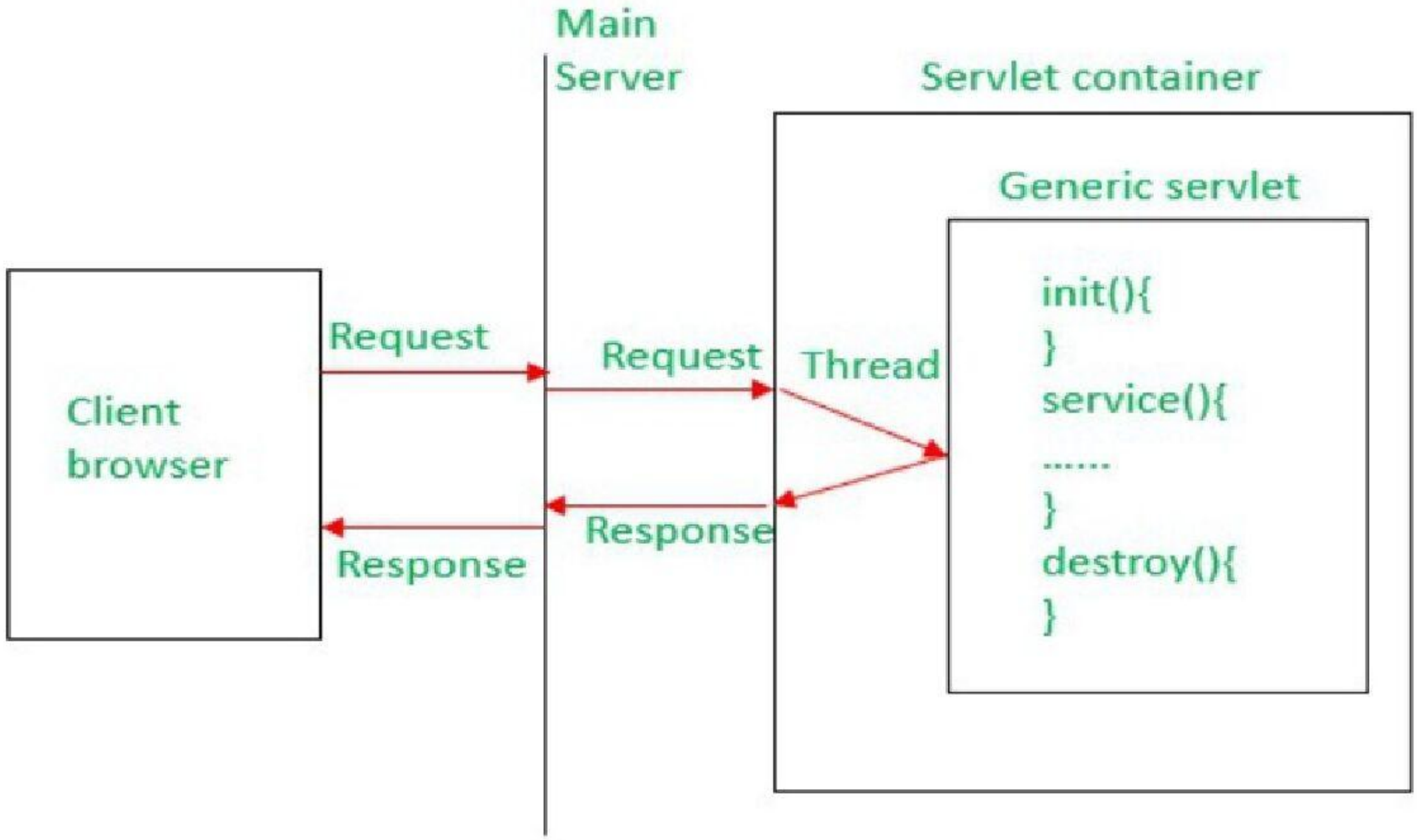
Servlet API

GenericServlet Class

- An abstract class that implements most of the servlet basic methods.
- Implements the Servlet, ServletConfig, and Serializable interfaces.
- Protocol-independent servlet.
- Makes writing servlets easier by providing simple versions of the lifecycle methods init() and destroy().
- To write a generic servlet, you need to extend ***javax.servlet.GenericServlet*** class and need to override the abstract service() method.

Servlet API

GenericServlet Class



Servlet API

HttpServlet Class

Servlet API provides ***HttpServlet*** class in javax.servlet.http package.

public abstract class HttpServlet
extends GenericServlet
implements java.io.Serializable

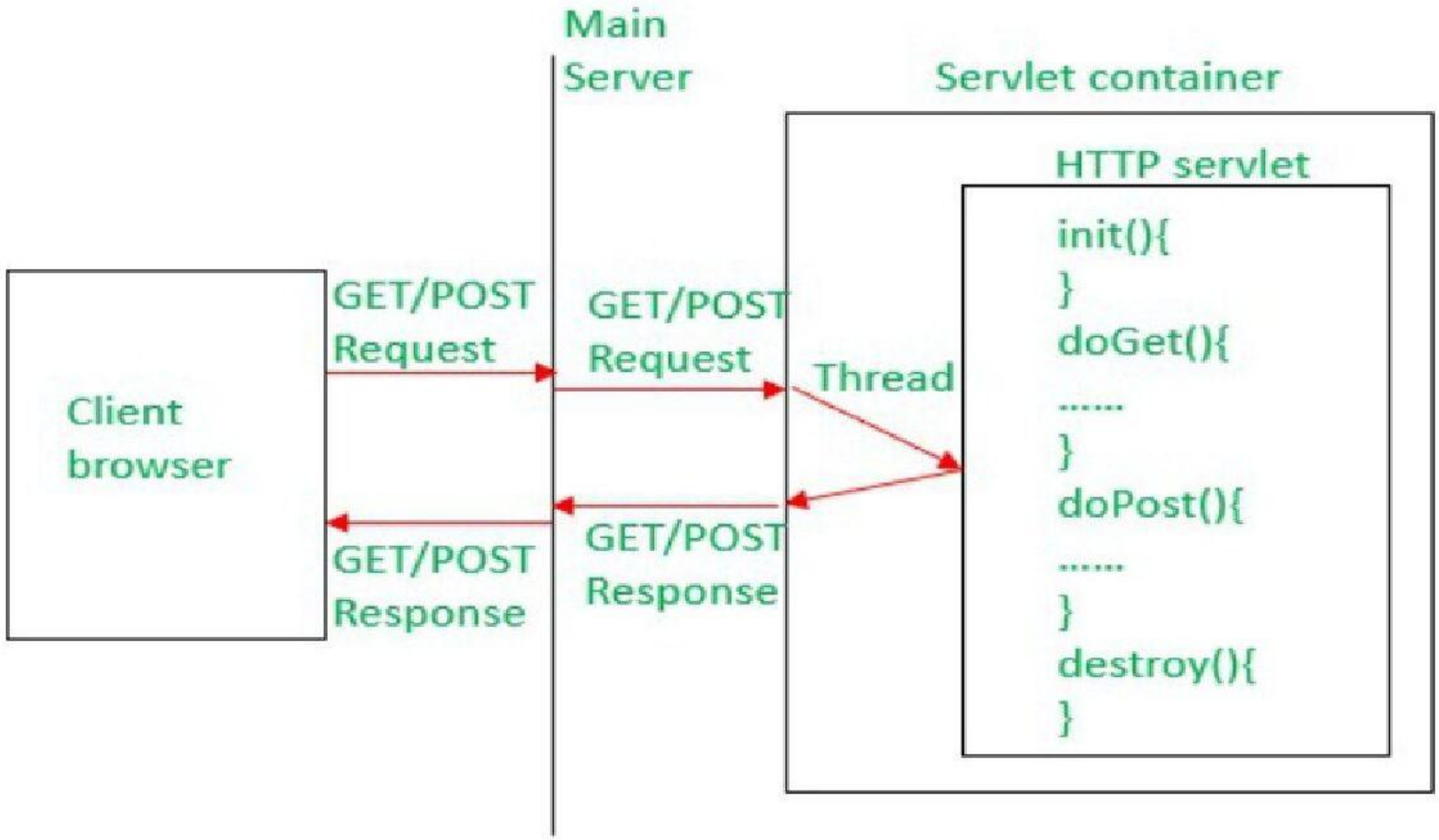
Servlet API

HttpServlet Class

- An abstract class to be subclassed to create an HTTP-specific servlet that is suitable for a Web site/Web application.
- Extends Generic Servlet class and implements Serializable interface.
- HTTP Protocol-dependent servlet.

Servlet API

HttpServlet Class



Servlet API

HttpServlet Class

To write a Http servlet, you need to extend ***javax.servlet.http.HttpServlet*** class and must override at least one of the below methods,

doGet() – to support HTTP GET requests by the servlet.
doPost() – to support HTTP POST requests by the servlet.
doPut() – to support HTTP PUT requests by the servlet.
doDelete() – to support HTTP DELETE requests by the servlet.
init() and destroy() – to manage resources held in the life of the servlet.

Servlet API

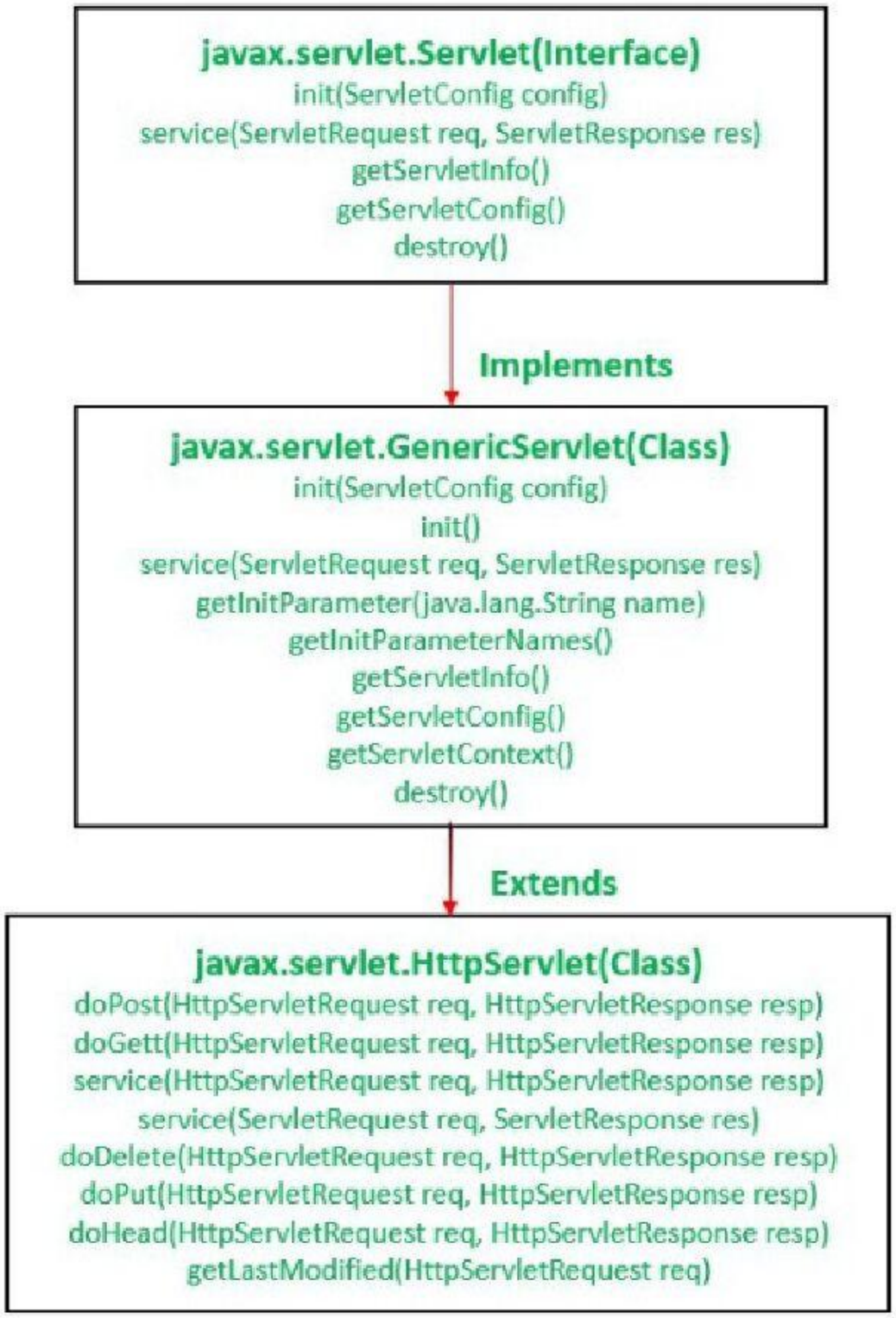
HttpServlet Class

getServletInfo() – To provide information about the servlet itself like the author of servlet or version of it etc.

Servlet API

Servlet API Package Hierarchy

- Servlet API provides all the required interfaces, classes, and methods to develop a web application.
- we need to add the ***servlet-api.jar*** file in our web application to use the servlet functionality.
- We can download this jar file from Maven Repository.



PPT Content Resources Reference Sample:

1. **Book Reference**

   Jim Farley, William Crawford, David Flanagan. Java Enterprise in a Nutshell, O'Reilly

2. **Book Reference**

   Rocha, R., Purificação, J. (2018). Java EE 8 Design Patterns and Best Practices: Build Enterprise-ready Scalable Applications with Architectural Design Patterns. Germany: Packt Publishing..

3. **Website Reference**

   https://www.scribd.com/document/268349254/Java-8-Programming-Black-Book .

4. **Sources**

   https://developers.redhat.com/topics/enterprise-java

5. **Article**

   https://www.researchgate.net/publication/276412369_Advanced_Java_Programming

Parul® University | NAAC GRADE A++

https://paruluniversity.ac.in/