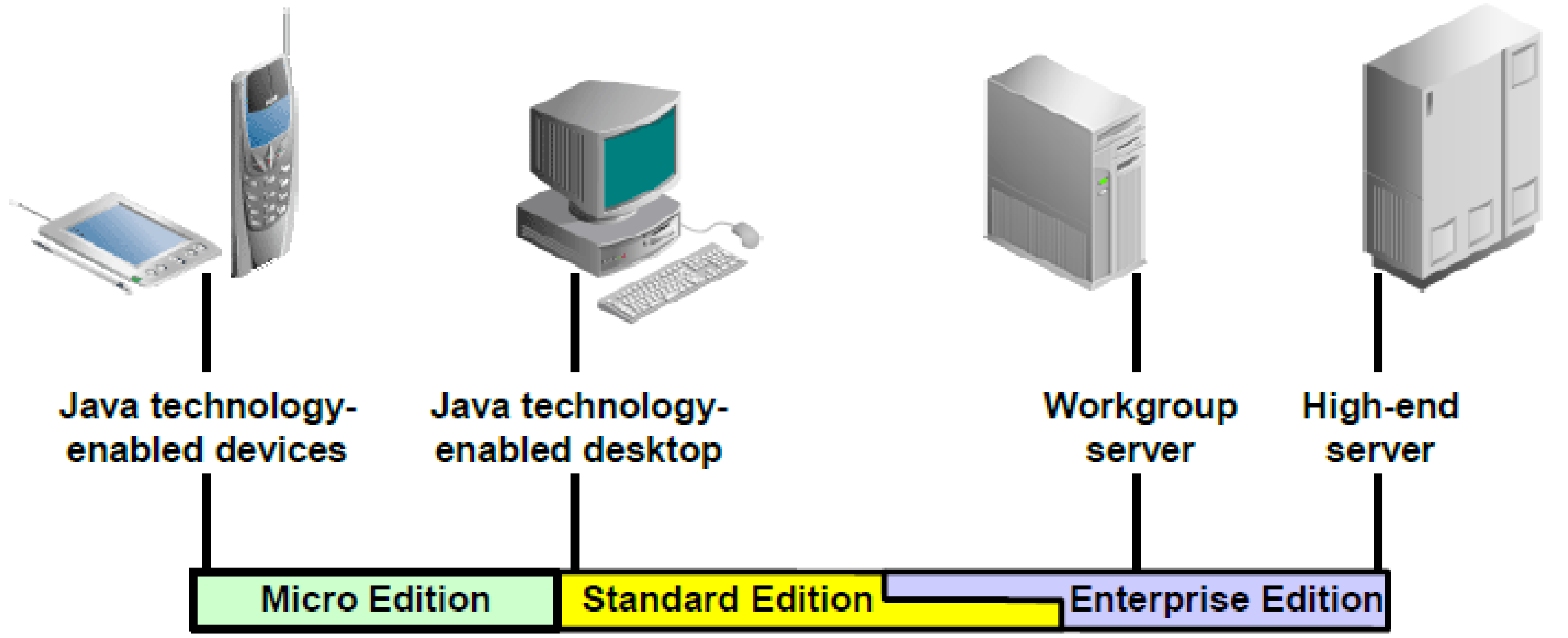
Aplicaciones Web con Java

emaravi@cjavaperu.com

Java Enterprise Edition

Plataforma Java



Java EE = Java Enterprise Edition = JEE

Necesidades de una aplicación empresarial

Build from the ground up

Developer's Checklist

- ☐ Business services
- ☐ Persistence
- ☐ Transaction management
- ☐ Multithreading
- ☐ Security management
- ☐ Networking
- ☐ Service publishing

Use Application Component Server

Developer's Checklist

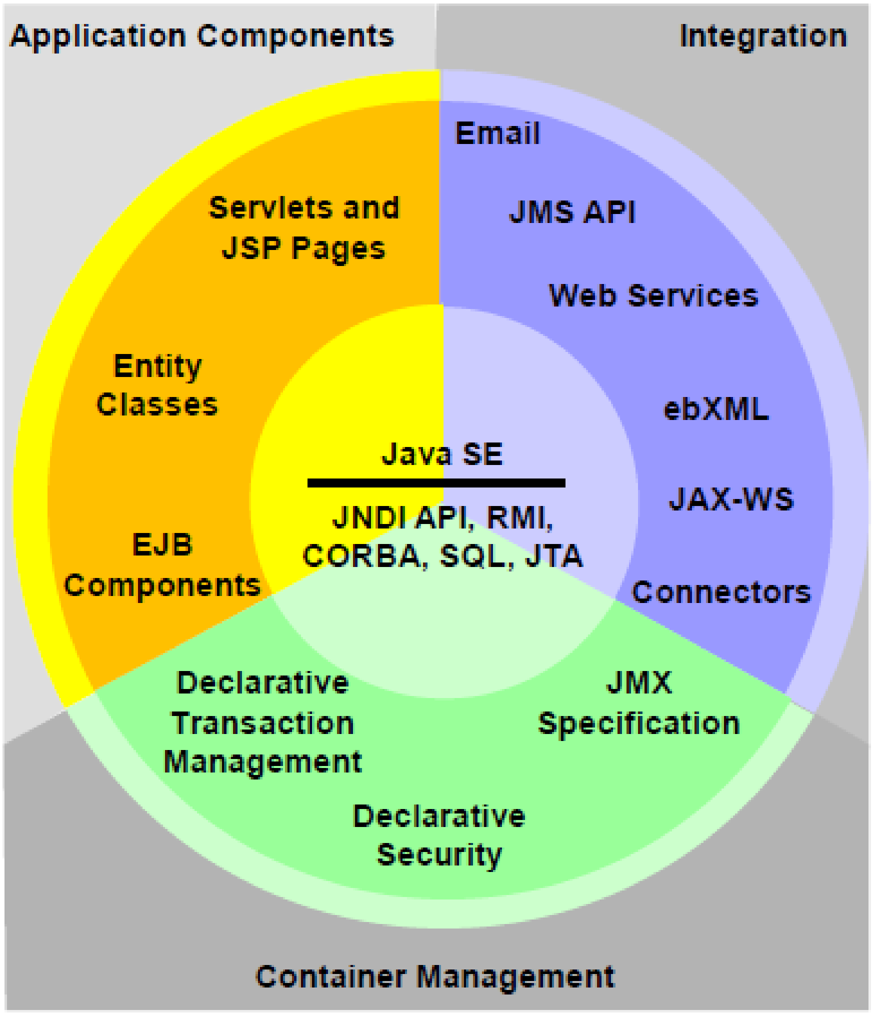
- ☐ Business services

Services Provided by Server

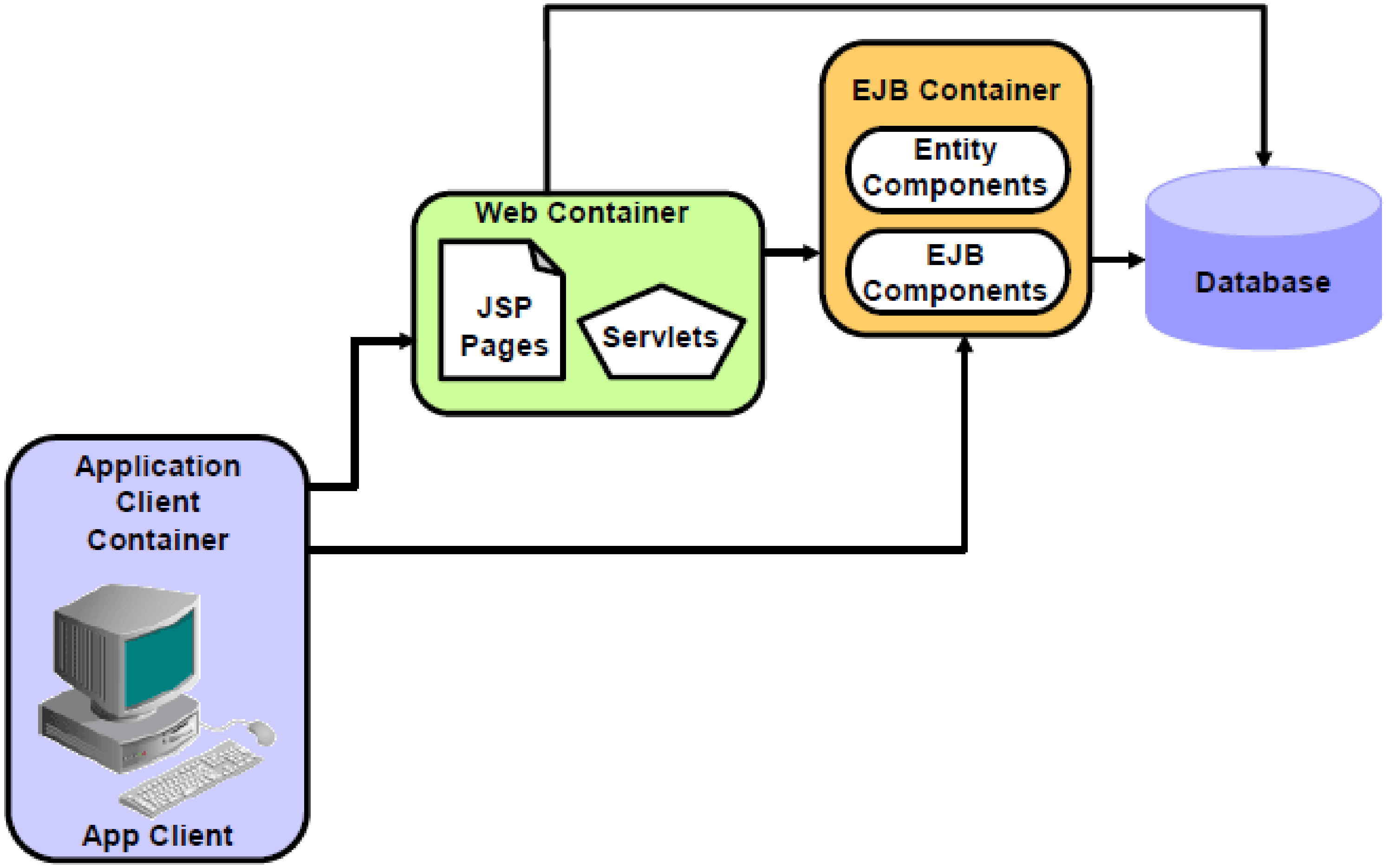
- ✓ Persistence and Transaction management
- ✓ Multithreading
- ✓ Security management
- ✓ Networking
- ✓ Service publishing

Lo proporciona JEE

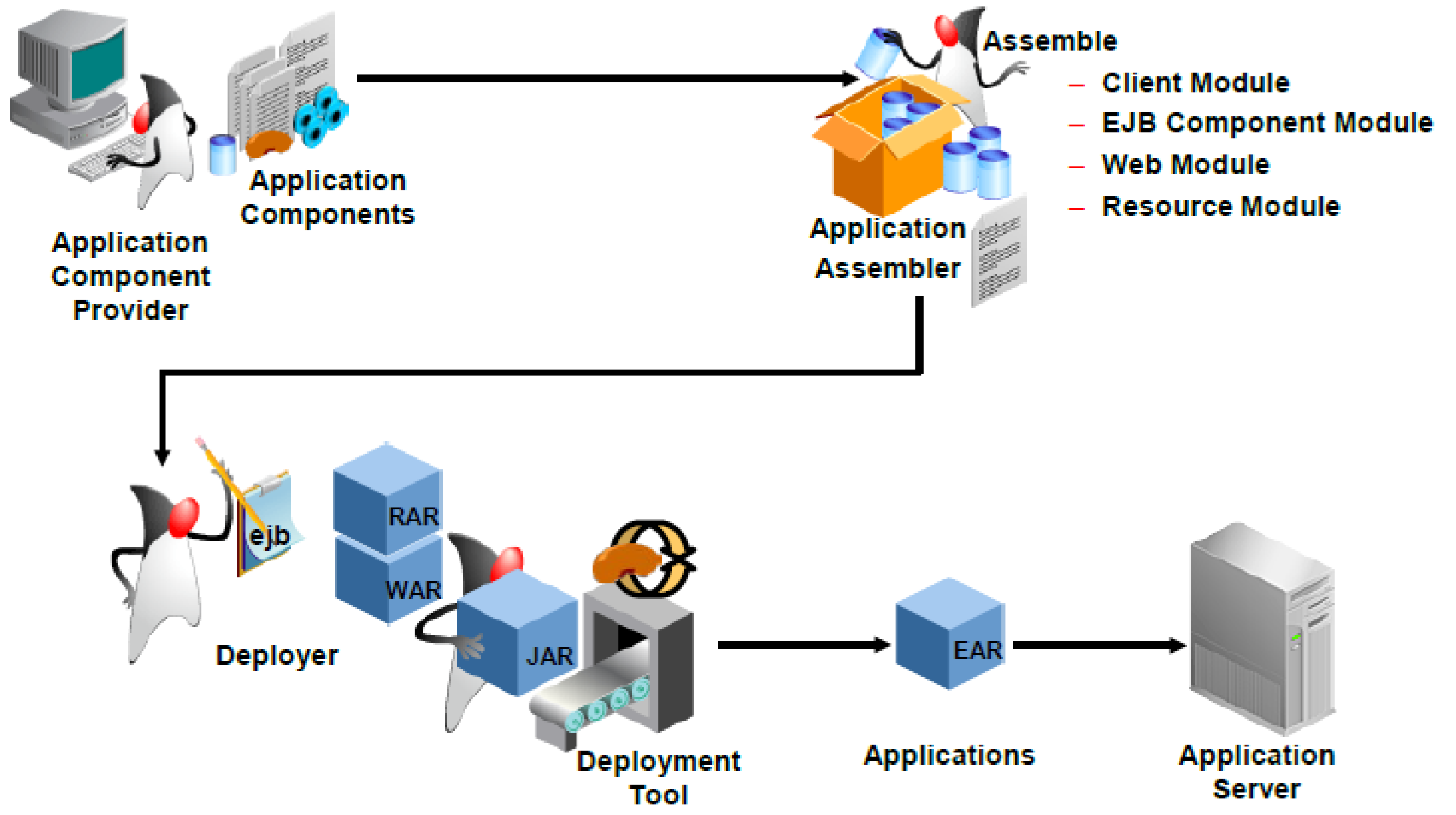
Java Enterprise Edition (JEE)



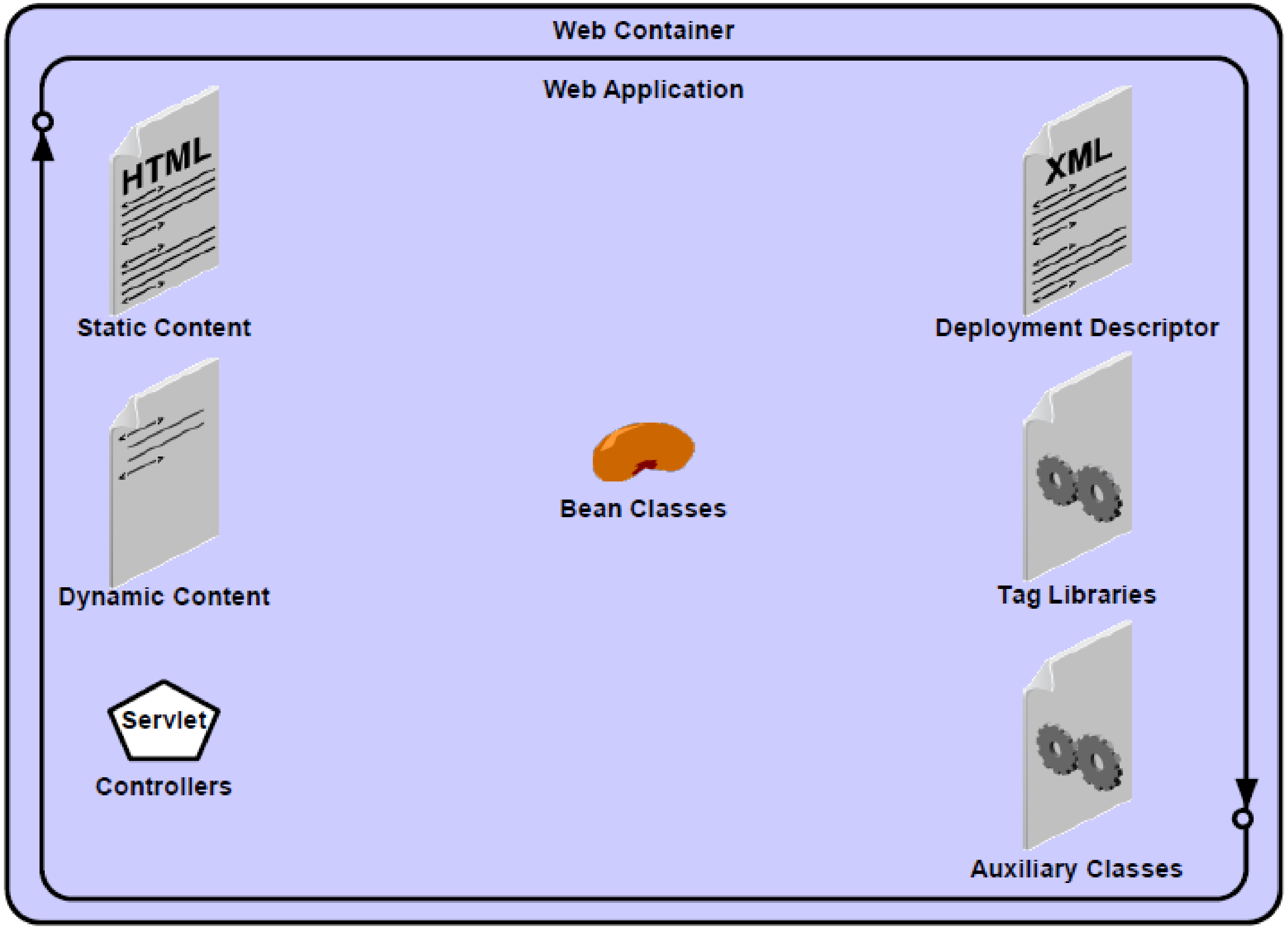
Algunos componentes de JEE



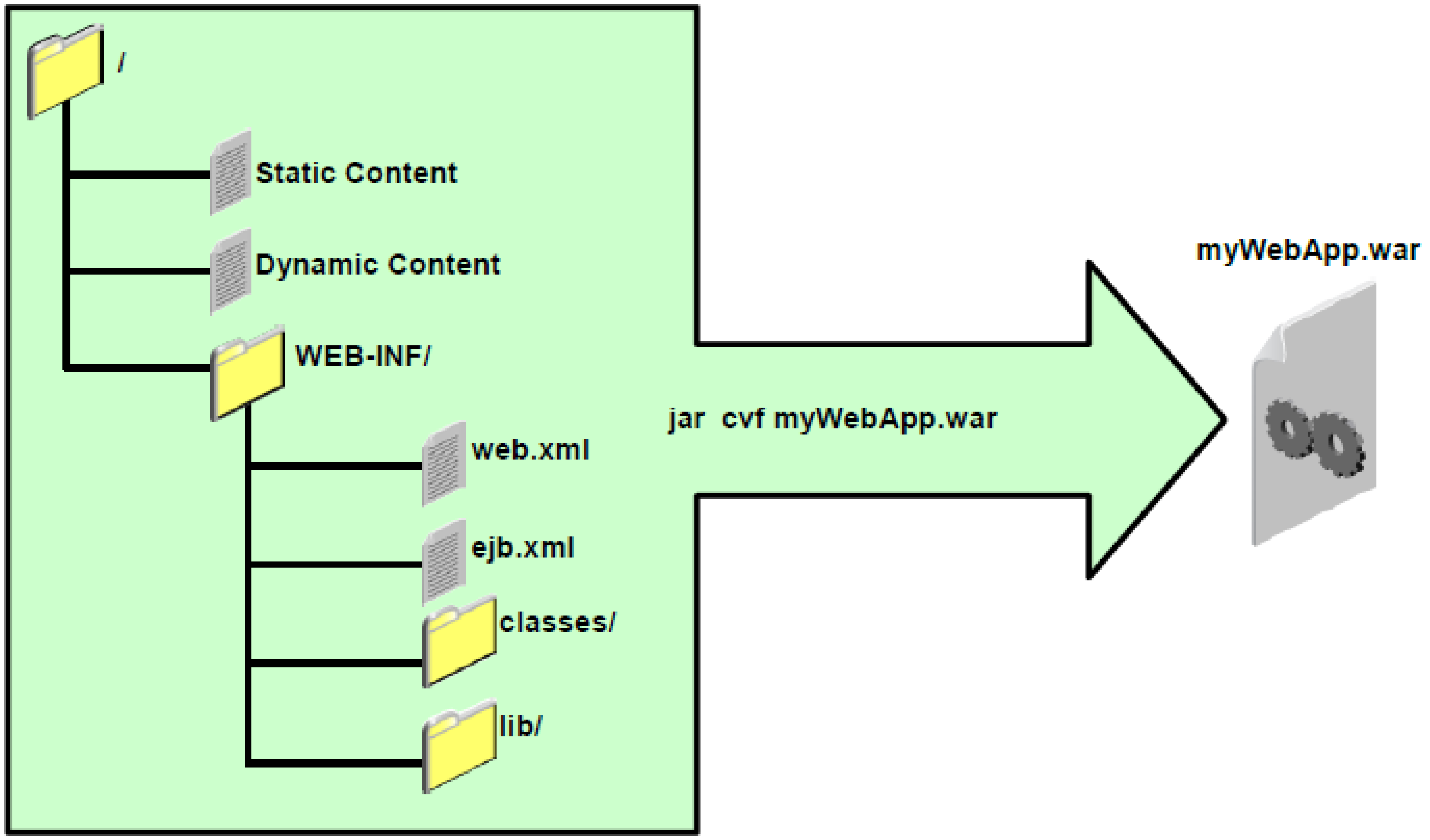
Proceso de desarrollo en JEE



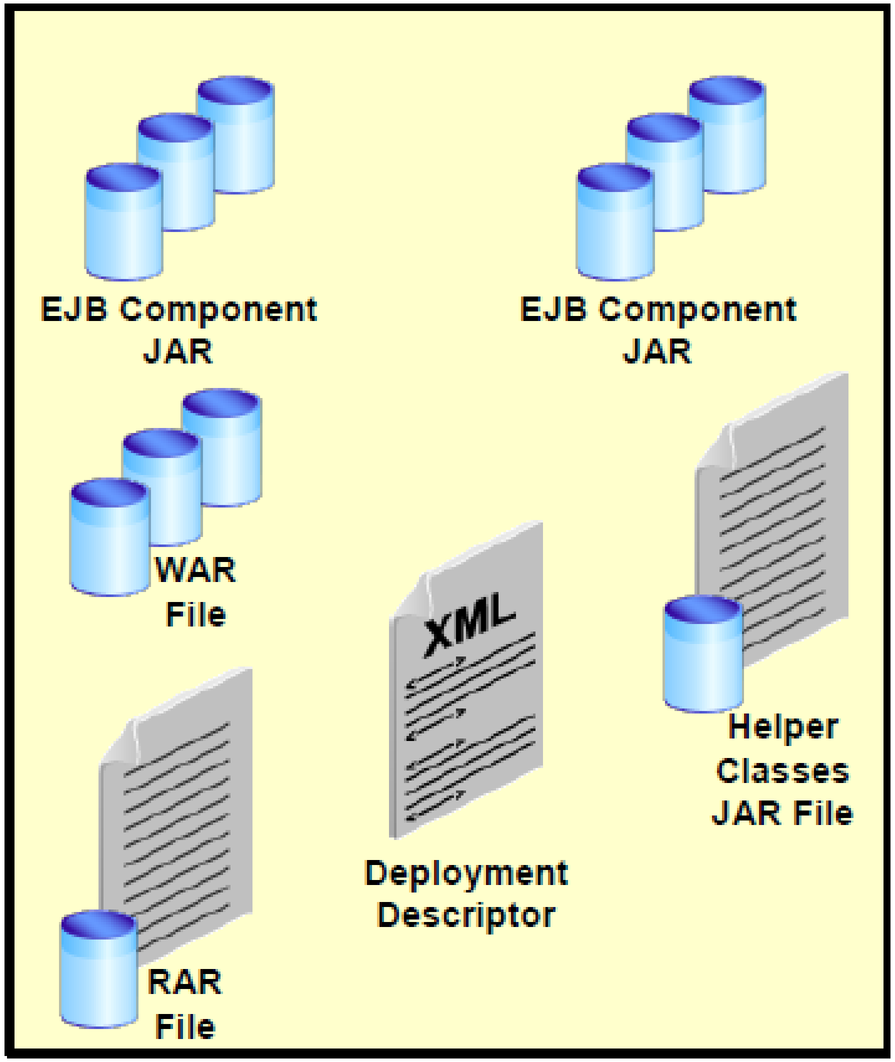
Aplicacion Web (WAR)



Estructura de una aplicación Web

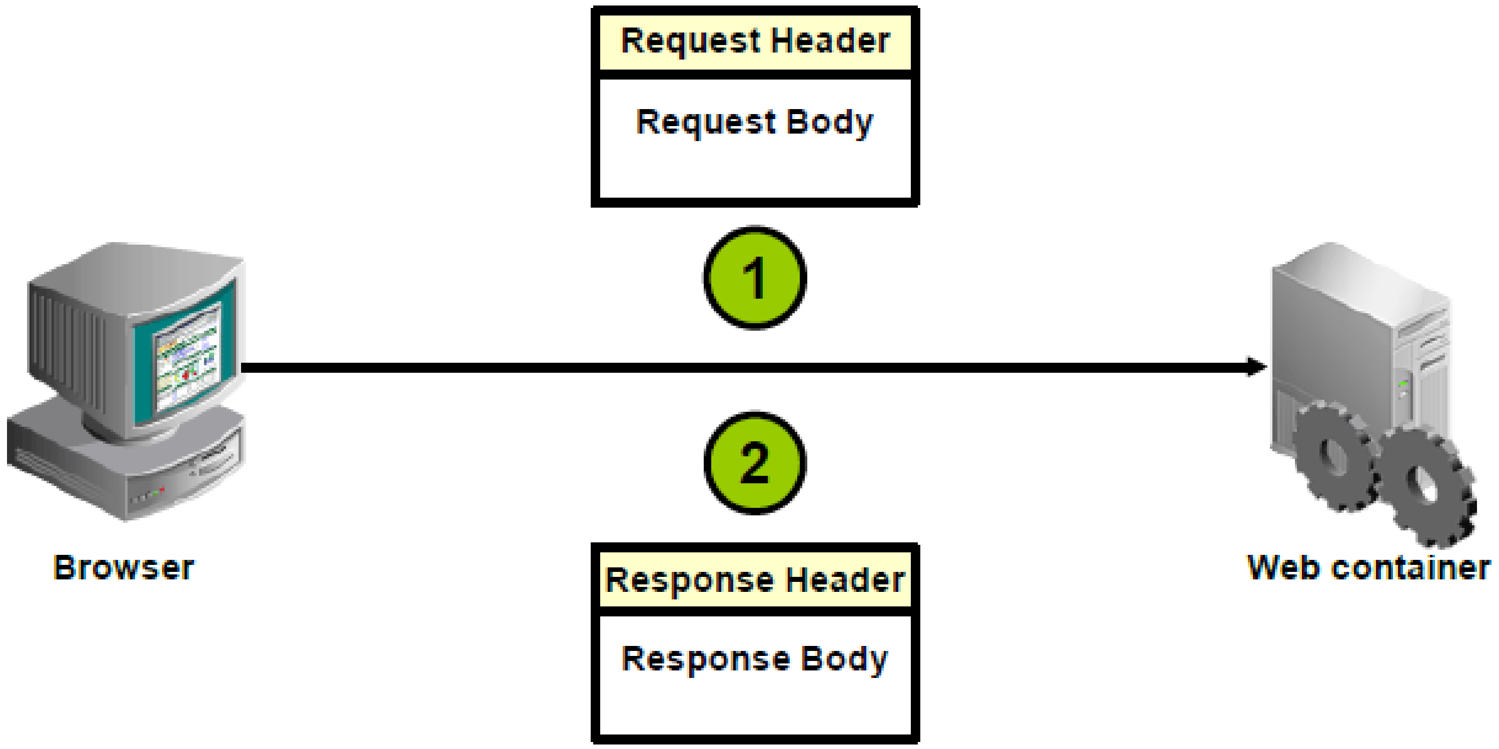


Aplicación empresarial (EAR)

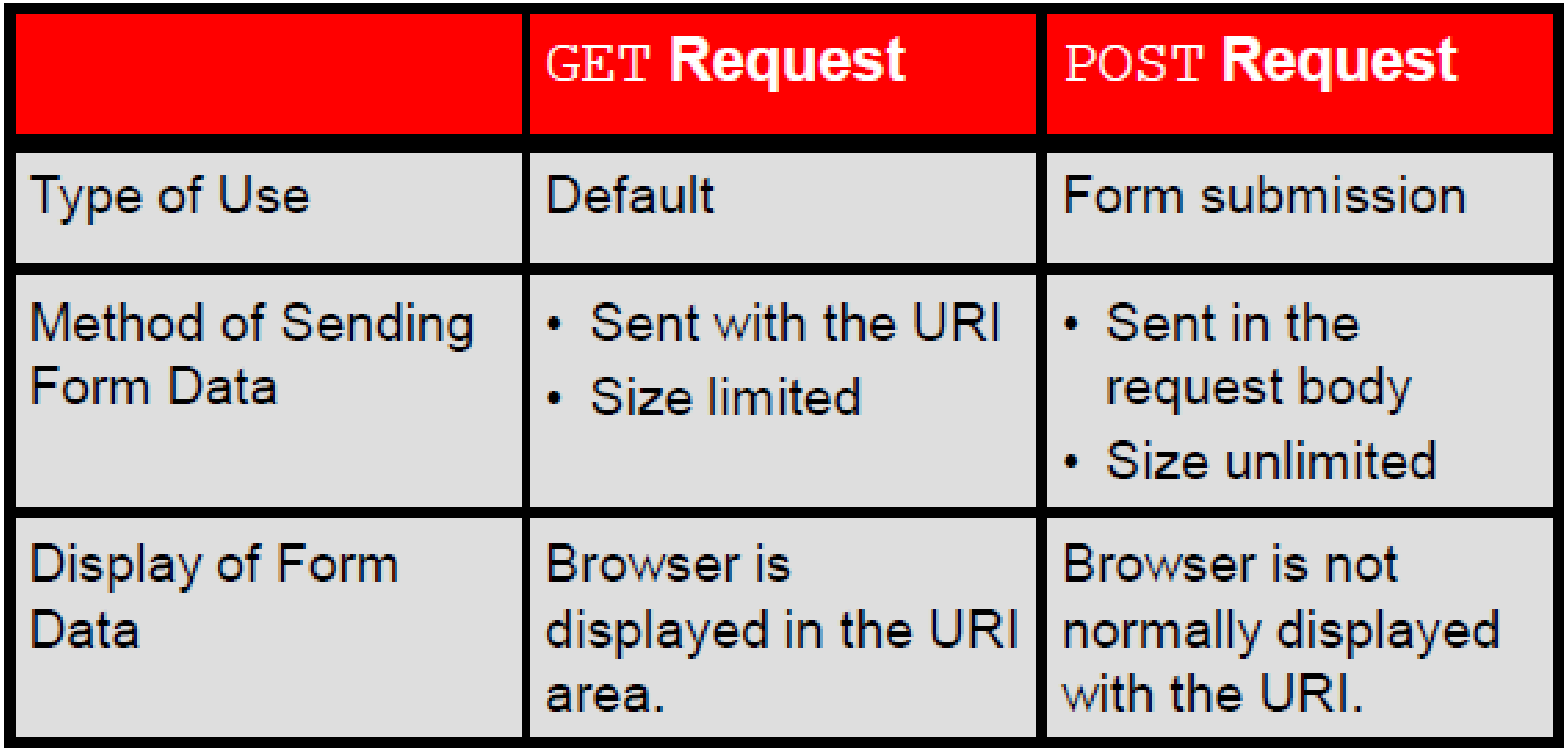


Aplicaciones Web

Protocolo HTTP



Metodos de envío GET y POST



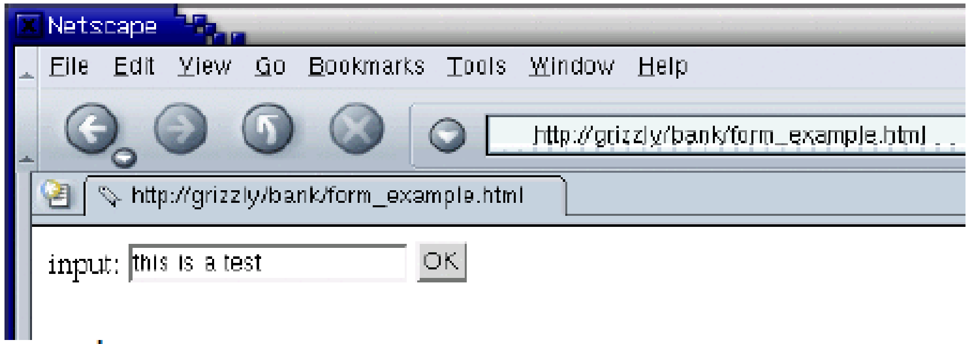
	GET **Request**	POST **Request**
Type of Use	Default	Form submission
Method of Sending Form Data	• Sent with the URI • Size limited	• Sent in the request body • Size unlimited
Display of Form Data	Browser is displayed in the URI area.	Browser is not normally displayed with the URI.

Formularios HTML

HTML snippet:

```
<FORM ACTION='form_test' METHOD='POST'>
<INPUT NAME='input1' SIZE='20'/>
<INPUT TYPE='SUBMIT' VALUE='OK'/>
</FORM>
```

Browser form:


Netscape
File Edit View Go Bookmarks Tools Window Help
http://grizzly/bank/form_example.html
http://grizzly/bank/form_example.html
input: this is a test
OK

Browser request:

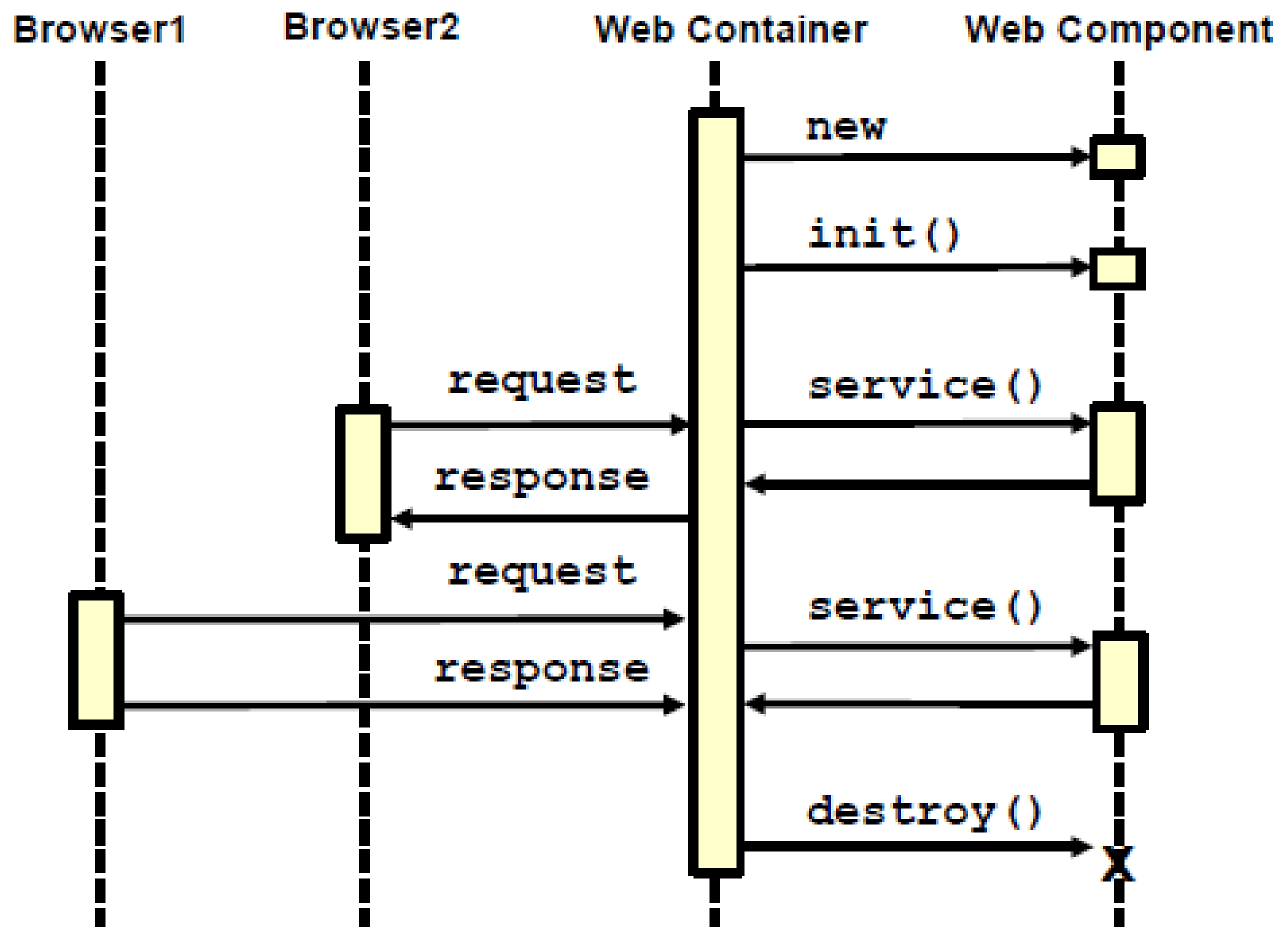
```
POST /bank/form_test HTTP/1.1
... request headers...

input1=this+is+a+test
```

Servlet vs JSP

	Servlets	JSP Components
Description	Web components authored in the Java programming language	Presentation content with embedded programmatic elements
Characteristics	Extend generic base classes in the API, typically the `HttpServlet` interface	• Can be enhanced with custom tags • Are translated into servlets by the web container
Created or Managed By	Developers	Content authors

Ciclo de vida de un componente Web



Ejemplo de Servlet

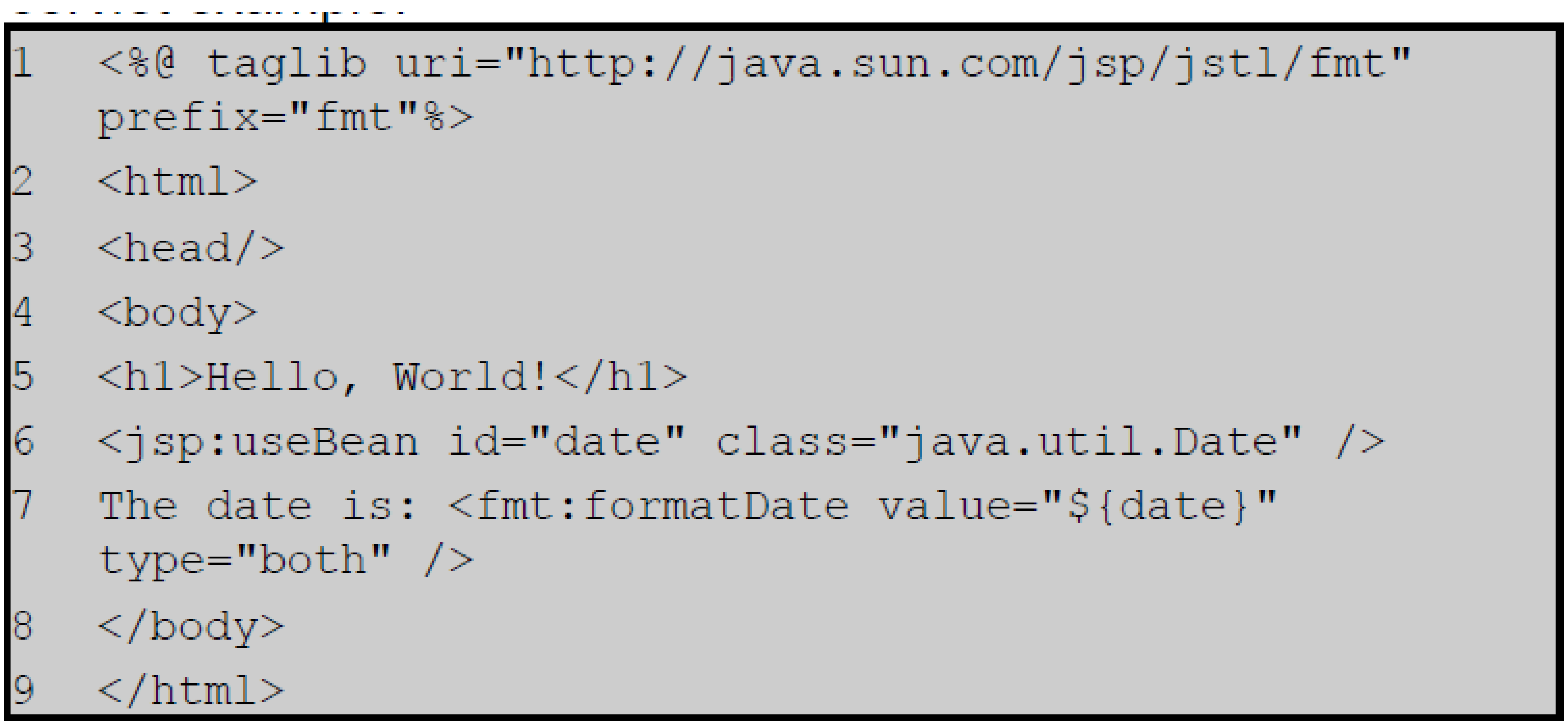
```java
package com.example;

import java.io.*;
import java.util.Date;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;

@WebServlet("/hello")
public class HelloServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><head/><body>");
        out.println("<h1>Hello, World!</h1>");
        out.println("The date is:" + new Date());
        out.println("</body></html>");
        out.close();
    }
}
```

Ejemplo de JSP



```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<html>
<head/>
<body>
<h1>Hello, World!</h1>
<jsp:useBean id="date" class="java.util.Date" />
The date is: <fmt:formatDate value="${date}" type="both" />
</body>
</html>
```

Colaboración de JSP y Servlet

	Servlet	JSP Component
Type of Operation	• Process form data • Perform computations • Collect data for rendering	Generate presentation, particularly HTML
Role	Handle requests, perform computations, transfer control to JSP components	Render a response to the initial request

JSPs en tiempo de ejecución

Because JSP components are translated into servlets, JSP components and servlets share runtime behaviors:

- Life cycle and container management
- API and container services
- Client session access

Both can be entered on multiple threads concurrently and must be implemented accordingly.

Web context root y alias mapping

Servlets and JSP components are packaged into a web application.

- Static content, such as HTML or images, is included.
- A web application URI has the following form:
  `http://server:port/context_root/resource`
- The context root maps to a web application.
- An alias maps from a URL to a JSP or servlet.
  `http://www.mybank.com/bank/main`

The `main` resource could be specified:

- In the deployment descriptor using a URL pattern(s)
- In the servlet class using an annotation to specify a URL pattern(s)

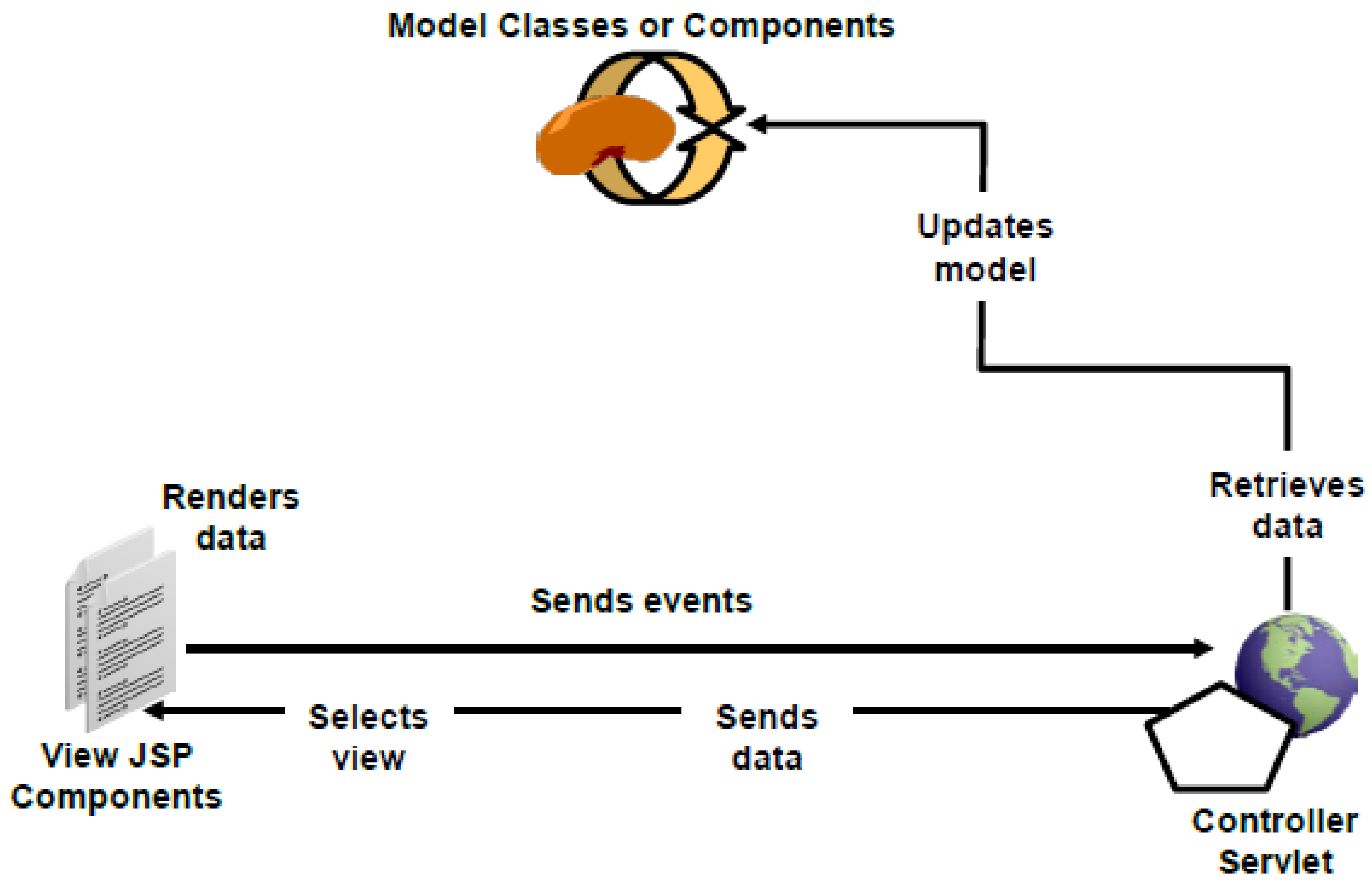
Manejo de sesiones en aplicaciones Web

The HTTP protocol is stateless. Conversational state might be stored on either the browser or the server:

	Browser	Server
Description	Simple and does not consume server resources	Must carry a session ID between the browser and server
Storage Availability	Limited	Less restricted

The Java EE model provides a simple mechanism for storing conversational state on the server.

Patrón de diseño Model View Controller (MVC)



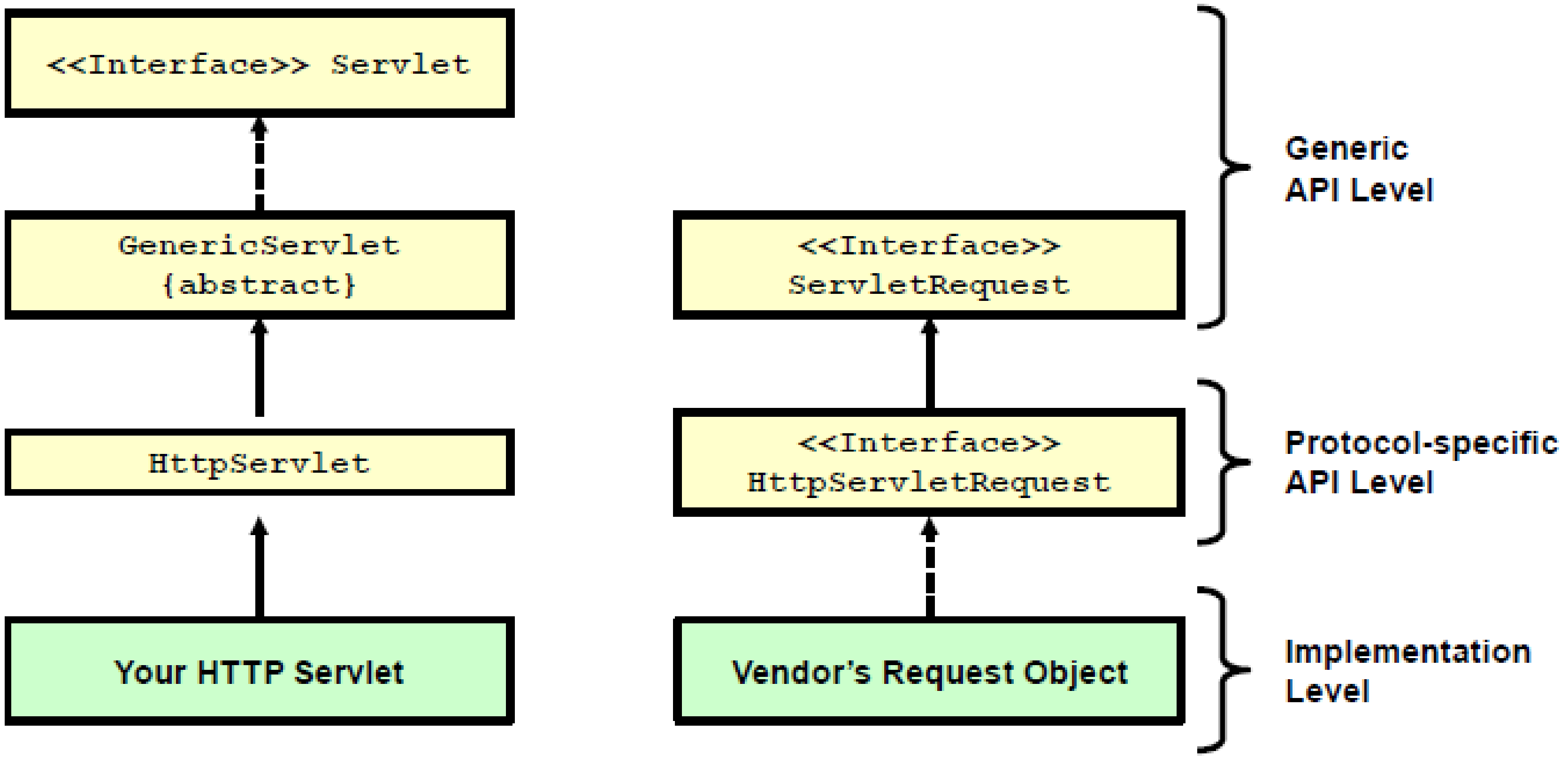
Aplicaciones Web con Java

emaravi@cjavaperu.com

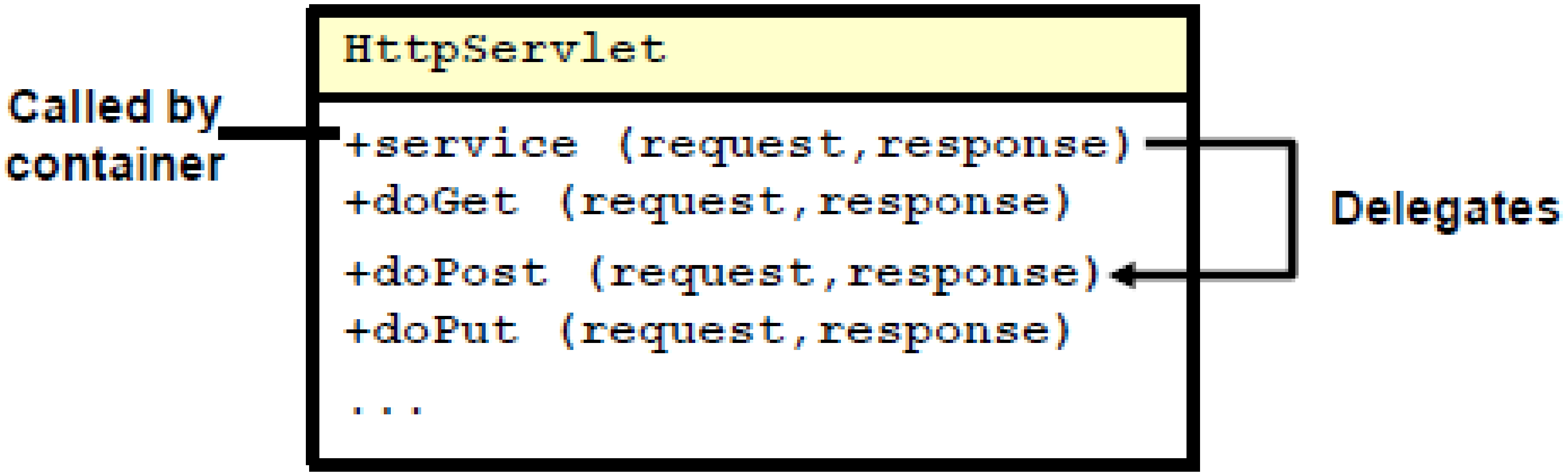
Java Servlets

emaravi@cjavaperu.com

El API Servlet


<<Interface>> Servlet
GenericServlet {abstract}
HttpServlet
Your HTTP Servlet
<<Interface>> ServletRequest
<<Interface>> HttpServletRequest
Vendor's Request Object
Generic API Level
Protocol-specific API Level
Implementation Level

La clase HttpServlet


Called by container
HttpServlet
+service (request,response)
+doGet (request,response)
+doPost (request,response)
+doPut (request,response)
...
Delegates

Ejemplo de Servlet

```java
package com.example;

import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;

@WebServlet("/example")
public class ExampleServlet extends HttpServlet {

    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response) {
        processRequest(request, response);
    }

    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse response) {
        processRequest(request, response);
    }

    public void processRequest(HttpServletRequest request, HttpServletResponse response) {
        // Process request and generate response
    }
}
```

Configuración de un Servlet

Without configuration, a servlet does not have an accessible URL. Beginning with Servlet spec 3.0 (Java EE 6), annotations are used to map URLs to servlets.

- Single-URL example:

```
@WebServlet("/myservlet")
public class MyHttpServlet extends HttpServlet{
//...
}
```

- Multiple-URL example:

```
@WebServlet(name="SomeName", urlPatterns={"/myservlet",
   "/foo", "/bar"})
public class MyHttpServlet extends HttpServlet{
//...
}
```

Deployment Descriptor de un WAR

To configure a servlet and many other aspects of a web application, with a deployment descriptor place a `web.xml` file in the `WEB-INF` directory.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://
java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
    <servlet>
        <servlet-name>SomeName</servlet-name>
        <servlet-class>package.MyHttpServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>SomeName</servlet-name>
        <url-pattern>/myservlet</url-pattern>
    </servlet-mapping>
</web-app>
```

Ciclo de vida de un Servlet

As outlined in the previous module, servlets are multithreaded. A single servlet instance is created for each `<servlet>` configured in the `web.xml` file. A servlet is instantiated sometime before its `service` method is called.

A servlet can have two styles of initialization methods that are called before the `service` method and two types of methods called before discarding the servlet.

```
public void init() {....}
@PostConstruct public void myInitMethod() {...}

public void destroy() {...}
@PreDestroy public void myDestroyMethod() {...}
```

La clase HttpServletRequest

Generic Methods	Purpose
`getParameter`	Gets form data elements
`getAttribute` and `setAttribute`	Gets and sets attributes, which are used for passing data between components
`getRequestDispatcher`	Gets a request dispatcher to transfer control to another component
HTTP-Specific Methods	**Purpose**
`getUserPrincipal`	The identity of the user, if authenticated
`getCookies`	Gets the identity of the user, if authenticated
`getSession`	Gets client session

La clase HttpServletResponse

Generic Methods	**Purpose**
`getOutputStream`, `getWriter`	Gets a stream or writer to send data to the browser
`setContentType`	Indicates the MIME type of response body
HTTP-Specific Methods	**Purpose**
`encodeURL`	Adds a session ID to a URL
`addCookie`	Sends a cookie to the browser
`sendError`	Sends an HTTP error code

Recibiendo datos de un formulario

```
public void processRequest (HttpServletRequest request,
HttpServletResponse response)
        throws IOException {
    response.setContentType ("text/plain");
    PrintWriter out = response.getWriter();
    String name = request.getParameter ("name");
    if (name == null || name.length() == 0)
        name = "anonymous";
    out.println ("Hello, " + name);
    out.close();
  }
```

Pasando el flujo a otro componente Web

There are two `getRequestDispatcher`("URI") methods that return a `RequestDispatcher` implementation.

The `ServletRequest` method that accepts relative paths or paths beginning with a "/"

```
RequestDispatcher requestDispatcher =
request.getRequestDispatcher("relativeURI");
```

The `ServletContext` method that must begin with a "/"

```
RequestDispatcher requestDispatcher
=    getServletContext().getRequestDispatcher
     ("/ServletName");
```

Pasando objetos a otro componente Web

The request object can carry data between components:

- In the calling component:

```
CustomerData customerData = // get customer data
request.setAttribute ("customerData",
customerData);
requestDispatcher.forward (request, response);
```

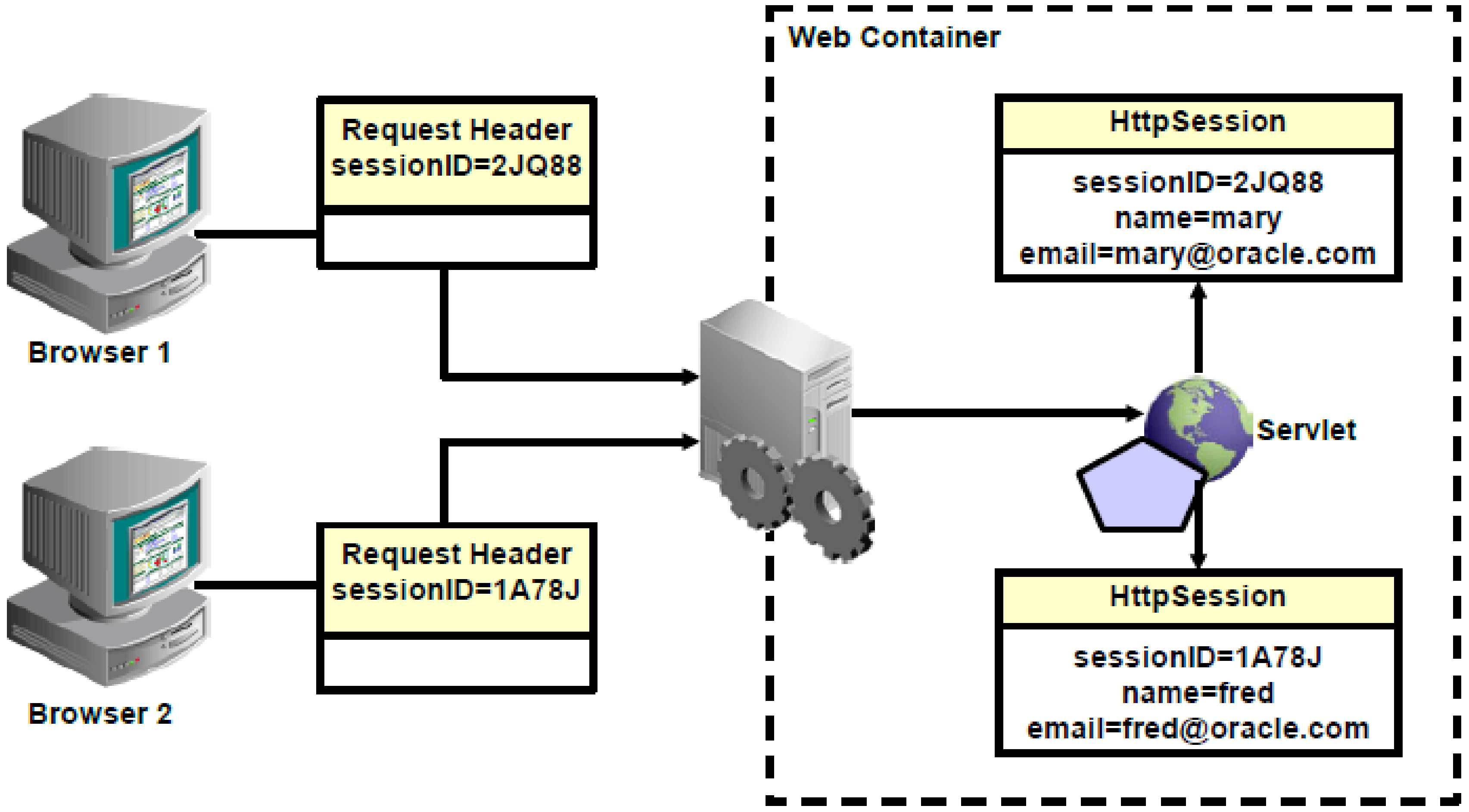
- If the target component is a servlet:

```
CustomerData customerData = (CustomerData)
request.getAttribute("customerData");
```

- If the target component is a JSP component:

```
<jsp:useBean id="customerData"
class="Bank.CustomerData" scope="request"/>
```

Manejo de sesiones en aplicaciones Web



La clase HttpSession

The `request.getSession` method always returns a session. The `HttpSession.isNew` method returns true in either of the following situations:

- The session is a new session with a new browser.
- The current browser session timed out before this request.

To close a session early, call its `invalidate` method:

```
if ("logout".equals(request.getParameter("action")) {
  session.invalidate();
}
```

Revisar sesiones nuevas

```java
// Get a session object for the current client, creating
// a new session if necessary
HttpSession session = request.getSession();

// If this is a new session, initialize it
if (session.isNew()) {
  // Initialize the session attributes
  // to their start-of-session values
  session.setAttribute ("account", new Account());
    // ... other initialization
  }

  // Get this client’s ‘account’ object
  Account account = (Account)session.getAttribute("account");
```

Java Servlets

emaravi@cjavaperu.com