Componentes Web

Componentes dinámicos en la plataforma Java EE.

Agenda

- ¿Qué es Servlet?
- Modelo Requerimiento-Respuesta
- Ciclo de vida
- Ejemplo en NetBeans

¿Qué es Servlet?

- Son objetos Java™ que extienden las funcionalidades de un servidor HTTP creando contenido dinámico.
- Basado en un modelo de programación requerimiento-respuesta.
- Permite ampliar las aplicaciones alojadas en el servidor web.

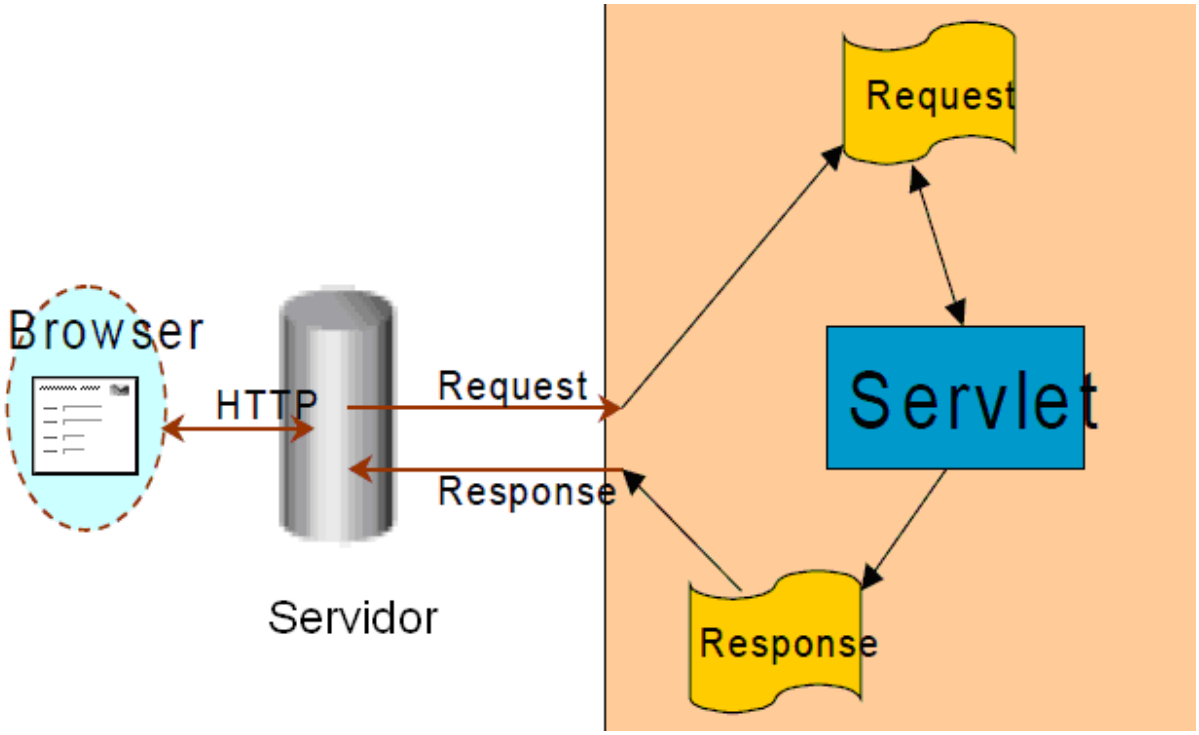
...Mi primer Servlet

```
@WebServlet(urlPatterns = {"/MiServlet"})
public class MiServlet extends HttpServlet {
   public void processRequest (Http… request, Http..
   response){
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            out.println("<title>Hello World!</title>");}

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

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

Modelo
Requerimiento -Respuesta



…Modelo RR

- Recibe el requerimiento del cliente (en forma de requerimiento HTTP)
- Extrae la información del objeto request
- Procesa la lógica del negocio (realiza accesos a la BD, invocando EJBs, web services etc)
- Crea y transmite la respuesta al cliente (en la forma de respuesta HTTP) o envía el objeto response a otro servlet o página.

...Modelo RR

Requerimiento:

Es la información que transmite el cliente al servidor, a través del encabezado HTTP.

Respuesta:

Es la información (texto-html o binario-imagen) que el servidor le transmite al cliente a través del encabezado HTTP.

HTTP

- Una petición HTTP contiene:
  - Encabezado
  - Un método HTTP:
    - Get: Los datos del formulario de entrada se pasan como parte del URL.
    - Post: Los datos del formulario de entrada se pasan dentro del cuerpo del mensaje.
  - Datos

https://www.rfc-editor.org/rfc/rfc9110.html#methods

…HTTP

- Métodos HTTP mas comunes:
  - Solicitud **GET**:
    - La información completa del usuario es agregada al URL como una consulta string.
    - Puede transmitirse solamente una cantidad de datos limitada como parámetros de la consulta.

    ..../Ejemplo/MiServlet?param1=Susana&param2=Chavez&param3=Susy&Enviar=Enviar
  - Solicitud **POST**:
    - La información completa del usuario es transmitida como un dato (no se agrega al URL)
    - Puede transmitir cualquier cantidad de datos.

	GET	POST
Datos	Los datos del cliente se agregan al URL	Se transmiten separados del URL
Almacenamiento en el history del browser	Como los datos son agregados al URL deben ser almacenados en el historial	No se almacenan en el historial del browser
Bookmark	La URL con los datos del cliente puede ser marcada. De esta manera, más tarde sin llenar el formulario HTML, se pueden enviar los mismos datos al servidor.	No es posible bookmark
Tamaño	Limitado a 2048 caracteres (dependiendo del browser)	Sin limites
Hacking	Fácil de piratear los datos ya que los datos se almacenan en el historial del navegador	Difícil
Tipo de datos	Solo se pueden enviar datos ASCII	Se puede enviar cualquier tipo de datos, incluidos los datos binarios
Privacidad	Los datos no son secretos ya que otras personas pueden ver los datos en el historial del navegador	Son privados
Cuando usar	Cuando los datos enviados no son secretos. Como son las contraseñas.	Conviene cuando los datos son críticos y confidenciales
Performance	Relativamente más rápido	Se creará un cuerpo de mensaje por separado
Por defecto	Si no se menciona, GET se supone como predeterminado	Se debe mencionar explícitamente

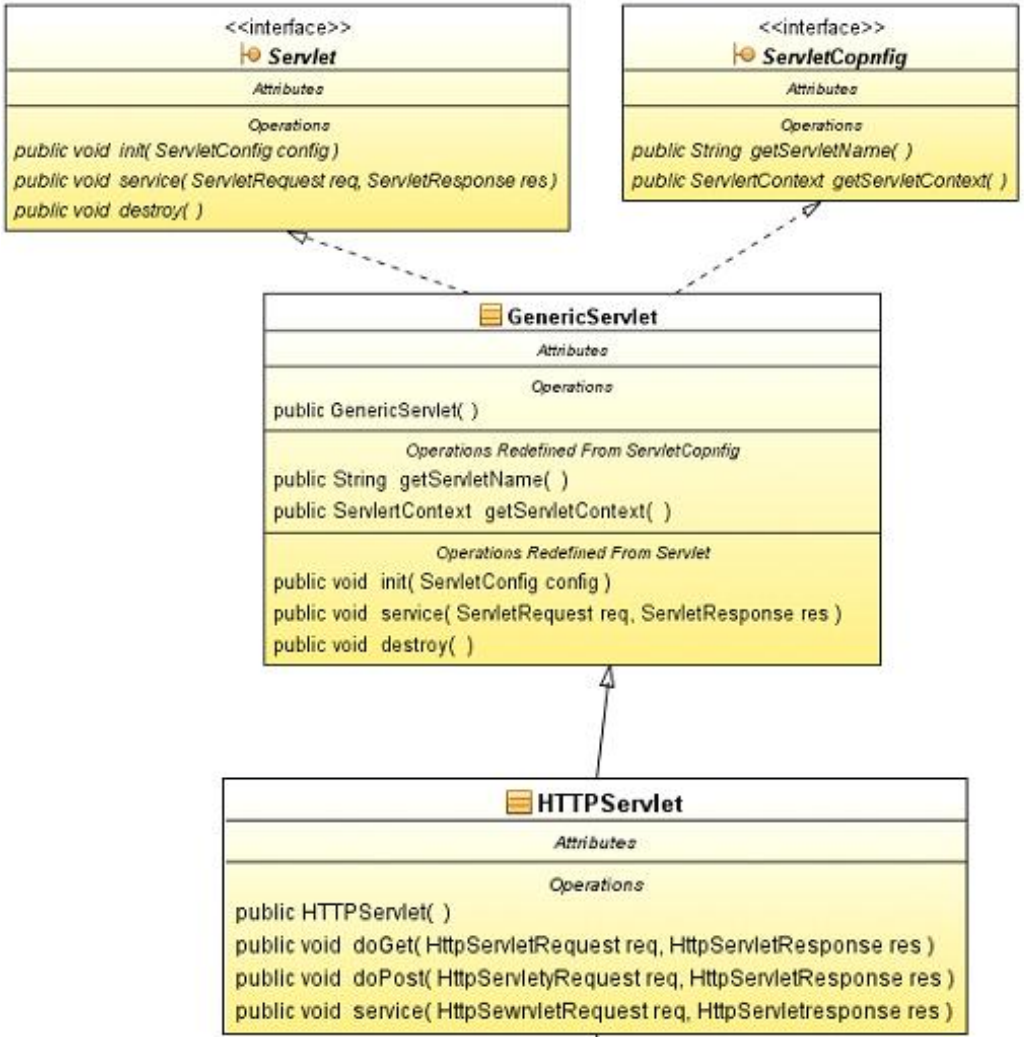
Cliclo de vida

…ciclo de vida

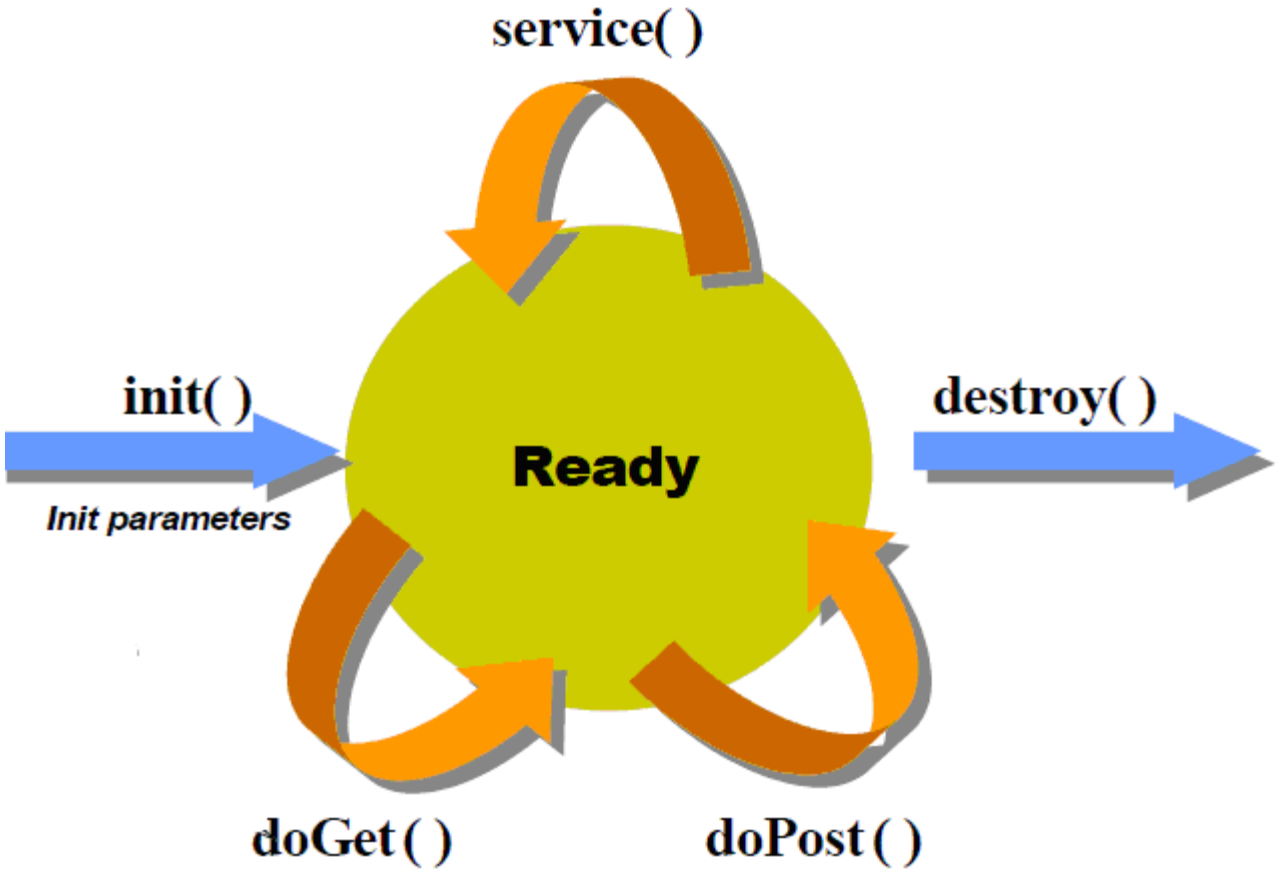
El ciclo de vida de un servlet esta controlado por el contenedor para:

1. Si no existe una instancia del servlet:
   a) Carga la clase servlet.
   b) Crear una instancia de la clase servlet.
   c) Inicializa el objeto llamando al método init.
2. Invoca el método service, pasando los objetos request y response.
3. Al cerrar la aplicación, el contenedor finaliza el servlet llamando al método destroy.

…ciclo de vida Jerarquía de clases - API -



…ciclo de vida



Métodos del ciclo de vida

El contenedor controla el ciclo de vida del servlet:

- init(): Se invoca una sola vez cuando el servlet es instanciado.
  - Se realizan las conecciones a la BD.
- destroy(): Se invoca antes de borrar el servlet
  - Cierra las conecciones a la BD.

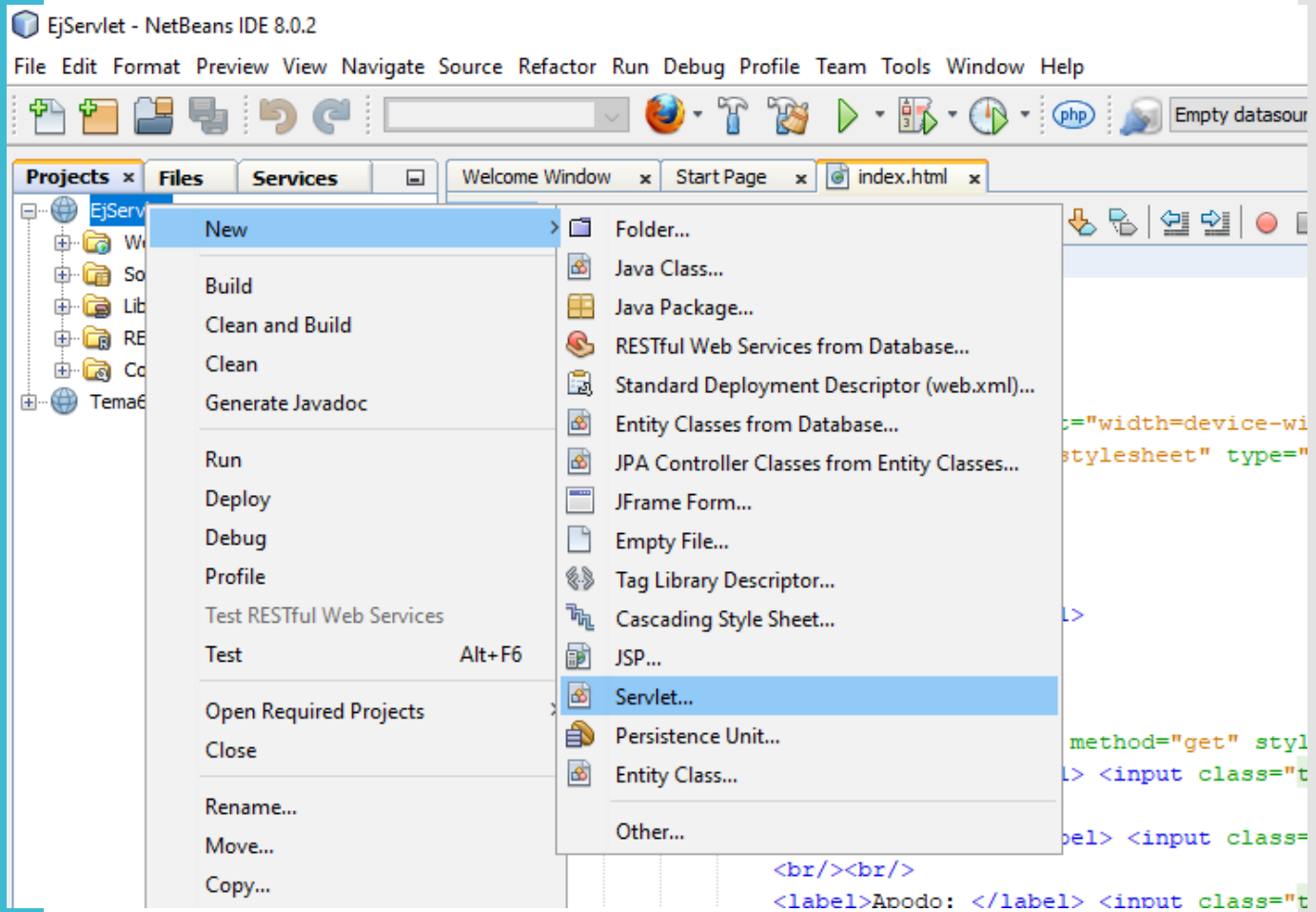
Métodos del ciclo de vida

- service():
  - Método abstracto en javax.servlet.GenericServlet e implementado en javax.servlet.http.HttpServlet
  - Invoca a doGet(), doPost()
  - No se debe sobrescribir!!!
- doGet(), doPost():
  - Definidos en javax.servlet.http.HttpServlet
  - Maneja las solicitudes GET, POST de HTTP
  - Se deben implementar para que tengan el comportamiento deseado.

Servlet en NetBeans

Un ejemplo

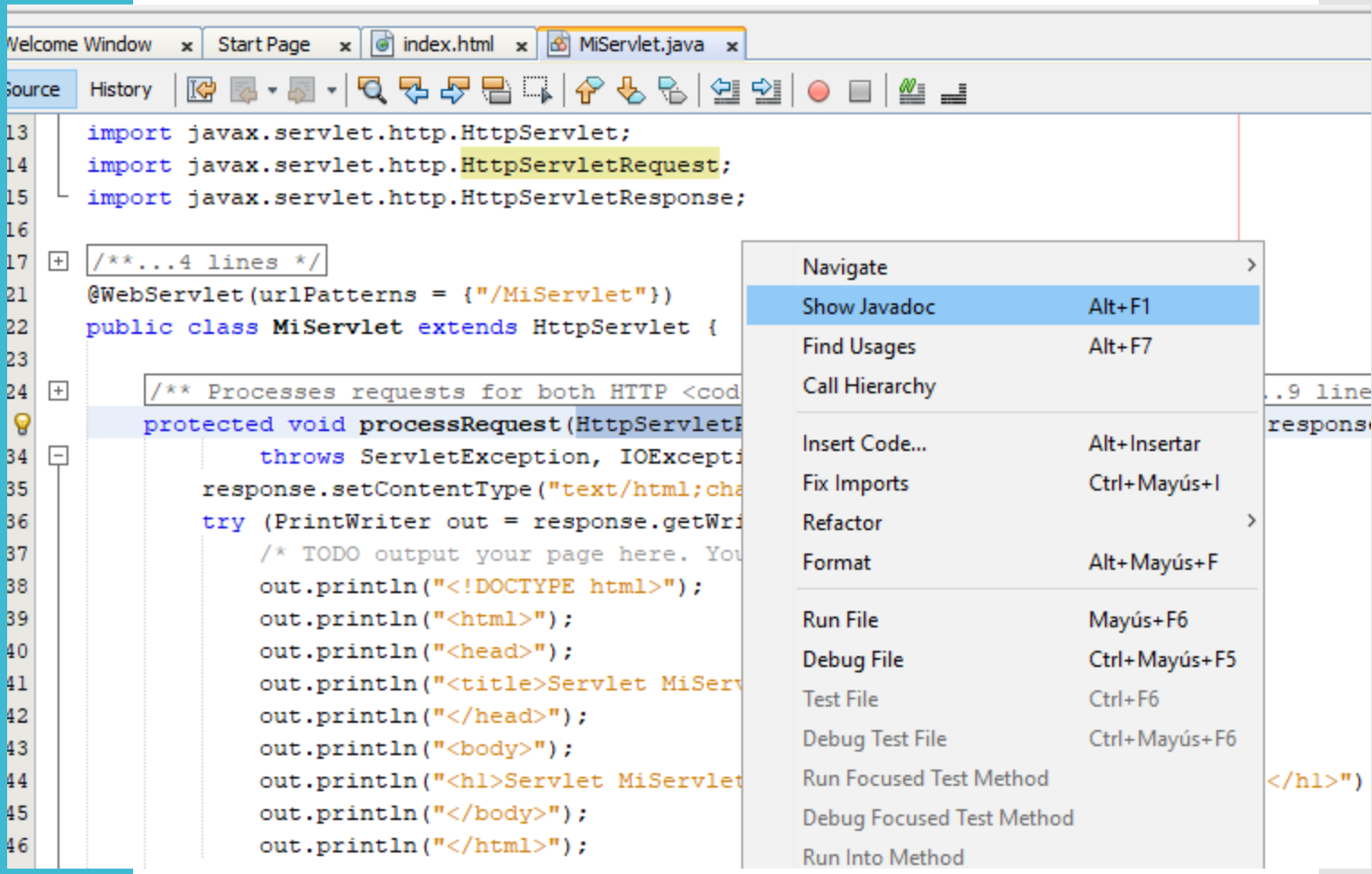
IDE NetBeans



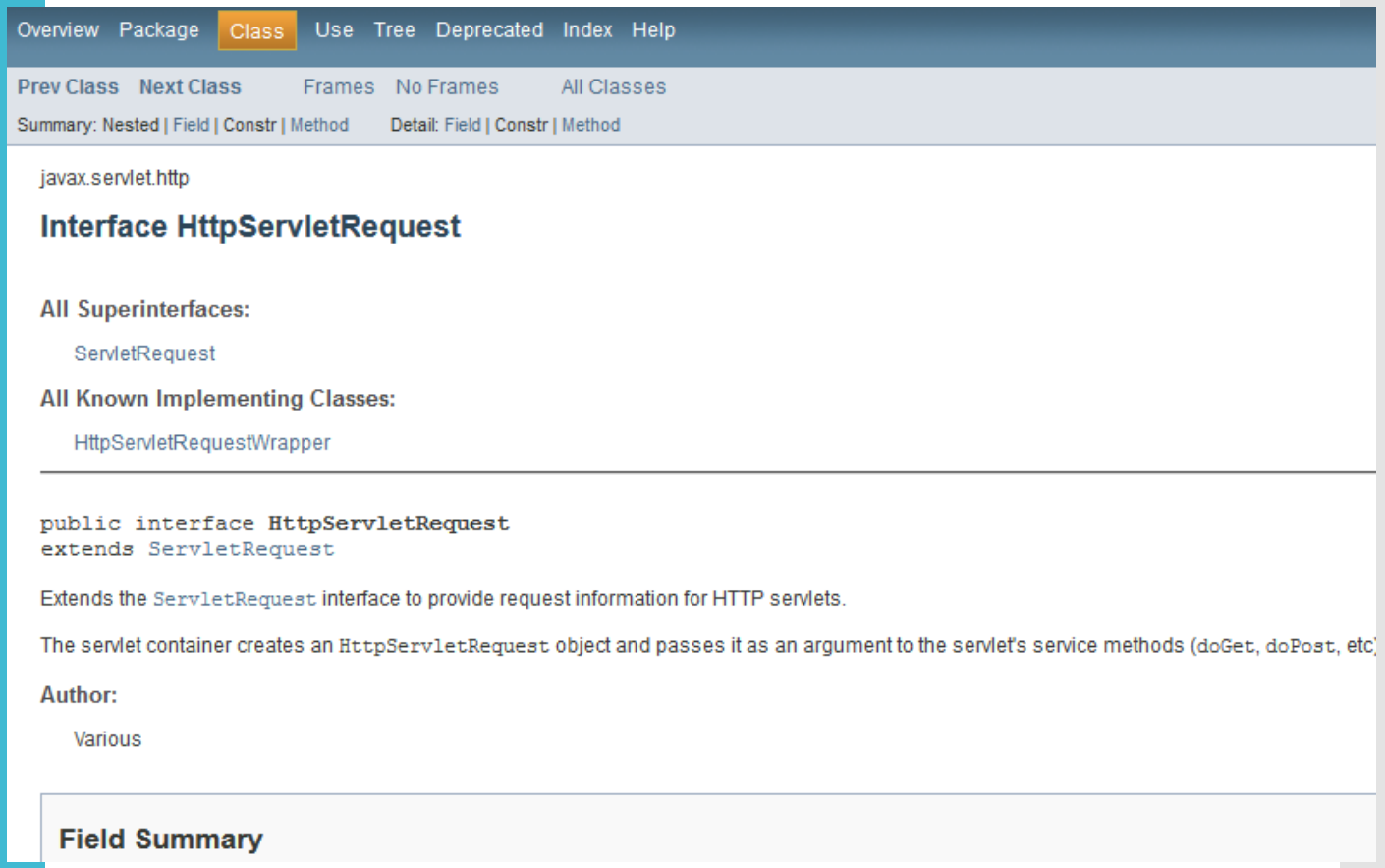
IDE NetBeans

```
@WebServlet(urlPatterns = {"/MiServlet"})
public class MiServlet extends HttpServlet {
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      try {
         out.println("<html>");
         out.println("<head>");
         out.println("<title>Servlet NewServlet</title>");
         out.println("</head>");
         out.println("<body>");
         out.println("<h1>Servlet NewServlet at " + request.getContextPath () + "</h1>");
         out.println("</body>");
         out.println("</html>");
      } finally { out.close();  }  }
@Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      processRequest(request, response);
   }
@Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
      processRequest(request, response);  }
```

IDE NetBeans



API Java EE



Overview Package Class Use Tree Deprecated Index Help

Prev Class Next Class Frames No Frames All Classes

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

javax.servlet.http

Interface HttpServletRequest

All Superinterfaces:

ServletRequest

All Known Implementing Classes:

HttpServletRequestWrapper

```
public interface HttpServletRequest
extends ServletRequest
```

Extends the `ServletRequest` interface to provide request information for HTTP servlets.

The servlet container creates an `HttpServletRequest` object and passes it as an argument to the servlet's service methods (`doGet`, `doPost`, etc)

Author:

Various

Field Summary

Ejemplo basado en GET

Source History

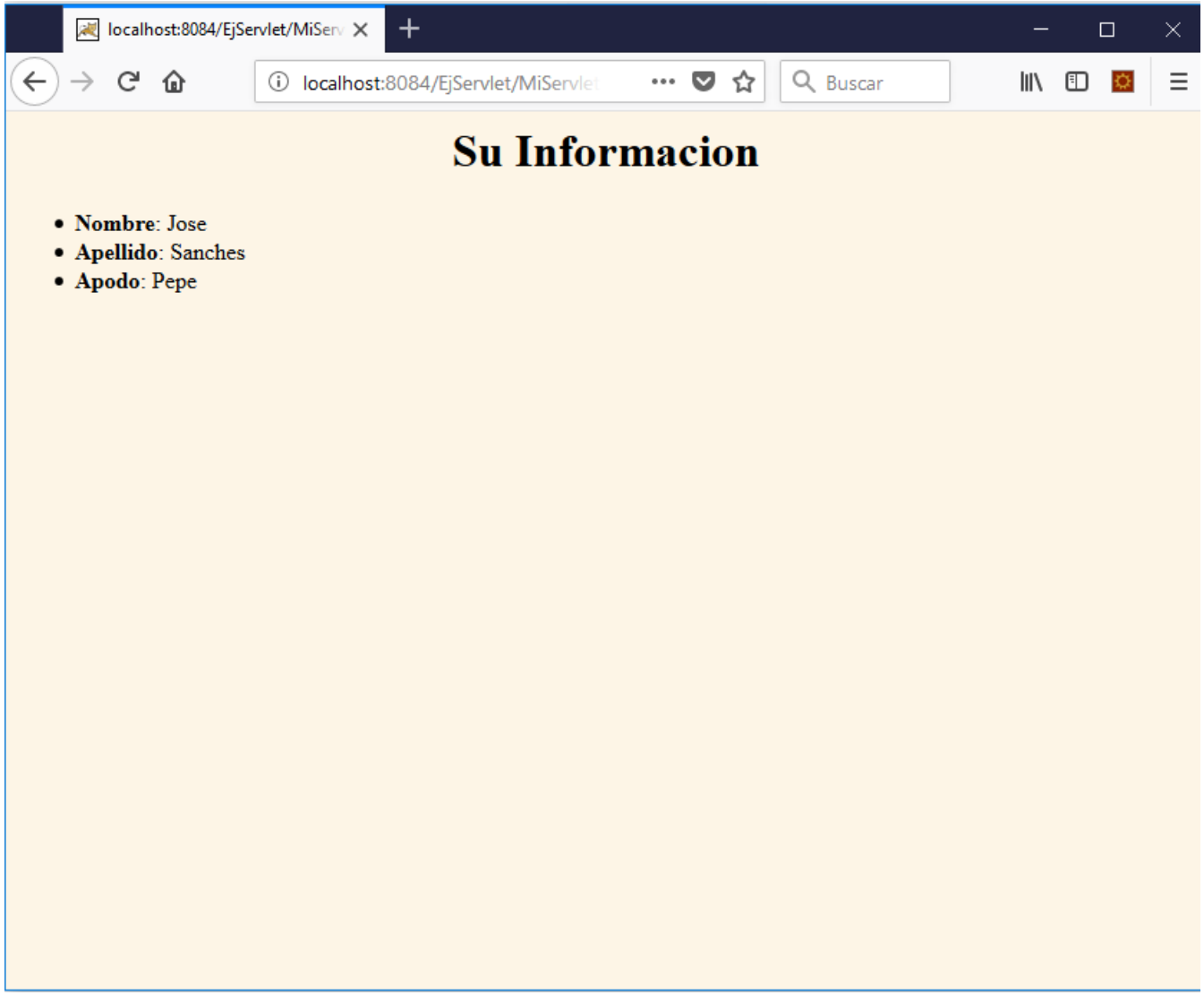
```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Tema6</title>
5          <meta charset="UTF-8">
6          <meta name="viewport" content="width=device-width, initial-scale=1.0">
7          <link href="tema6.css" rel="stylesheet" type="text/css"/>
8      </head>
9      <body>
10         <div id="capaMadre">
11         <div id="cabecera">
12             <h1>Ingrese sus Datos</h1>
13         </div>
14         <div id="cuerpo">
15
16             <form action="MiServlet" method="get" style="margin-top: 50px">
17                 <label>Nombre:</label> <input class="texto" name="nombre" type="text"/>
18                 <br/><br/>
19                 <label>Apellido:</label> <input class="texto" name="apellido" type="text"
20                 <br/><br/>
21                 <label>Apodo: </label> <input class="texto" name="apodo" type="text"/>
22                 <br/><br/><br/>
23                 <input class="texto" name="boton" type="submit"/>
24             </form>
25         </div>
26         <div id="pie">Tecnologías Web</div>
27         </div>
```


EjServlet
localhost:8084/EjServlet/
Buscar
Ingrese sus Datos
Nombre:
Jose
Apellido:
Sanches
Apodo:
Pepe
Enviar consulta
Tecnologías Web

http://localhost:8084/EjServlet/MiServlet?nombre=Jose&apellido=Sanches&apodo=Pepe&boton=Enviar+consulta

```
public class MiServlet extends HttpServlet {
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)  throws ServletException,
   IOException {

     response.setContentType("text/html;charset=UTF-8");
     String nom = request.getParameter(" nombre");  String titulo = "Su Informacion";
     if (nom == null || nom.isBlank()) { response.sendError(HttpServletResponse.SC_BAD_REQUEST); return; }
     try {PrintWriter out = response.getWriter();
         out.println("<html>" + "<body bgcolor=\"#fdf5e6\">\n" +
             "<h1 align=center>" + titulo + "</h1>\n" +"<ul>\n" +
             " <li><b>Nombre</b>: " + nom + "\n" +
             " <li><b>Apellido</b>: " + request.getParameter("apellido") + "\n" +
             " <li><b>Apodo</b>: "    + request.getParameter("apodo") + "\n" +
             "</ul>\n" + "</body></html>");
     } finally {
   } }
   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
     processRequest(request, response);   }
```


localhost:8084/EjServlet/MiServ
localhost:8084/EjServlet/MiServlet
Buscar
Su Informacion
Nombre: Jose
Apellido: Sanches
Apodo: Pepe

Para obtener inf. del cliente

- String request.getRemoteAddr()
  - Obtengo la informacion IP del cliente
- String request.getRemoteHost()
  - Obtengo el nombre host del cliente

Para obtener inf del servidor

- String request.getServerName()
  - Ej.: “localhost"
- int request.getServerPort()
  - Ej.: Numero de puerto "8080"

Recursos

- API documents
- Servlet
- Tutorial Java EE5 (cap. 4)
- Tutorial Java EE6 (cap. 15)

…. manos a la obra

- Retomar el Proyecto de la clase anterior.
- Agregar Servlet que genere un JSON con la respuesta.
- Recursos:
  - Agregar al entorno la libreria GSON
  - Consultar manejo de MAP