Asignatura

Acceso a datos

UNIVERSAE
Instituto Superior de FP

Asignatura

Acceso a datos

UNIDAD 8

Componentes para el acceso a datos

UNIVERSAE
Instituto Superior de FP

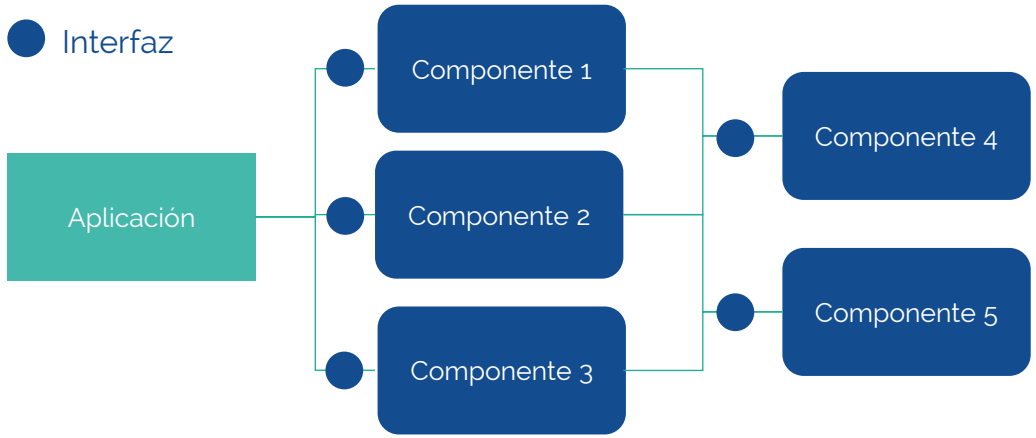
Componentes de software

Arquitectura basada en componentes

- Descompone la aplicación en componentes funcionales o lógicos
- Debe de haber interfaces para la comunicación.

¿Que es un componente?

- Bloque o parte de código que cumple una funcionalidad concreta
- Establece una forma de interactuar a partir de su interfaz
- Cada componente debe ser:
    - Independiente
    - Reutilizable
    - Extensible
    - Encapsulado
- Se suele utilizar en sistemas electrónicos y software

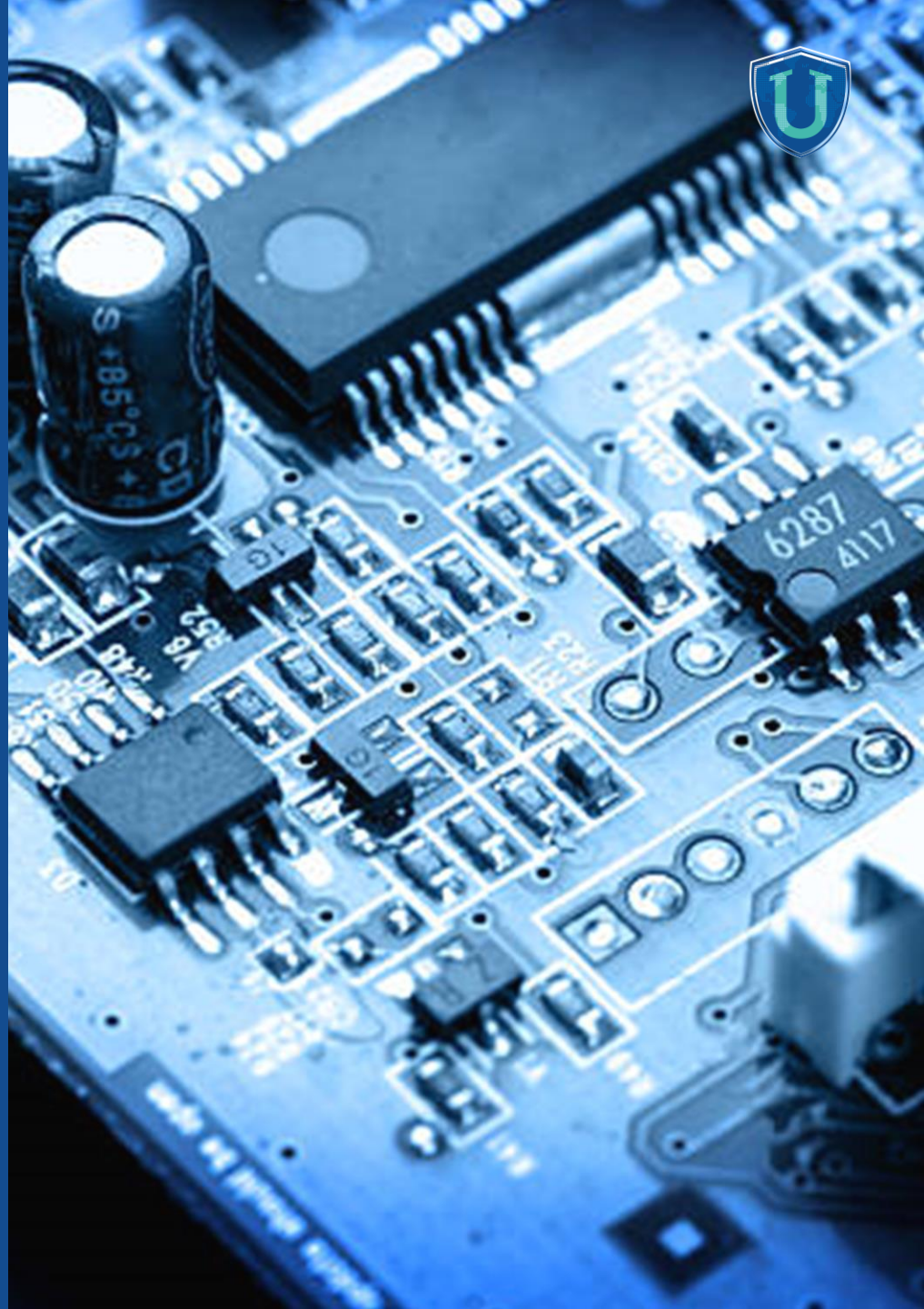


Modelo de componentes

- Infraestructura de software:
  - Con servicios básicos
  - Permita desplegar components
  - Herramientas de interacción
- Los primeros modelos de componentes fueron visuales. **OLE y COM**
- Posteriormente se usaron modelo de componentes no visuales. **DAO**

Modelo de componentes

- Microsoft
  - ActiveX
  - COM+
  - DCOM
- Java
  - JavaBeans
  - EjB (Enterprise JavaBeans)
- OMG
  - CORBA



Plataforma Java

Java SE

- Standard Edition
- Bibliotecas de uso general
- Usa modelo de componentes basada en **JavaBeans**

Java EE

- Enterprise Edition
- J2EE o Jakarta EE
- Especificación de java para aplicaciones empresariales.
- Se añaden bibliotecas para dar soporte al despliegue de software
- Usa modelo de components basado en **EjB**

JavaBeans

¿Qué son?

- Definen objetos de acceso a datos
- Son clases con algunos requisitos concretos
- Un JavaBeans es un POJO especifico

Requisitos

- Disponer de un constructor vacío
- Opcional constructores parametrizados
- Uso de la interfaz Serializable
- Propiedades privadas
- Métodos get y set
- La nomenclatura de los get y set debe ser correcta respecto al nombre de la propiedad.

Mecanismos

- Persistencia
- Reflexión e introspección
- Personalización

POJO	JavaBean
No tienen restricciones. Solo las del lenguaje Java	Es un POJO con restricciones
No proporciona control sobre sus miembros	Tiene control sobre sus miembros
Puede implementar Serializable	Es obligatorio implementar serializable
Los campos pueden ser accedidos por sus nombres	Los campos solo pueden ser accedidos por sus métodos get y set
Los campos pueden tener cualquier visibilidad	Es obligatorio que los campos sean privados
Es opcional que haya un constructor sin argumentos	Es obligatorio que haya un constructor sin argumentos
Se usa cuando no se quiere limitar al usuario a su acceso completo	Se usa cuando se quiere limitar al usuario el acceso

```java
class Bean implements Serializable {

    private static final long serialVersionUID = -1964338843566745355L;

    // Propiedad privada
    private Integer property;

    // Constructor sin argumentos
    public Bean() {

    }

    // Método set
    public void setProperty(Integer property) {
        if (property == 0) {
            // if property is 0 return
            return;
        }
        this.property = property;
    }

    // Método get
    public int getProperty() {
        return property;
    }

}
```

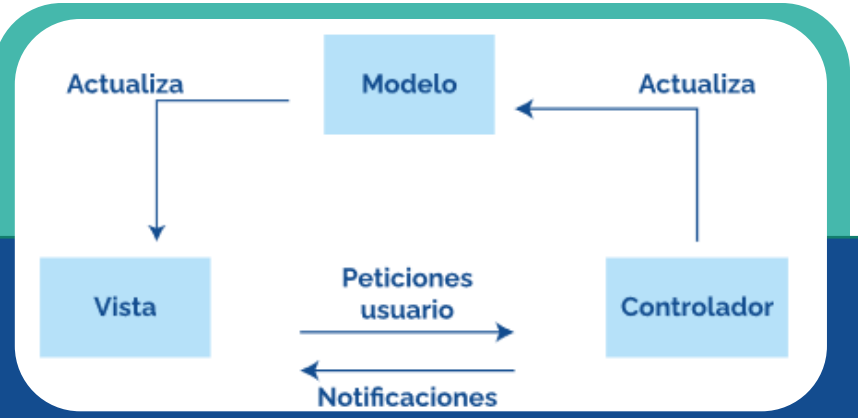
El modelo MVC

MVC (Modelo, vista, controlador)

- Es un patrón de diseño y desarrollo de aplicaciones.
- Divide en 3 capas las aplicación, **Modelo, Vista y Controlador**
- Permite reutilizar código y funcionalidad
- Desarrollar de forma independiente cada capa.

Capas

- **Modelo:** Se encarga de manejar los datos y el acceso a la base de datos. **JavaBeans** o **EjB**
- **Vista:** Es la interfaz. Lo que ve el usuario. **JSP**
- **Controlador:** Encargado de dirigir las peticiones del usuario con el modelo. **Servlets**



JSP

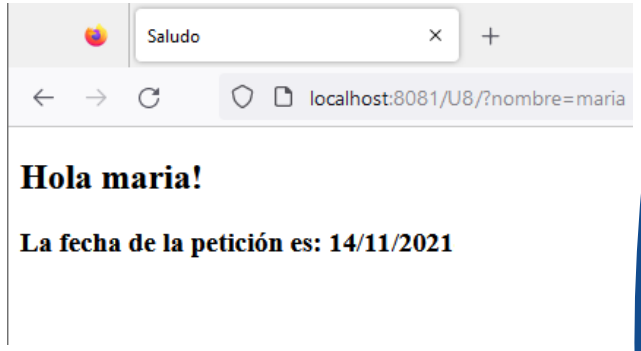
JavaServer Pages

- Ficheros java para desarrollar páginas web dinámicas.
- Son útiles para diseñar interfaces
- Su funcionamiento es similar a PHP o ASP.

Contenido

- Etiquetas HTML o XML
- Directivas JSP
- Código del lenguaje java

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@ page import="java.util.Date, java.text.SimpleDateFormat" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
        <title>Saludo</title>
    </head>
    <body>
        <%
            String nombre = request.getParameter("nombre");
        %>

        <h2>Hola <%= nombre %>!</h2>
        <%
            SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
            Date fecha = new Date();
        %>
        <h3> La fecha de la petición es: <% out.println(sdf.format(fecha)); %> </h3>
    </body>
</html>
```



JSP. Composición

Directivas

- <% atributo="valor" %>

Atributo	Descripción
import	Importa clases y paquetes para usar dentro del fichero JSP
session	Permite indicar si se usan datos de la sesión
contentType	Especifica el tipo de contenido del fichero, por defecto es "text/html"
errorPage	Indica la página de error a la que dirigirse en cado de una excepción
isErrorPage	Determina si la página gestiona excepciones

Scriptlets

- <% código java %>

Variables implícitas

- Ámbitos

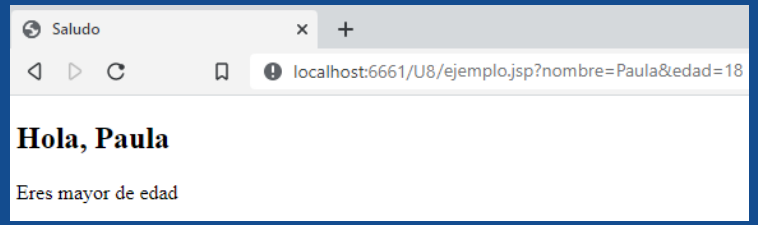
Página (page) < Petición (request) < Session (sesión) < Aplicación (application)

Variable implicita	Clase
pageContext	javax.servlet.jsp.PageContext
request	javax.servlet.http.HttpServletRequest
response	javax.servlet.http.HttpServletResponse
session	javax.servlet.http.HttpSession
config	javax.servlet.ServletConfig
application	javax.servlet.ServletContext
page	java.lang.Object
exception	java.lang.Exception

Referencias a variables en Java

- <%= variable %>
- <%out.print(variable)%>

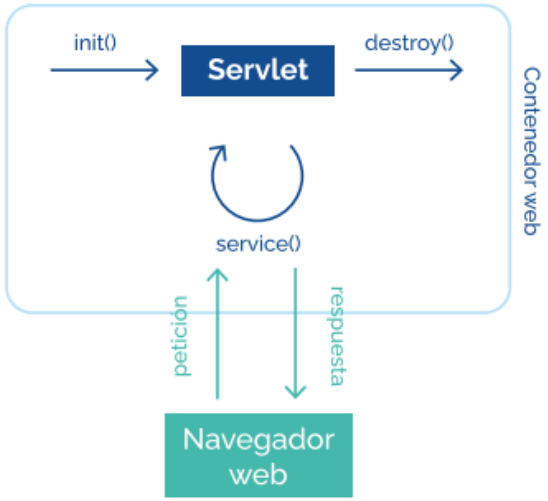
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@ page import="java.util.Date, java.text.SimpleDateFormat" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
        <title>Saludo</title>
    </head>
    <body>
        <!-- Este comentario lo vera el cliente desde el codigo fuente -->
        <%-- Este comentario no se mandara al cliente y solo se vera desde el servidor --%>
        <% String nombre = request.getParameter("nombre");
           int edad = (request.getParameter("edad")!=null) ? Integer.parseInt(request.getParameter("edad")) : 0;
        %>

        <h2>Hola, <%= nombre %></h2>

        <% if (edad >= 18) { %>
            <p> Eres mayor de edad </p>
        <%  } else { %>
            <p> Eres menor de edad </p>
        <%  } %>
    </body>
</html>
```



Controlador. Servlets

¿Qué son?

- Son componentes en java
- Utilizan el protocolo HTTP
- Gestionan peticiones y respuestas en HTML o JSP
- Se ejecutan en un servidor web contenedor de servlets



Características

➢ Rendimiento.
Un servlet será un proceso y cada petición un hilo
➢ Portabilidad
Se puede llevar de un servidor a otro
➢ Rápido desarrollo
Es una estructura modular, y de fácil desarrollo.
➢ Robustez
Las proporcionadas por JAVA
➢ Amplio soporte
Se utiliza en diferentes compañías y comunidades

Servlet. Estructura

- Uso de los paquetes
  - *javax.servlet*
  - *jakarta.servlet*
- Deben de heredar de :
  - *GenericServlet*
  - *HttpServlet*

Métodos

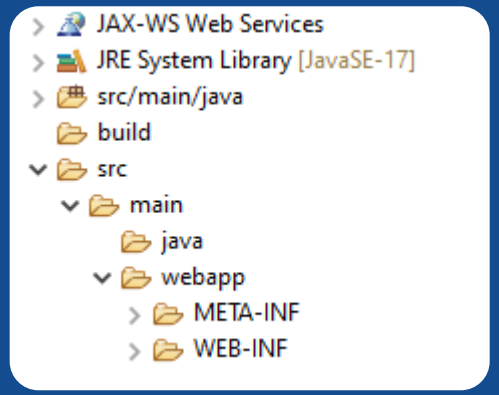
- doGet – Peticiónes GET
- doPost – Peticiones POST
- doPut – Peticiones PUT
- doDelete – Peticiones DELETE
- init - Inicializar
- Destroy - Liberar
- getServletInfo – Información del servlet

```java
import java.io.IOException;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class Servlet extends HttpServlet {
    private static final long serialVersionUID = -5301531031557762439L;

    public void init() throws ServletException {
        // Inicialización de recursos
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Acciones para una petición GET
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Acciones para una petición POST
    }

    public void destroy() {
        // Liberacion de recursos
    }
}
```

Estructura y recursos de una aplicación web java

Estructura

- Archivos WAR
  - META-INF
    - Define el contexto de la aplicación.
    - Contiene el fichero context.xml
  - WEB-INF
    - Define los componentes de configuración
    - Contiene el **descriptor de despliegue** (web.xml)
    - WEB-INF/clases/ - Las clases java y servlets
    - WEB-INF/lib/ - Las librerías que necesita el proyecto




```xml
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
         http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">
    <servlet>
        <servlet-name>saludar</servlet-name>
        <servlet-class>com.servlet.SaludarServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>saludar</servlet-name>
        <url-pattern>/saludo</url-pattern>
    </servlet-mapping>

    <session-config>
        <session-timeout>30</session-timeout>
    </session-config>

    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>

</web-app>
```

Recursos

- Servlets
- Páginas jsp
- Ficheros HTML
- Ficheros CSS
- Ficheros .java
- Ficheros de imágenes, sonido y videos

Servidor de aplicaciones. Tomcat

Servidor de aplicaciones

- Gestiona aplicaciones. Despliegue y puesta en marcha.
- Da soporte de seguridad, transacciones, sesiones, logs, etc.
- WildFly, GlassFish, JOnAs, **Tomcat**, Weblogic

Apache-Tomcat

- Pertenece a Apache
- Empleado para JAVA
- Contenedor de servlets y páginas JSP
- Gratuito.

¿Es lo mismo un servidor web que de aplicaciones?

- NO. Se puede considerar un servidor web como una parte concreta de un servidor de aplicaciones
- El servidor web solo puede servir contenido HTTP y por defecto solo contenido estático. Se puede modificar y servir contenido dinámico si se usa un modulo aparte, PHP, Perl, etc.
- El servidor de aplicaciones funciona con el protocolo HTTP, RMI/RPC, ..

Instalación de Apache Tomcat

Linux

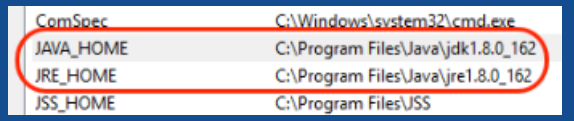
- Instalar JAVA JDK
  apt install default-jre default-jdk
- Verificar que es esta instalado correctamente
  java –versión
- Descargar Tomcat. https://tomcat.apache.org/download-10.cgi
- Descomprimir en /opt/tomcat
  tar xzf apache-tomcat-XXXX.tag.gz –C /opt/tomcat
- Añadir variables de entorno
  echo 'export JAVA_HOME="/opt/jdk/jdk/jdkX.X.X.X/jre"'>>/etc/profile.d/tomcatXX.sh"
  echo 'export JRE_HOME="/opt/jdk/jdk/jdk1.8.0_261/jre"'>>/etc/profile.d/tomcat9.sh"
- Arranchar tomcat
  /opt/tomcat/apache-tomcat-XXXX/bin/startup.sh
- Verificar que podemos conectar.
  Acceder con el navegador a http://localhost:8080

Windows

- Instalar JAVA JDK

https://www.oracle.com/java/technologies/downloads/

- Añadir las variables de entorno
  JAVA_HOME="Ruta de la instalación de java JDK"
  JRE_HOME="Ruta de la instalación JRE"



- Verificar que es esta instalado correctamente
  java –versión
- Descargar Tomcat. https://tomcat.apache.org/download-10.cgi
- Descomprimir la descarga en la ruta C:\ u otra significativa.
- Arrancar tomcat
  RUTA/bin/startup.bat
- Verificar que podemos conectar.
  Acceder con el navegador a http://localhost:8080

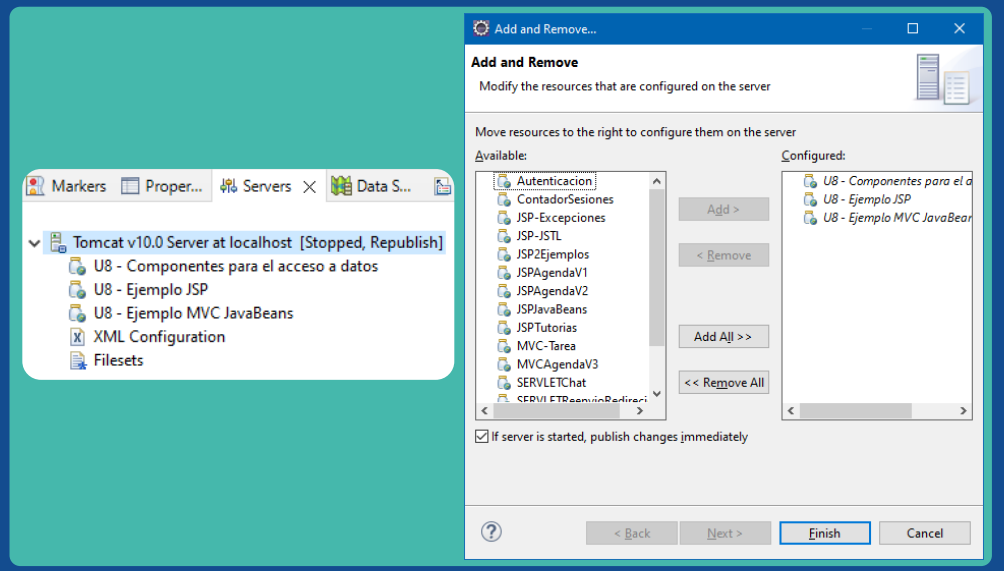
Integrar Tomcat con Eclipse

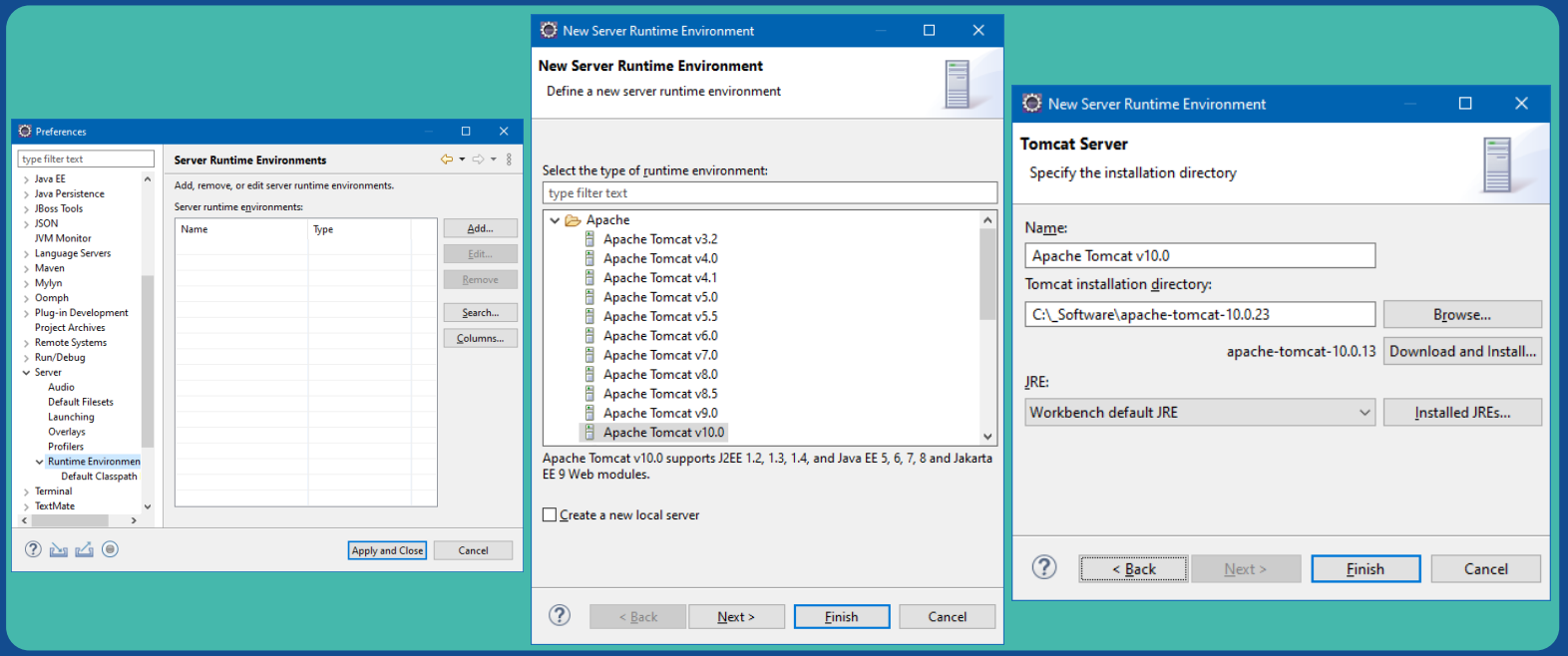
Proceso para añadir el servidor a eclipse

- En eclipse ir a *Windows/Preferences/Server/Runtime Environment*
- Añadir el servidor Apache e indicar la ruta donde esta la instalación
- Una vez finalizado. Aparece el servidor en la pestaña **Servers** de la pantalla principal. Si no apareciera la pestaña **Servers** *Windows/Show view/Server*

Vincular proyectos con el servidor

- Botón secundario sobre el servidor
- Add and Remove





Enterprise JavaBeans

- Similar a los JavaBeans.
- Aparece para la Plataforma JavaEE
- No todos los servidores soportan EjB.
- GlassFish.

Caracteristicas

- Uso API JPA para persistencia de objetos y JTA para transacciones.
- Gestión de transacciones.
- Control de concurrencia.
- Mecanismos CDI. Uso de anotaciones. *@EJB @Stateless*
- Sistema de mensajes JMS *(Java Message Service).*

Tipos

➢ **Entidad**

Representan el modelo de datos que están en la base de datos.

➢ **Sesión**

Procesos o acciones que gestionan la forma de comunicarse con el servidor.

➢ **Dirigidos por mensajes**

Representan objetos de servicios de mensajes JMS.

Resumen

1. Componentes de software
2. Modelo de componentes
3. Plataforma Java
4. JavaBeans
5. El modelo MVC
6. JSP
7. JSP. Composición
8. Servlets
9. Servlets. Estructura
10. Estructura y recursos de una aplicación web java
11. Servidor de aplicaciones. Tomcat
12. Instalación de Apache Tomcat
13. Integrar Tomcat con Eclipse
14. Aplicación web en java
15. Enterprise JavaBeans

UNIVERSAE

CHANGE YOUR WAY