J2EE

Visión Arquitectónica

Jose A. Espinosa

ÍNDICE

- ¿Qué es arquitectura?
- Plataforma Java
- ¿Qué es J2EE?
- APIs J2EE
- Contenedores J2EE
- Internacionalización
- Otras “tecnologías”
- Caso de estudio

Arquitectura

- La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad. *(Philippe Kruhten)*
- Una **Arquitectura Software**, también denominada *Arquitectura lógica*, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.
- La arquitectura software establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema de información.

El arquitecto

- ¿Qué hace un arquitecto software?

Un desarrollador se preocupa de qué hará la aplicación cuando un usuario apriete un botón, un arquitecto lo hace sobre qué pasará cuando lo aprietan miles.

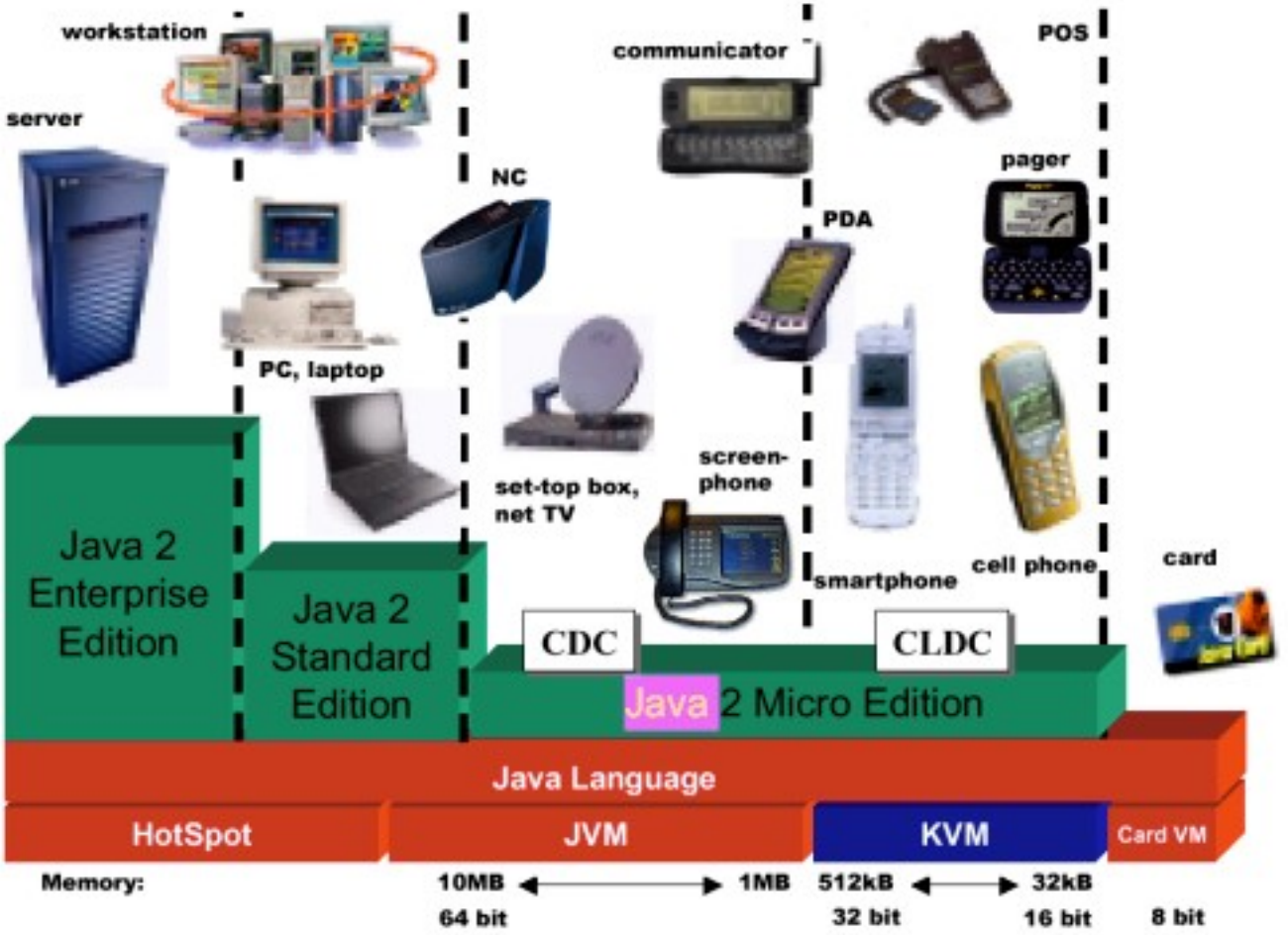
El arquitecto debe...

- Liderar el desarrollo para asegurar que se siga la arquitectura
- Tomar todas las decisiones que afecten a la arquitectura del sistema
- Comunicarse con desarrolladores y diseñadores constantemente
- Vigilar activamente el cumplimiento de los requisitos no funcionales

Requisitos de nivel de servicio

- Rendimiento
- Escalabilidad
- Fiabilidad
- Disponibilidad
- Ampliabilidad
- Mantenibilidad
- Gestionabilidad
- Seguridad

Plataforma Java



J2EE

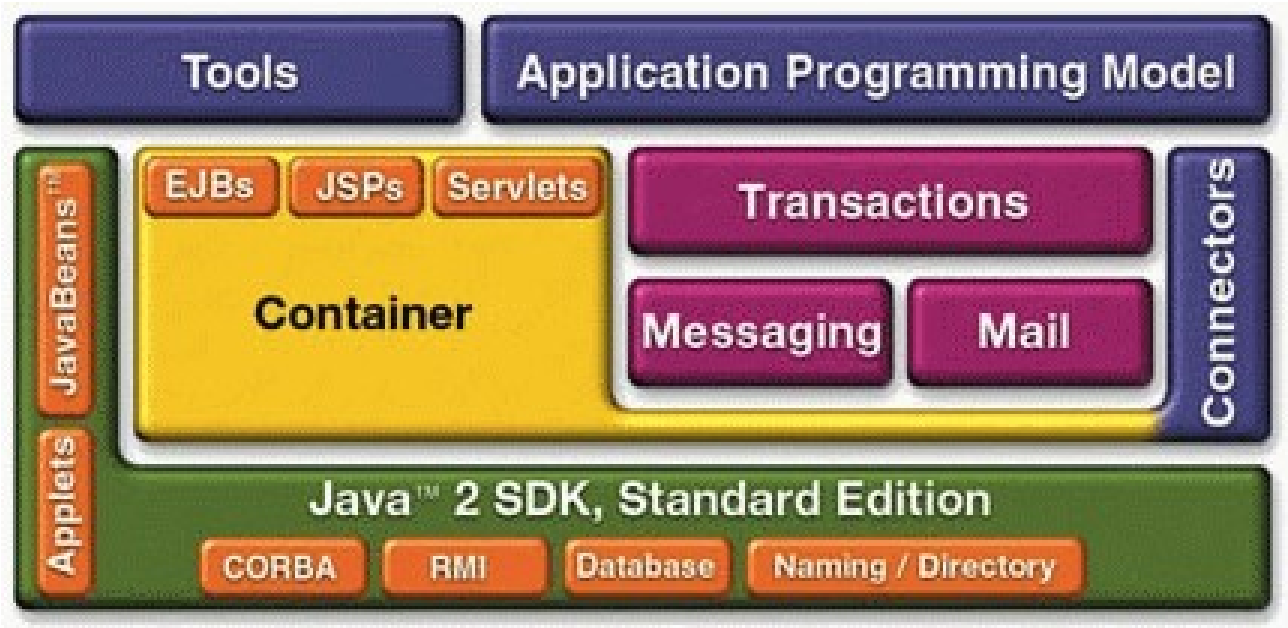
- J2EE es una plataforma para desarrollar aplicaciones distribuidas empresariales.
- J2EE consiste en:
  - Plataforma
  - Implementación de referencia
  - Test de compatibilidad
  - Planos de APM (Aplication Programming Model)

J2EE APIs

- J2SE
- JDBC
- RMI-JRMP
- Java IDL
- RMI-IIOP
- EJB
- Servlets
- JSP
- JMS
- JNDI
- JTA
- JavaMail
- JAF

Componentes J2EE

- Aplicación Java
- Applets
- Servlets y JSP
- EJB



Componentes / API

- Cada componente ha de cumplir una serie de APIs

	J2SE	*JDBC*	*JRMP*	*JIDL*	*RMI-IIOP*	*EJB*	*Servlets*	*JSP*	*JMS*	*JNDI*	*JTA*	*JavaMail*	*JAF*
Aplicación	x	x			x				x	x			
Applet	x												
Servlets/jsp	x	x			x		x	x	x	x	x	x	x
EJB	x	x	x	x	x	x	x	x	x	x	x	x	x

J2SE

- Java Platform, Standard Edition o Java SE (antes J2SE), es una colección de APIs del lenguaje de programación Java útiles para muchos programas de la Plataforma Java.
- J2SE 1.4 (Merlin), Java Community Process. JSR 59
- J2SE 5.0 (Tiger) = JSR 176 especificó
- Java SE 6 (Mustang) = JSR 270.

Librerías J2SE

- Generales
  - java.lang
  - java.io
  - java.math
  - java.net
  - java.text
  - java.util

Librerias J2SE (II)

- Propósito especial
  - java.applets
  - java.beans
  - java.awt
  - java.rmi
  - java.security
  - java.sql
  - javax.rmi
  - org.omg.CORBA
  - javax.swing

JDBC

- Java Database Connectivity
- API que permite interactuar con fuentes de datos, independientemente del fabricante
- Conjunto de definiciones de acceso y gestión de controladores

JDBC (ex.)

- Cargar Drivers

```
try{

Class.forName("nombreDelDriver").newInstance();
} catch (ClassNotFoundException cnfe) {
   Cnfe.printstacktrace();
}
```

- Abrir conexión

```
Connection c = DriverManager (url, "usuario", "password");
```

- Cerrar conexión

```
if (¡c.isClosed())
   c.close();
```

JDBC (ex2)

- Ejecutar sentencias SQL

```
Statement stmt = conn.createStatement();
try {
    ResultSet rs = stmt.executeQuery( "SELECT * FROM MyTable" );
    try {
        while ( rs.next() ) {
            int numColumns = rs.getMetaData().getColumnCount();
            for ( int i = 1 ; i <= numColumns ; i++ ) {
               System.out.println( "COLUMN " + i + " = " +
rs.getObject(i) );
            }
        }
    } finally {
        rs.close();
    }
} finally {
    stmt.close();
}
```

JDBC (transacciones)

- JDBC tiene soporte para transacciones

```
try {
  con.setAutoCommit(false);
  // run some SQL
  stmt.executeUpdate("UPDATE INV SET OH = 10 WHERE ID = 5");
  stmt.executeUpdate("INSERT INTO SHIP (QTY) VALUES (5)");
  con.commit();
}
catch (SQLException e) {
  con.rollback(); //undo the results of the transaction
}
```

JDBC (proc. almacenados)

- JDBC se puede usar para ejecutar procedimientos almacenados

```
CallableStatment
       cstmt = con.prepareCall("{call sp_interest(?,?)}");
cstmt.registerOutParameter(2, Types.FLOAT);
cstmt.setInt(1, accountID);
cstmt.setFloat(2, 2343.23);
cstmt.execute();
out.println("New Balance:" + cstmt.getFloat(2));
```

JDBC 2.0

- JDBC 1.0 – Mayo 1998
- JDBC 2.0 – 1999
  - Mejora en la lectura de los resultsets
  - Actualizaciones batch
  - Soporte OODBMS
  - BLOBS y CLOBS

JDBC 3.0

- JDBC 3.0 – 2002 (JSR-54)
  - Savepoints en las transacciones
  - Recuperación de claves autogeneradas
  - Actualización BLOB y CLOB
  - Multiples resultset abiertos
  - Recuperación de metadatos de parámetros
  -

Drivers JDBC

- Existen drivers para casi todas las BBDD del mercado, los drivers pueden ser:
  - **Tipo 1**, puente JDBC-ODBC
  - **Tipo 2**, Driver de API Nativa
  - **Tipo 3**, Driver de protocolo de red
  - **Tipo 4**, Drivers de protocolo nativo
  - **Interno**, driver JDBC embebido con el JRE
  - **URL JDBC**, cadena de conexión

RMI-JRMP

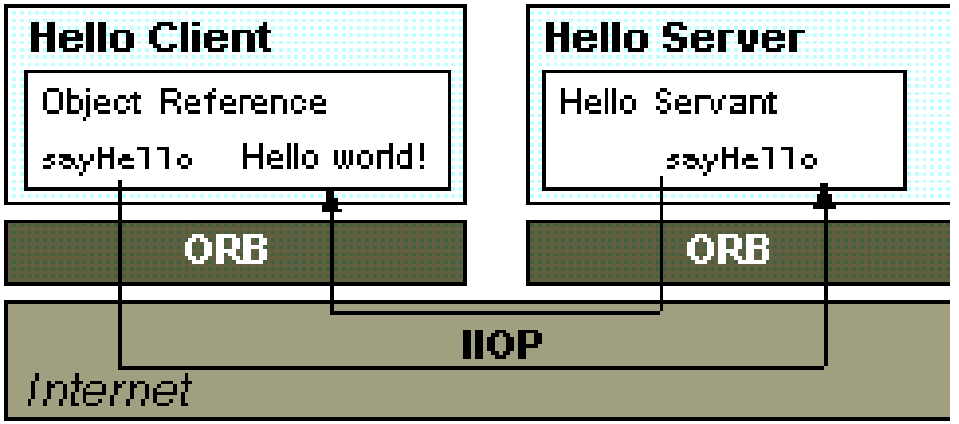
- Java Remote Method Protocol
- Protocolo específico para Java
- Permite referencias objetos remotos
- Es un protocolo que corre sobre TCP/IP bajo RMI.
- Se puede sustituir por IIOP o por otros desarrollados por terceros.

Java IDL

- Tecnología de objetos distribuidos
- Basado en "Common Object Request Brokerage Architecture" (CORBA)
- Permite intercomunicar sistemas escritos en lenguajes distintos
- Proporciona un broker (ORB) Object Request Broker
- **idltojava** para generar stubs y skeletons

Java IDL (ii)

- Comunicación CORBA:



RMI-IIOP

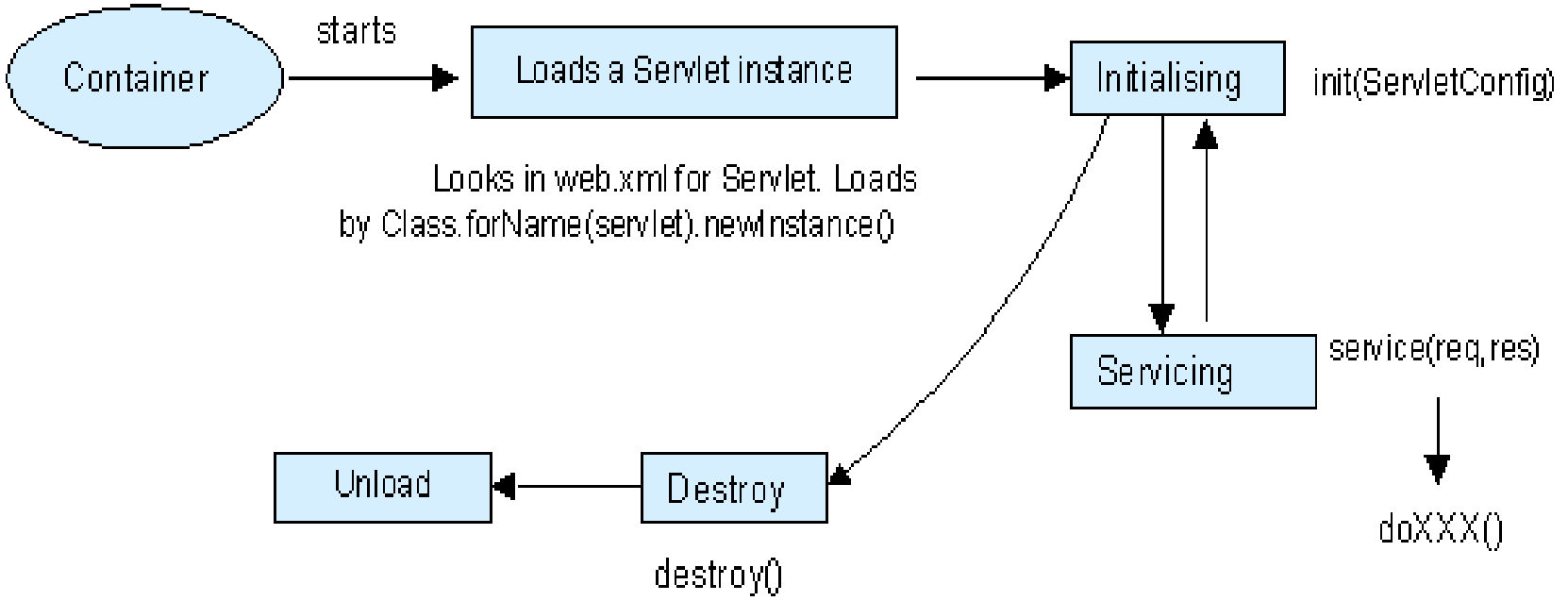
- IIOP (Internet Inter-Orb Protocol)
- Permite interoperar aplicaciones diseñadas para RMI con otras CORBA.
- RMI no requiere declaracion de IDL
- Presente desde Java 1.3
- Pretende simplificar los desarrollos CORBA

Servlet

- Un servlet es un objeto que se ejecuta en un contenedor web
- Fue diseñado para servir páginas dinámicas web
- Su nombre surge como contraposición a applet.
- Un servlet es un objeto Java que implementa la interfaz `javax.servlet.Servlet` o hereda alguna de las clases más convenientes para un protocolo específico (ej: `javax.servlet.HttpServlet`).

Ejecución Servlet

- Cargar instancia
- Inicializar
- Servir peticiones (todo el tiempo)
- Destruir
- Descargar



Servlet (ex.)

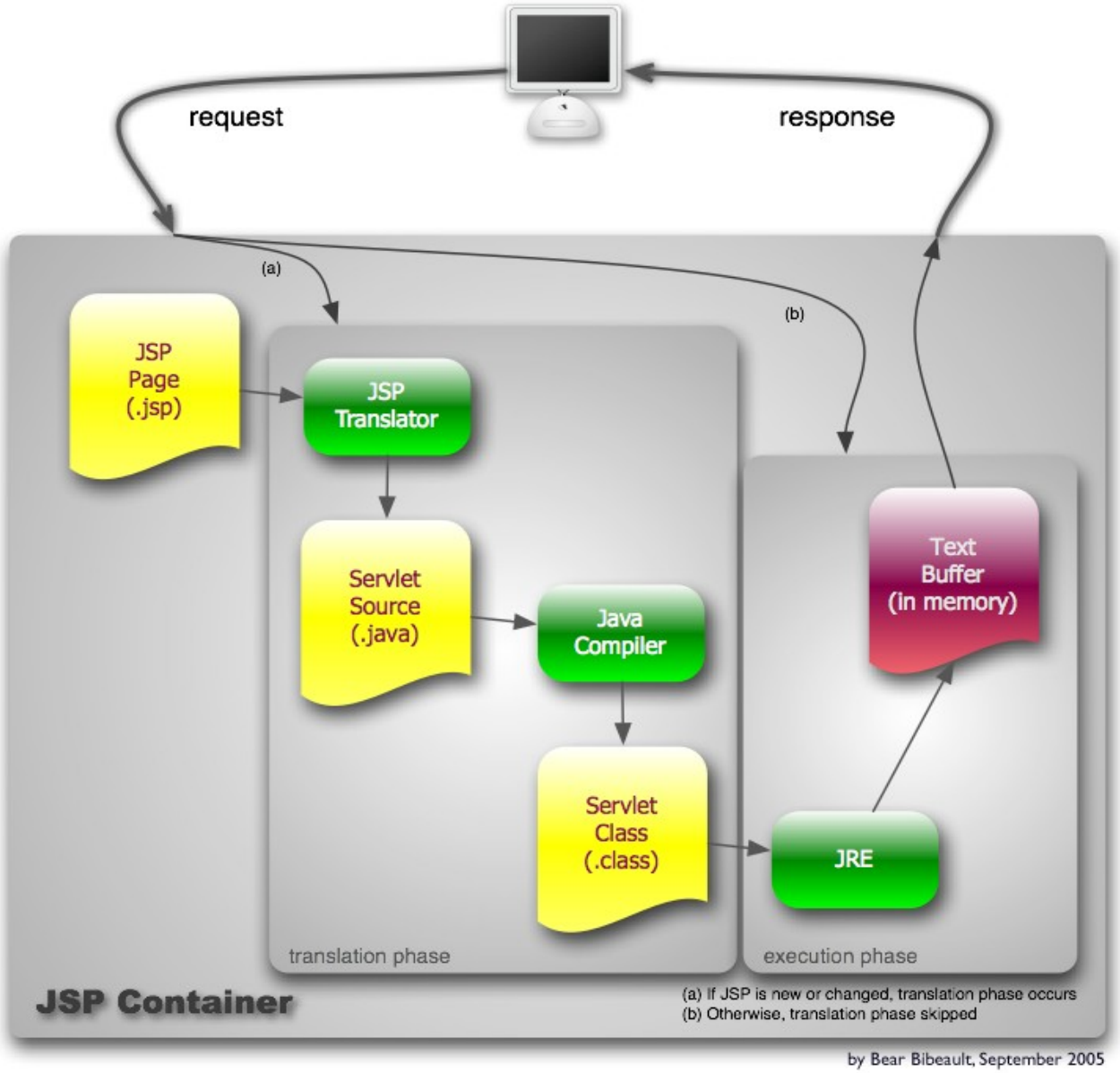
- El servlet “Hola Mundo”

```
public class HelloWorldServlet extends HttpServlet {

   protected void service (HttpServletRequest request,
                           HttpServletResponse response)
                  throws ServletException, IOException
    {
      ServletOutputStream out = response.getOutputStream();
      out.println("<html><body><h1>Hello
World</h1></body></html>");
    }

}
```

JSP

- Los servlets son muy tediosos para generar código html
- Java Server Pages se diseñó para generar la capa de presentación en forma de scripts
- Se permite “embeber” código java dentro de una página html o xhtml y que sea interpretada dinámicamente
- Se puede sustituir el código Java por etiquetas (TagLibs)

Ciclo de vida de un JSP



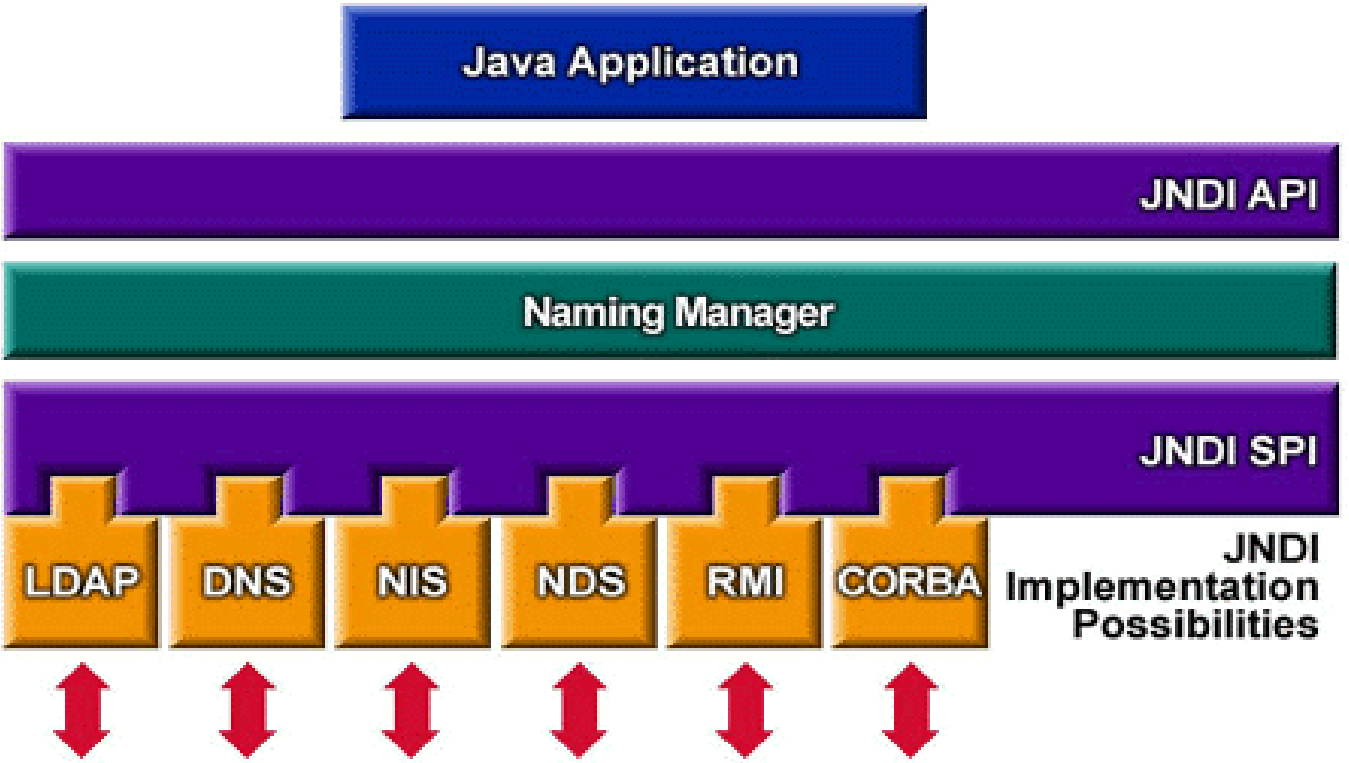
JSP (ex.)

```
<%@ page errorPage="myerror.jsp" %>
<%@ page import="com.foo.bar" %>
<html>
<head>
<% int serverInstanceVariable = 1;%>
<% int localStackBasedVariable = 1; %>
</head>
<body>
<table>
<tr>
<td><%=localStackBasedVariable%></td>
</tr>
</table>
</html>
```

JNDI

- Java Naming and Directory Interface
- API de acceso a servicios de nombre y directorios en Java
- Pretende asociar nombres con objetos para poder acceder a ellos de una manera estandar
  - Archivos
  - Nombres DNS
  - EJBs
  - Bases de datos ...

Arquitectura JNDI



JNDI (ex.)

```
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
    "com.sun.jndi.fscontext.RefFSContextFactory");
Context ctx = new InitialContext(env);
Object obj = ctx.lookup (name);
try {
    // Create the initial context
    Context ctx = new InitialContext(env);

    // Look up an object
    Object obj = ctx.lookup(name);

    // Print it
    System.out.println(name + " is bound to: " + obj);

} catch (NamingException e) {
    System.err.println("Problem looking up " + name + ": "
+ e);
}
```

Factorias JNDI

- Filesystem
  - `com.sun.jndi.fscontext.FSContextFactory`
  - `com.sun.jndi.fscontext.RefFSContextFactory`
- LDAP
  - `com.sun.jndi.ldap.LdapCtxFactory`
- NDS
  - `com.novell.naming.service.nds.NdsInitialContextFactory`
- RMI registry
  - `com.sun.jndi.rmi.registry.RegistryContextFactory`

JNDI (ex2)

- Recuperar una conexión de bbdd de un servidor de aplicaciones

```
DataSource ds = null;

 try
 {
     Context ctx = new InitialContext();
     if(ctx == null) throw new Exception("JNDI Context could not be found.");
     ds = (DataSource)ctx.lookup(“jdbc/MifuenteCoreDS”);
     if(ds == null) throw new Exception("Datasource could not be found");
     connection = ds.getConnection("user","pass");
     connection.setAutoCommit(false);
}
catch(Exception e)
{
     e.printStackTrace();
}
```

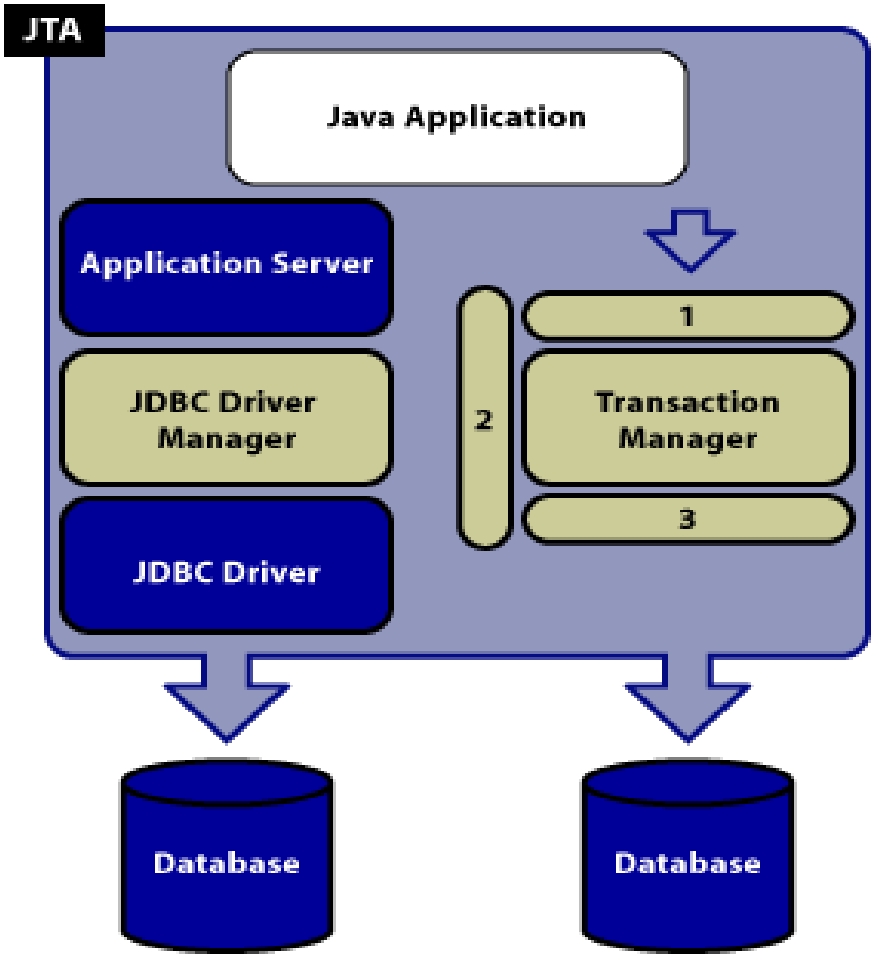
JMS

- Java Message Service es el API encargada de mensajería y MOM (message-oriented middleware)
- Mensaje = conjunto de datos enviados desde un sistema a otro
- JMS es un API que sirve de intermediario, como JDBC a implementaciones de fabricantes
- JMS provider = driver mensajería

JTA

- Java Transaction API
- Define XA para Java (estandar del Open Group para transacciones distribuidas)
- JTA permite usar transacciones distribuidas trasparentemente
- Generalmente JTA se usa internamente en el servidor de aplicaciones
  - UserTransaction
  - TransactionManager
  - XAResource

JTA


JTA
Java Application
Application Server
JDBC Driver Manager
JDBC Driver
2
1
Transaction Manager
3
Database
Database

JavaMail

- API que provee un framework para crear aplicaciones de correo y mensajería (no JMS)
- La versión actual es la 1.4 (JSR-919)
- Requiere JAF para gestionar tipos MIME
- Opcional para J2SE, incluido con J2EE
- Permite enviar y recibir correos (smtp, pop,imap,...)

JavaMail (ex.)

```
String smtpHost = "smtp.auna.com";
Properties props = System.getProperties();
props.put("mail.smtp.host",smtpHost);
Session
   sesion = Session.getDefaultInstance(props,null);
try {
  Message mensaje = new MimeMessage(sesion);
  mensaje.setSubject("Hola Mundo");
  mensaje.setFrom(new InternetAddress(from));
  mensaje.addRecipient( Message.RecipientType.TO,
                        new InternetAddress(to));
  mensaje.setText("Este es el cuerpo del mensaje");
  Transport.send(mensaje);
} catch (MessagingException e) {
   System.err.println(e.getMessage());
}
```

JAF

- JavaBeans Activation Framework
- API que permite determinar el tipo de un conjunto de datos
- Permite registrar nuevos tipos de datos y acciones asociadas
- Permite que los programas carguen JavaBeans que implementan acciones sobre un tipo de datos

Versiones J2EE

- La versión más actual es Java EE SDK 5 Update 3
- Básicamente varía según la versión de los componentes utilizados
- Para entornos de producción se utilizan servidores de aplicación que soportan una u otra versión de J2EE o de las especificaciones de sus contenedores

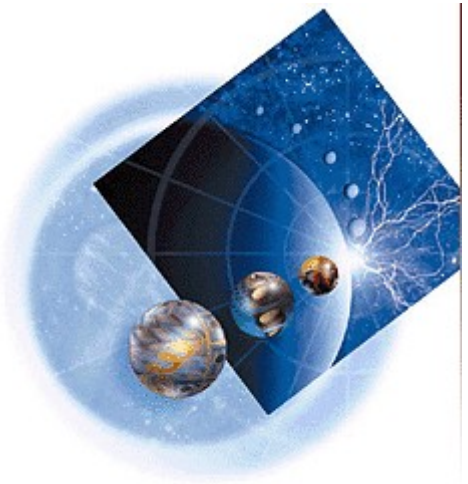
Servidores de aplicación J2EE

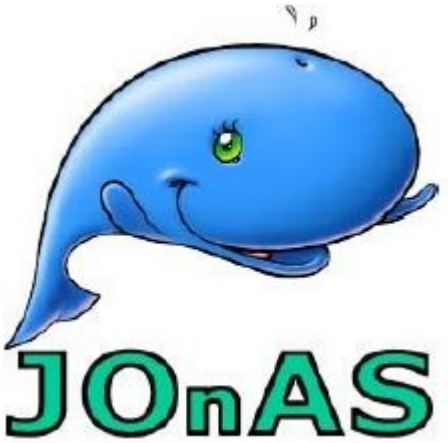
- Implementan los contenedores de manera más eficiente
- Incluyen herramientas de administración
- Están diseñados para entornos de producción.
- Los más conocidos:
  - BEA Weblogic
  - IBM WebSphere
  - Oracle Application Server
  - Sun Java System Application Server

Servidores J2EE Open Source

- Utilizados cada vez más a menudo
- Suelen tener menos herramientas de administración
- Cumplen con las especificaciones de la plataforma
  - GlassFish
  - RedHat JBoss
  - JOnAS
  - Geronimo

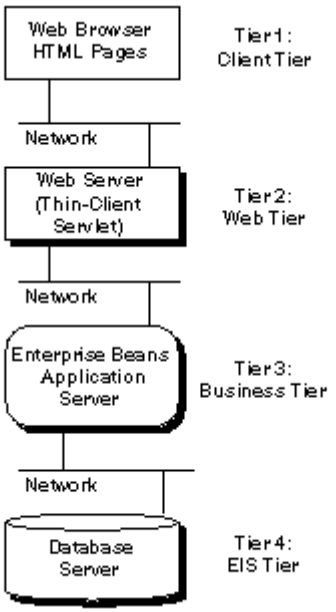
Servidores J2EE





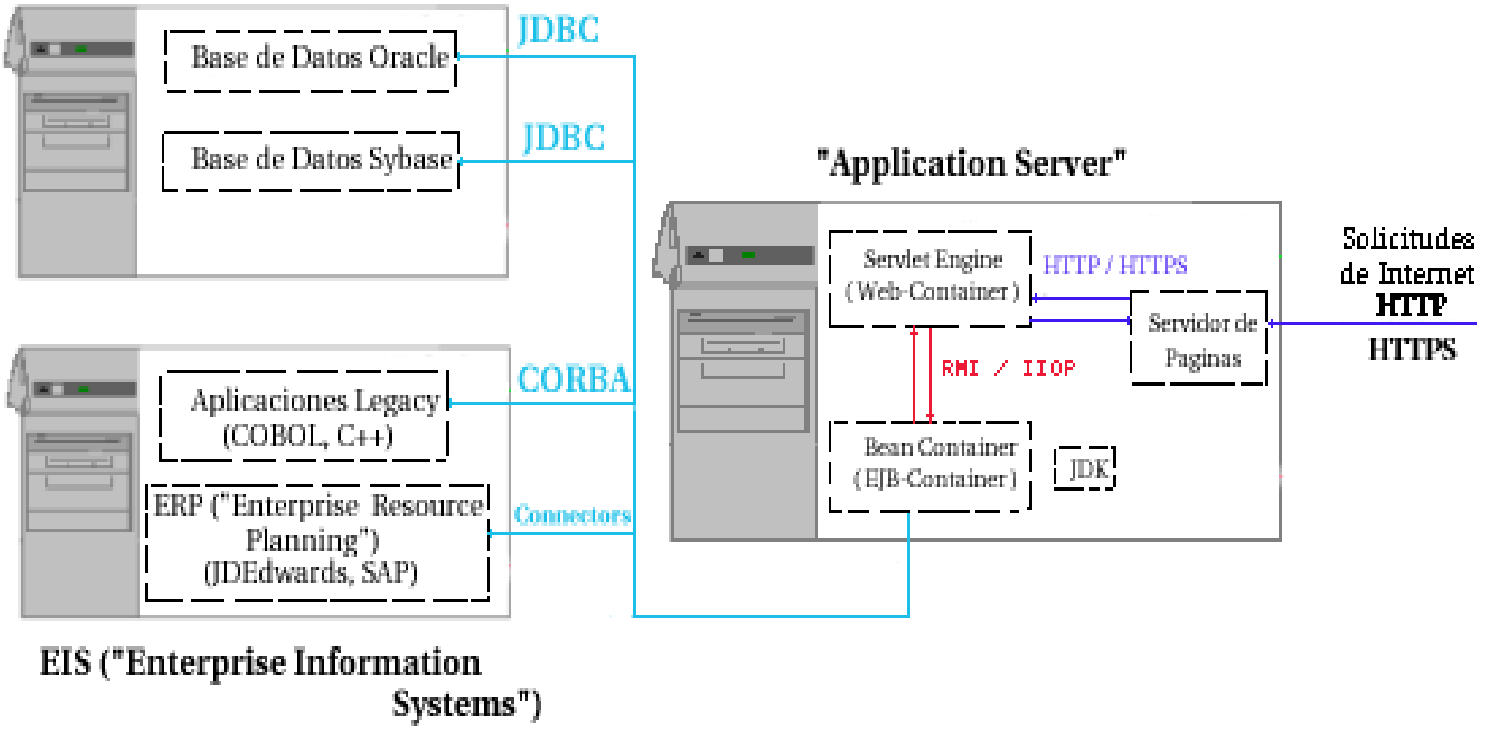
Las 4 capas

- Capa 1: Cliente
- Capa 2: Web
- Capa 3: Negocio
- Capa 4: Datos



- Cada capa se ejecuta en un contenedor distinto

Arquitectura típica



El contenedor web

- Permite ejecutar JSP y Servlets
- Históricamente fue el primero en desarrollarse
- Hay muchos proyectos que no utilizan nada más que este contenedor
- Lo único que no se puede ejecutar son los EJBs

Contenedores Web

- Apache Tomcat
- Jetty
- Caucho (Resin Server)
- BEA Weblogic Server
- Oracle AS
- IBM Websphere
- Macromedia Jrun
- ...

Tomcat

- **Tomcat 3.x (distribución inicial)**
  - Implementado a partir de las especificaciones **Servlet 2.2 y JSP 1.1**
  - Recarga de servlets
  - Funciones básicas HTTP
- **Tomcat 4.x**
  - Implementado a partir de las especificaciones **Servlet 2.3 y JSP 1.2**
  - Contenedor de servlets rediseñado como Catalina
  - Motor JSP rediseñado con Jasper
  - Conector Coyote
  - Java Management Extensions (JMX), JSP Y administración basada en Struts
- **Tomcat 5.x**
  - Implementado a partir de las especificaciones **Servlet 2.4 y JSP 2.0**
  - Recolección de basura reducida
  - Capa envolvente nativa para Windows y Unix para la integración de las plataformas
  - Análisis rápido JSP
- **Tomcat 6.x**
  - Implementado de **Servlet 2.5 y JSP 2.1**
  - Soporte para Unified Expression Language 2.1
  - Diseñado para funcionar en Java SE 5.0 y posteriores
  - Soporte para Comet a través de la interfaz CometProcessor

JARs, WARs, EARs y más

- JAR : Java ARchives es un formato desarrollado por Sun que permite agrupar y comprimir archivos (como un zip)
- WAR : Web ARchives especificación de archivo JAR que permite agrupar un conjunto de clases y recursos que conforman una aplicación web y que pueden ser utilizados por contenedores web diréctamentene.
- EJB-JAR : Equivalente al War pero para EJBs
- EAR : Enterprise Archives: combinación de WAR y EJB-JAR

Estructura WAR

- La estructura de un (Web-Archive) WAR es la siguiente:
  - / *.html *.jsp *.css : Este directorio base contiene los elementos que comúnmente son utilizados en un sitio, Documentos en HTML , JSP's , CSS("Cascading Style Sheets") ,etc.
  - **/WEB-INF/web.xml** : Contiene elementos de seguridad de la aplicación así como detalles sobre los Servlets que serán utilizados dentro de la misma.
  - **/WEB-INF/classes/** : Contiene las clases Java adicionales a las del JDK que serán empleadas en los JSP's y Servlets
  - **/WEB-INF/lib/** : Contiene los JAR's que serán utilizados por su aplicación.

Estructura HelloWorld

- ./WEB-INF:
  - web.xml
- ./WEB-INF/classes
- ./WEB-INF/src/com/digimate/example:
  - HelloWorldServlet.java

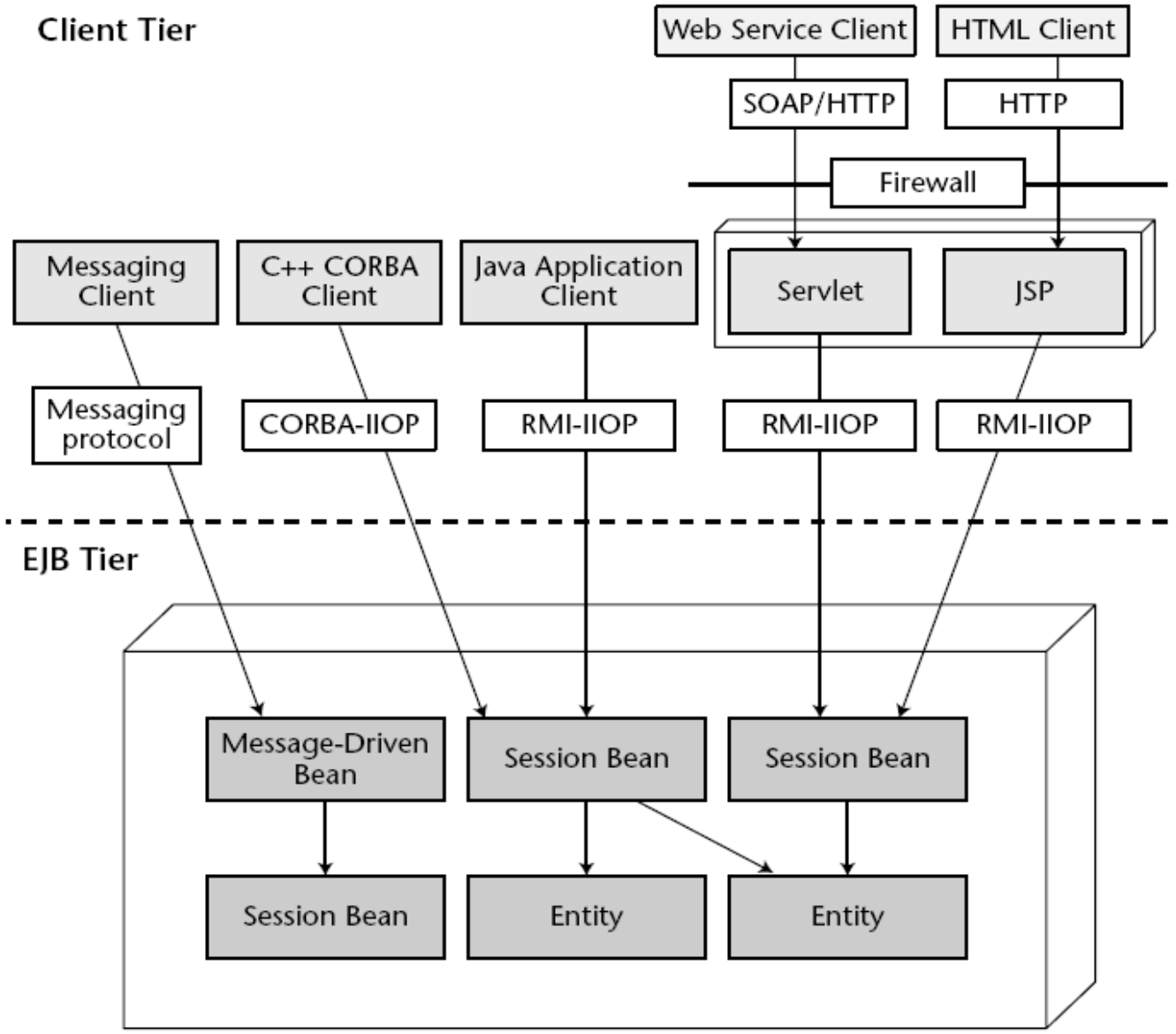
web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
        xmlns="http://java.sun.com/xml/ns/j2ee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
        <servlet>
                <servlet-name>HelloWorldServlet</servlet-name>
                <servlet-
class>com.digimate.example.HelloWorldServlet</servlet-class>
        </servlet>
        <servlet-mapping>
                <servlet-name>HelloWorldServlet</servlet-name>
                <url-pattern>/Hello</url-pattern>
        </servlet-mapping>
</web-app>
```

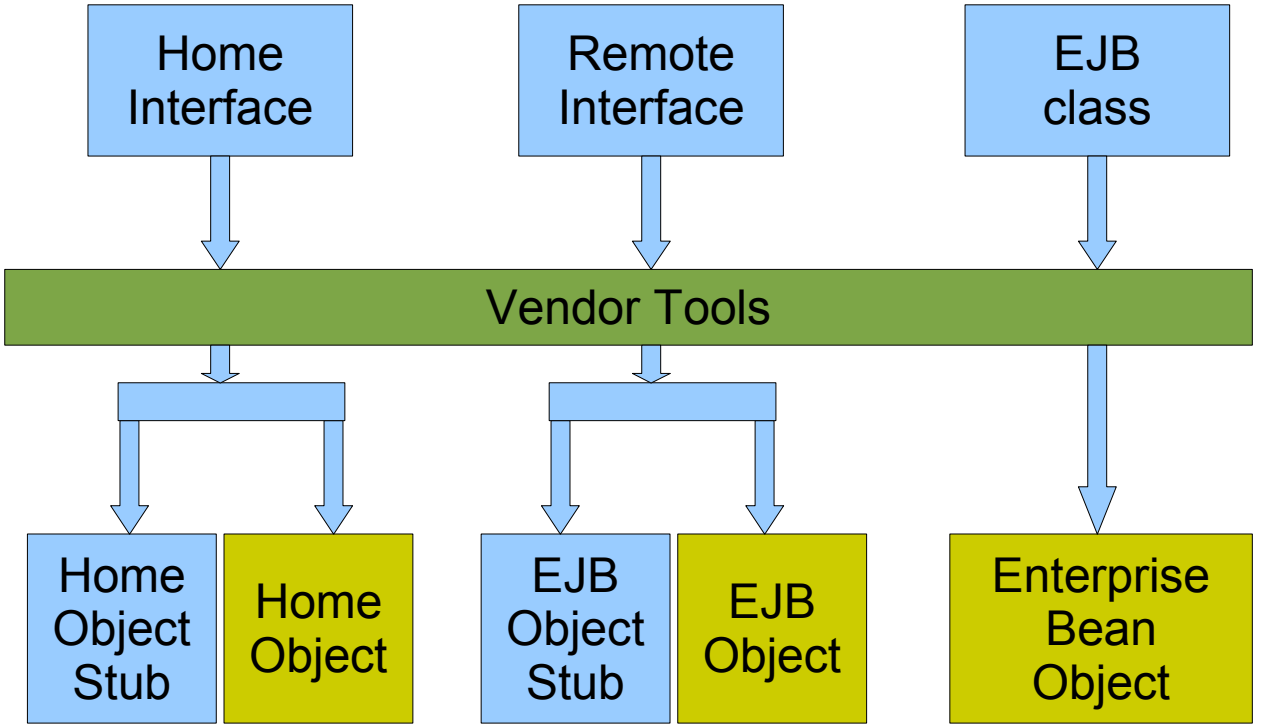
Usando EJBs

- Un EJB tiene las siguientes características:
  - Contiene lógica de negocio
  - Creado y gestionado por un contenedor
  - Media en el acceso al cliente
  - Contiene metadatos como transacciones, seguridad, etc. separados del bean
  - Provee de gestión de transacciones, gestión de estado, pooling de recursos y chequeos de seguridad.

Uso EJB

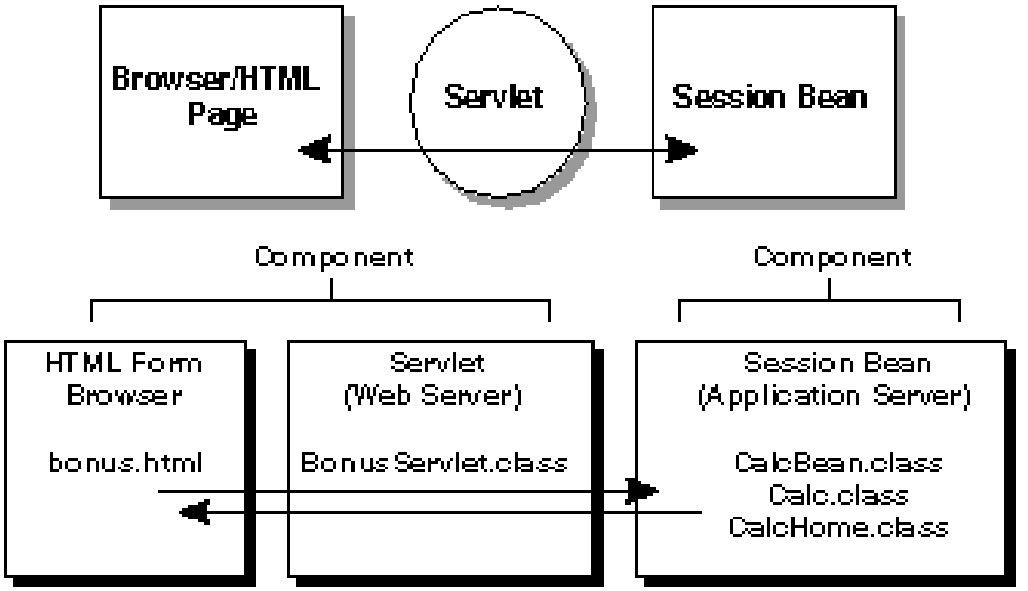

Client Tier
Web Service Client
HTML Client
SOAP/HTTP
HTTP
Firewall
Messaging Client
C++ CORBA Client
Java Application Client
Servlet
JSP
Messaging protocol
CORBA-IIOP
RMI-IIOP
RMI-IIOP
RMI-IIOP
EJB Tier
Message-Driven Bean
Session Bean
Session Bean
Session Bean
Entity
Entity

Clases e interfaces EJB



Ejemplo EJB

- Los EJBs pueden ser:
  - Sesion
  - Entidad
- Ejemplo de Session Bean



EJB

- Formulario HTML

```
<HTML>
<BODY BGCOLOR = "WHITE">
<BLOCKQUOTE>
<H3>Bonus Calculation</H3>
<FORM METHOD="GET"  ACTION="BonusAlias">
<P>Enter social security Number:<P>
<INPUT TYPE="TEXT" NAME="SOCSEC"></INPUT>
<P>Enter Multiplier:<P>
<INPUT TYPE="TEXT" NAME="MULTIPLIER"></INPUT>
<P>
<INPUT TYPE="SUBMIT" VALUE="Submit">
<INPUT TYPE="RESET">
</FORM>
</BLOCKQUOTE>
</BODY>
</HTML>
```

EJB - Servlet

- Recupera los datos del usuario
- Busca el bean de sesión
- Le pasa los datos al bean de sesión
- Después de recibir un valor de vuelta desde el bean de sesión, crea una página HTML para mostrar el valor devuelto al usuario.

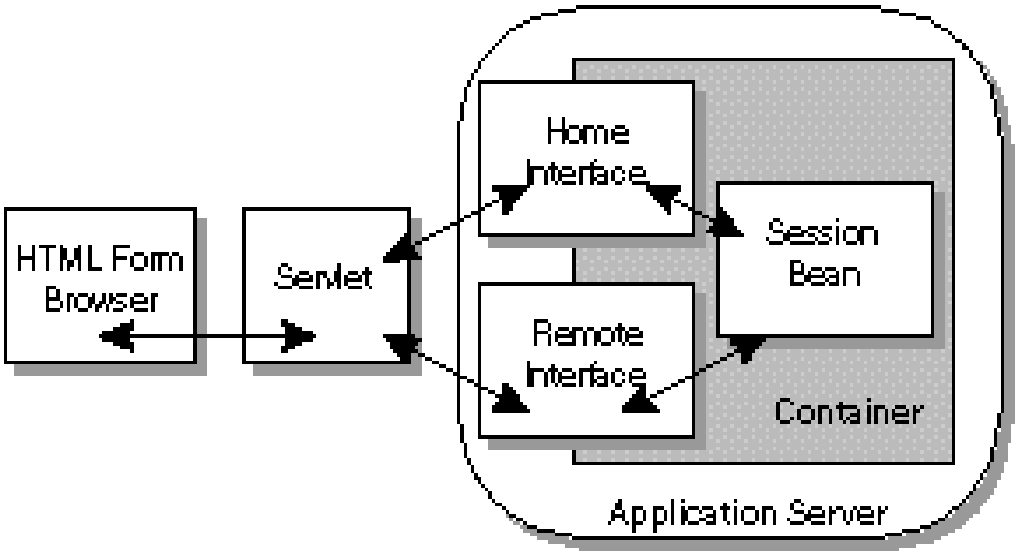
EJB – Servlet (ii)

```
public void doGet (HttpServletRequest
request,
                   HttpServletResponse
response)
throws ServletException, IOException {
    String socsec = null;
    int multiplier = 0;
    double calc = 0.0;
    PrintWriter out;
    response.setContentType("text/html");
    out = response.getWriter();
    out.println("<HTML><HEAD><TITLE>")
    out.println("EJB Example");
    out.println("</TITLE></HEAD><BODY>");

    try{
      String strMult =
request.getParameter("MULTIPLIER");
      Integer integerMult = new
Integer(strMult);
      multiplier = integerMult.intValue();
      socsec =
request.getParameter("SOCSEC");
      double bonus = 100.00;
      theCalculation = homecalc.create();
      calc = theCalculation.calcBonus(
             multiplier, bonus);
    }catch(Exception CreateException){
      CreateException.printStackTrace();
    }
```

```
out.println("<H1>Bonus
Calculation</H1>");
     out.println("<P>Soc Sec: " +
socsec + "<P>");
    out.println("<P>Multiplier: " +
                multiplier + "<P>");
    out.println("<P>Bonus Amount: " +
calc + "<P>");
    out.println("</BODY></HTML>");
    out.close();
  }
```

EJB Sesion

- Un bean de sesion representa una conversación temporal con el cliente.
- NO es persistente



EJB (Home/Object)

- CalcHome

```
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;

public interface CalcHome extends EJBHome {
  Calc create() throws CreateException,
                       RemoteException;
}
```

- Calc

```
import javax.ejb.EJBObject;
import java.rmi.RemoteException;

public interface Calc extends EJBObject {
  public double calcBonus(int multiplier,
                          double bonus)
                   throws RemoteException;
}
```

EJB (Bean)

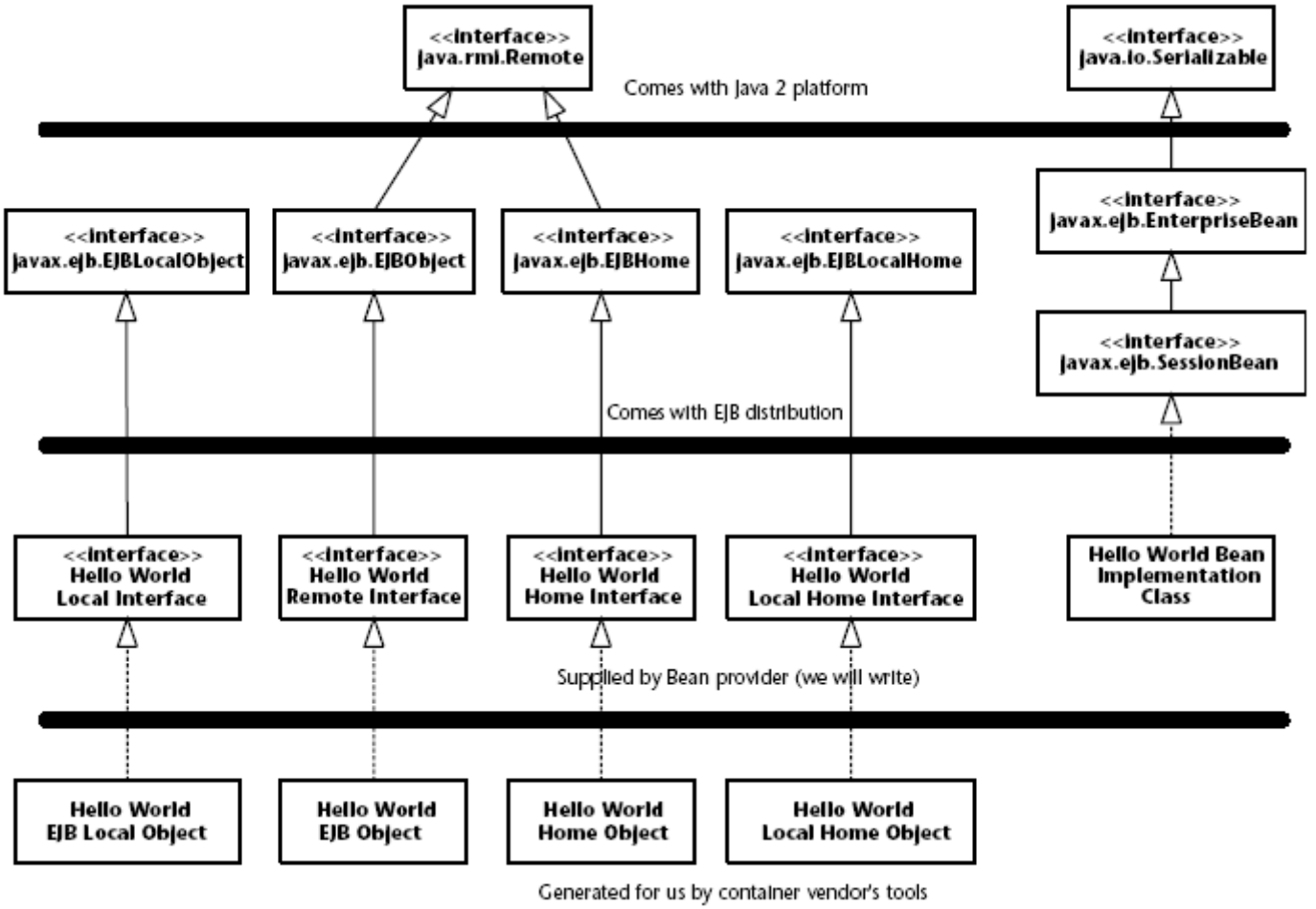
- CalcBean

```
import java.rmi.RemoteException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;

public class CalcBean implements SessionBean {
  public double calcBonus(int multiplier,
                          double bonus) {
    double calc = (multiplier*bonus);
    return calc;
  }

  public void ejbCreate() { }
  public void setSessionContext(
      SessionContext ctx) { }
  public void ejbRemove() { }
  public void ejbActivate() { }
  public void ejbPassivate() { }
  public void ejbLoad() { }
  public void ejbStore() { }
}
```

EJB 2.x

- Complejidad alta para el desarrollador
- Ventajas no utilizables para la mayoría de los sistemas
- Dificultad muy grande para desplegar / probar
- Penalización general al rendimiento por el uso de tantos interfaces

EJB 2.x (ex.)


<<interface>>
java.rmi.Remote
Comes with Java 2 platform
<<interface>>
java.io.Serializable
<<interface>>
javax.ejb.EJBLocalObject
<<interface>>
javax.ejb.EJBObject
<<interface>>
javax.ejb.EJBHome
<<interface>>
javax.ejb.EJBLocalHome
<<interface>>
javax.ejb.EnterpriseBean
<<interface>>
javax.ejb.SessionBean
Comes with EJB distribution
<<interface>>
Hello World
Local Interface
<<interface>>
Hello World
Remote Interface
<<interface>>
Hello World
Home Interface
<<interface>>
Hello World
Local Home Interface
Hello World Bean
Implementation
Class
Supplied by Bean provider (we will write)
Hello World
EJB Local Object
Hello World
EJB Object
Hello World
Home Object
Hello World
Local Home Object
Generated for us by container vendor's tools

EJB 3.0

- Mejora la especificación EJB
- Reduce la complejidad de cara al desarrollador
  - Anotaciones
  - API Simplificada
  - Mejoras EJB-QL
- Reacción a los frameworks

EJB 3.0

- Eliminación de los interfaces Home y Object
- Eliminación del componente interfaz
- Uso de anotaciones (metadata)
- Acceso simplificado al entorno
- Simplificación en el despliegue

Frameworks Web

- Muchos desarrolladores de aplicaciones Java decidieron que no necesitaban EJBs
- Nunca usarían lógica de negocio tan compleja
- No necesitarían distribución
- Querían simplificar el despliegue y configuración de las aplicaciones

Frameworks software

- Un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.
- Diseño reusable de un sistema (o subsistema)
- Conjunto de clases abstractas y el modo en que sus instancias colaboran para un tipo específico de software.

Framework web

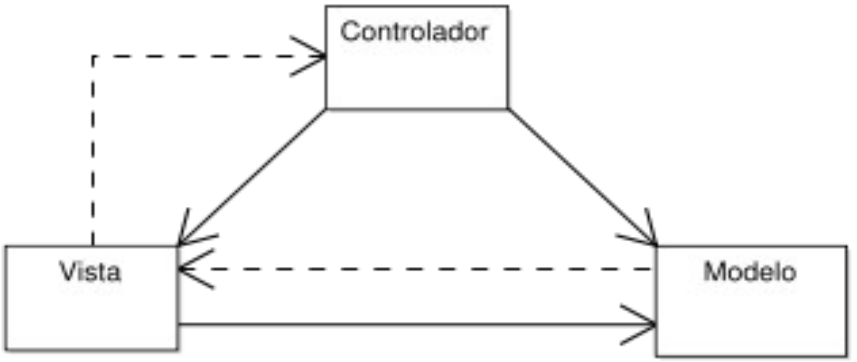
- Estructura definida, reusable en el que sus componentes facilitan la creación de aplicaciones web.
- Framework web Java, es aquel framework que utiliza la plataforma Java como base

Framework Modelo 1 / 2

- Modelo 1
  - Se incluye la lógica de redirección en los JSP
  - Cada página procesa su propio input
- Modelo 2
  - Servlet controlador (Front controller)
  - Según la entrada y el estado actual de la aplicación decide a qué página redirigir
  - Las vistas no se relacionan con el modelo

MVC

- Modelo – Vista – Controlador
- **Controlador**: recibe acciones de usuarios externos
- **Modelo**: estado interno y reglas de negocio
- **Vistas**: formato del modelo para mostrar

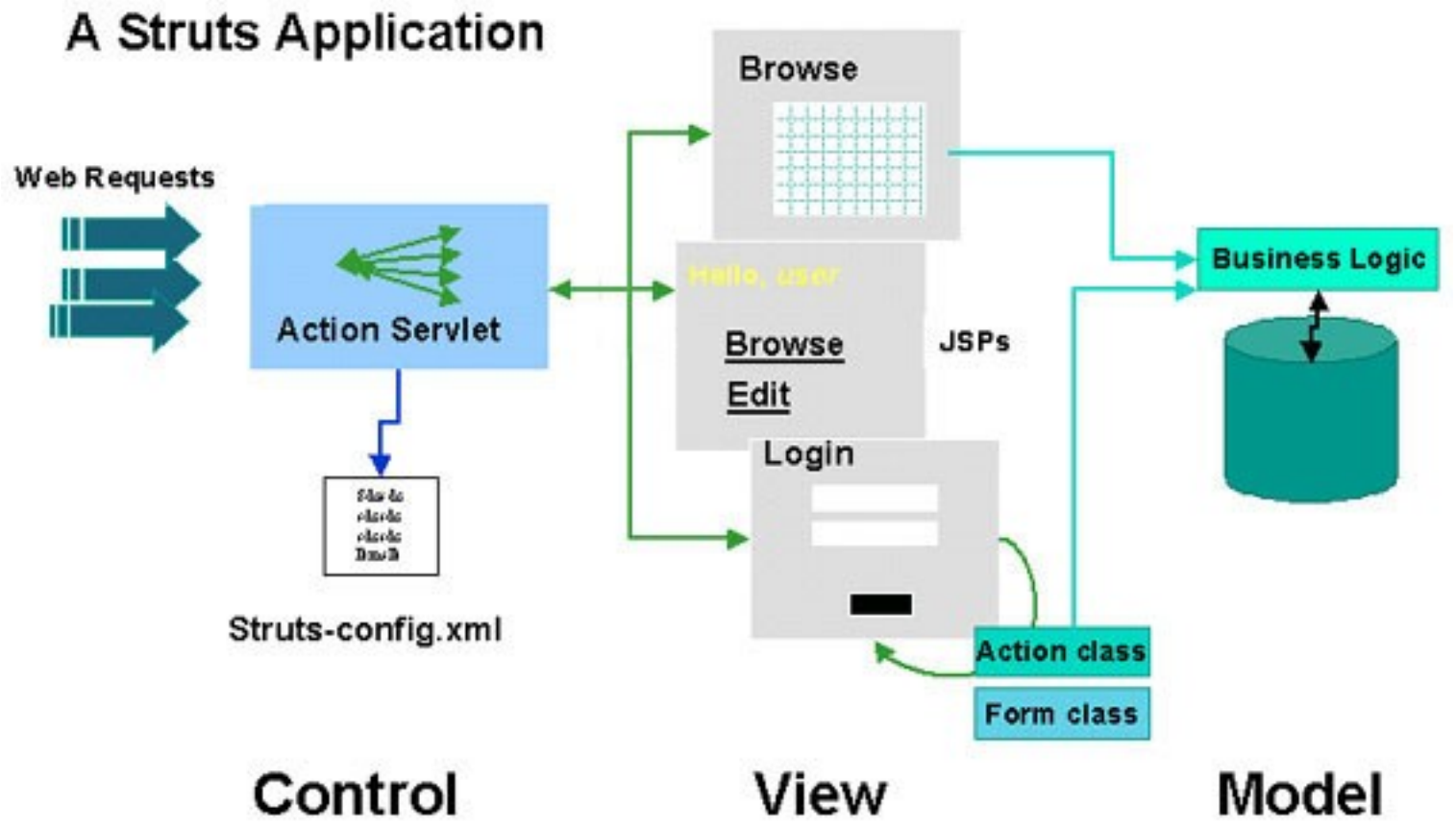


Frameworks Java

- Struts
  http://struts.apache.org
- Tapestry
  http://tapestry.apache.org
- JSF
  http://java.sun.com/javaee/javaserverfaces
- Cocoon
  http://cocoon.apache.org
- Spring
  http://www.springframework.org

Struts

Struts 2



Struts (facilidades)

- Lógica de navegación entre páginas
- Binding entre Java y html
- Validación de entradas
- Internacionalización
- Independencia del motor de visualización
- Maquetación

Struts (funcionamiento básico)

- El cliente solicita una página que contiene datos a completar. (no mostrado)
- El servidor le envía la página. (no mostrado)
- El cliente, con los datos completados envía de regreso la página. El ActionServlet verifica la ruta con la que se lo invocó y extrae el path de esa ruta y busca en los actionMappings cual es la Acción a invocar y que formulario necesita recibir como entrada.
- El controlador crea o reutiliza el Formulario dependiendo el ámbito en que es ejecutada la petición, carga los datos en el formulario, los valida y luego crea la acción y le pasa el formulario como parámetro.
- La acción recibe el formulario y con sus datos invoca a las reglas del negocio (generalmente delegadas en otros objetos).
- A partir de la respuesta recibida, carga los valores de salida y selecciona la siguiente vista a enviar.

Struts-config.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC ... >
    <struts-config>
        <form-beans>
        <form-bean name="loginForm" type="com.empresa.LoginForm"/>
        </form-beans>
        <action-mappings>
            <action path="/inicio" forward="/jsp/inicio.jsp"/>
            <action path="/login" forward="/jsp/login.jsp"/>
            <action path="/slogin" type="com.empresa.LoginAction"
name="loginForm"
            scope="request" validate="true"
input="/jsp/login.jsp">
                <forward name="exito" path="/jsp/inicio.jsp"/>
                <forward name="falla" path="/jsp/login.jsp"/>
            </action>
            <action path="/salir" type="com.empresa.SalirAction">
                <forward name="exito" path="/jsp/salir.jsp"/>
            </action>
        </action-mappings>
        <message-resources parameter="resources.application"/>
    </struts-config>
```

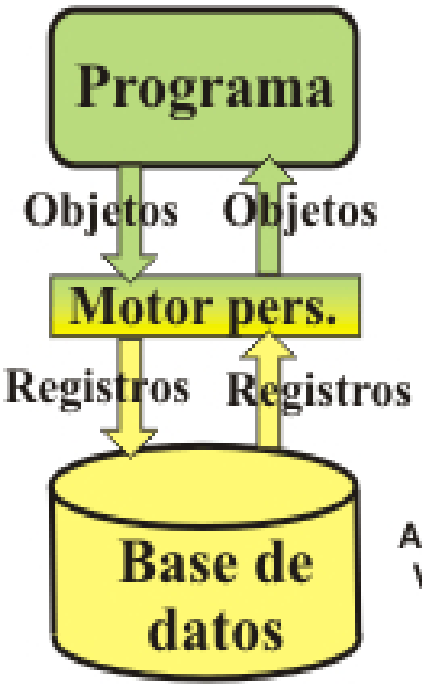
Ejemplo

- Tutorial JDeveloper
  http://www.oracle.com/technology/oramag/oracle/04-sep/o54jdev.html

ORM

- Object Relational Mapping
- En realidad un concepto independiente de Java
- Pretende abstraer el almacenamiento de objetos en bases de datos
- Motor de persistencia

Motores de persistencia



Amarillo significa relacional.
Verde significa orientado a objetos.

Motores de persistencia Java

- Hibernate
- Oracle TopLink
- Castor
- Torque
- Cayenne
- JDO

POJO

- Plain Old Java Object
- Uso de clases simples independientes del framework
- Un enfoque actual para los motores de persistencia persigue permitir la utilización de POJOs para representar la información y automatizar su almacenaje.

“Everything should be made as simple as possible, but not simpler”
Albert Einstein

EJEMPLO TopLink

Portlets

- Componentes modulares de presentación
- Requieren un portal web para ser visualizados
- Especificaciones:
  - JSR168
  - WSRP (Web Services for Remote Portlets)

Portlets - ventajas

- Se puede desacoplar la ejecución del portlet de la máquina portal
- Se puede reutilizar el portlet en distintas páginas
- Se pueden auditar dentro del portal
- Se puede controlar el acceso y la visualización dentro del portal

Portlets: portales

- Oracle Portal
- IBM
- BEA Portal
- Apache Pluto
- uPortal
- Sun Portal
- Apache JetSpeed
- JBoss Portal

Portlets Ejemplo

AJAX

- Asynchronous JavaScript And XML
- La intención es crear clientes “ricos”
- Mantiene comunicación asíncrona con el servidor en segundo plano
- Mejora de:
  - Interactividad
  - Velocidad
  - Usabilidad
- No mantiene la accesibilidad

AJAX

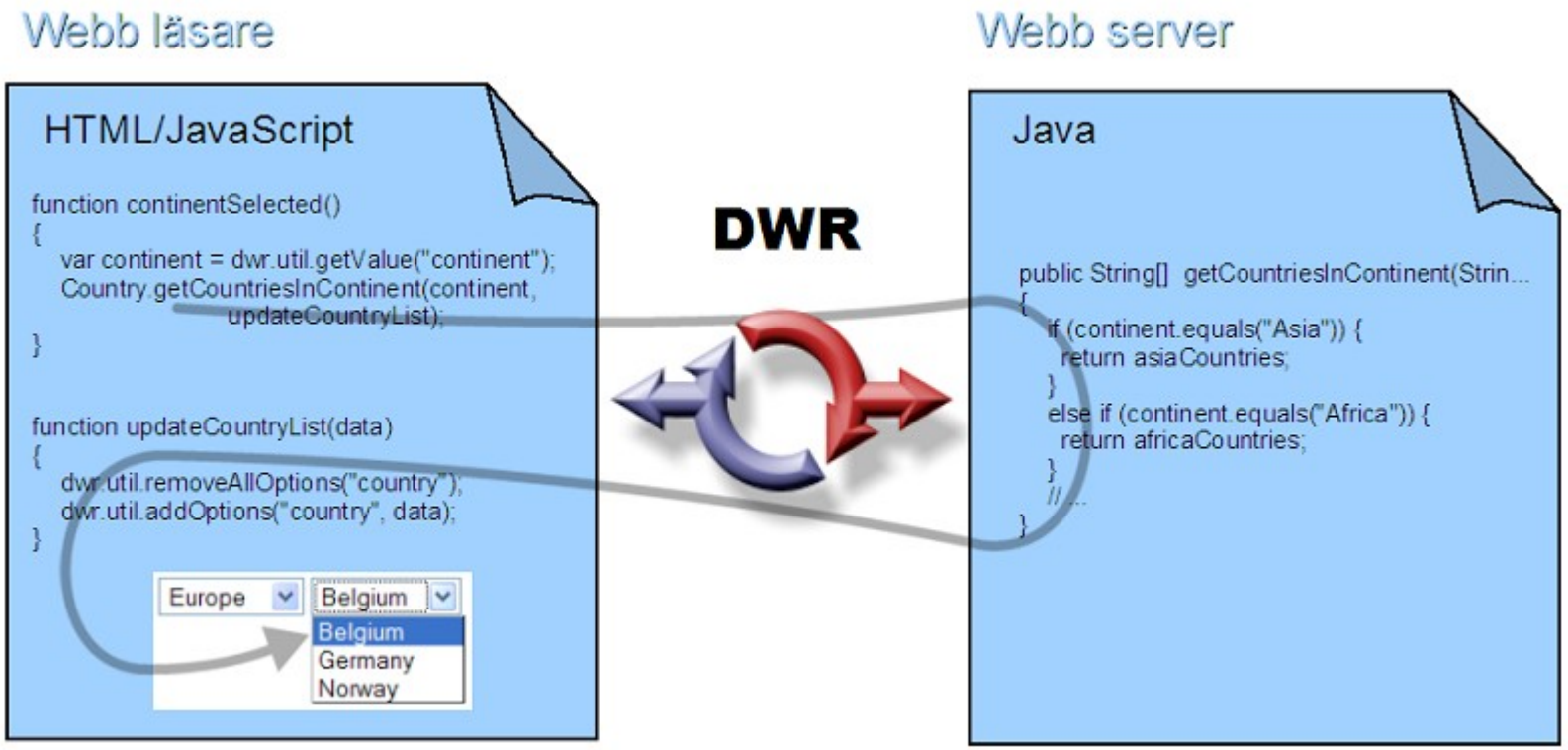
- Combinación de tecnologías
  - XHTML / HTML + css
  - Document Object Model (DOM)
  - XMLHttpRequest
  - XML
- Ajax no es una tecnología nueva, simplemente agrupa las que existen
- No está soportado por todos los navegadores

Frameworks Ajax

- AJAX no es exclusivo de Java
- Frameworks / librerías
  - Bindows
  - Prototype
  - Script.aculo.us
  - Dojo
  - DWR
  - ...

DWR

- Generación automática del código JavaScript correspondiente a Java



dwr.xml

```
<!DOCTYPE dwr PUBLIC
    "-//GetAhead Limited//DTD Direct Web Remoting 1.0//EN"
    "http://www.getahead.ltd.uk/dwr/dwr10.dtd">

<dwr>

  <init>
    <creator id="..." class="..."/>
    <converter id="..." class="..."/>
  </init>

  <allow>
    <create creator="..." javascript="..."/>
    <convert converter="..." match="..."/>
  </allow>

  <signatures>
    ...
  </signatures>

</dwr>
```

dwr - web.xml

```
<servlet>
        <servlet-name>dwr-invoker</servlet-name>
        <display-name>DWR Servlet</display-name>
        <description>Direct Web Remoter Servlet</description>
        <servlet-class>uk.ltd.getahead.dwr.DWRServlet</servlet-class>
        <init-param>
            <param-name>debug</param-name>
            <param-value>true</param-value>
        </init-param>
    </servlet>


    <servlet-mapping>
        <servlet-name>dwr-invoker</servlet-name>
        <url-pattern>/dwr/*</url-pattern>
    </servlet-mapping>
```

DWR - Ejemplo

Problemas AJAX

- Accesibilidad
- Perdida de gestionabilidad
- Cambio del control de páginas
- Vuelta a un esquema pseudo-cliente-servidor
- No todos los navegadores lo soportan
- Aumenta la concurrencia de los accesos.

SOA

- Service Oriented Architecture
- Utilización de servicios orquestados
- Evolución de Workflow y BPM
- Basada en ejecución de servicios web
- Independiente de plataforma y de lenguaje
- Alta complejidad

SOA - JAVA

- Tecnologías usadas:
  - XML
  - HTTP
  - SOAP
  - WSDL
  - UDDI
- Oracle BPM
- JBPM

¿Más?

- ¡MAS!
- Nos dejamos tecnologías y tendencias
  - JSF
  - J2ME
  - Entornos de desarrollo
  - Junit (tests)
  - JavaFX
  - Herramientas (ant, ..)
  - ...

Bibliografía

- **Sun Certified Enterprise Architect for J2EE Technology Study Guide**. Mark Cade, Simon Roberts. Sun Microsystems press (2002)
- **The J2EE Architect's Handbook**. Derek C. Ashmore.
- **Java Enterprise in a Nutshell**. O'Reilly. 2001
- **Mastering Enterprise Java Beans 3.0.** Rima Patel Sriganesh. Wiley Publishing. 2006

Tutoriales y Ejemplos

- Java en Castellano:
  www.programacion.com/java
- Sitio java de sun :

  http://java.sun.com
- Frameworks :
  http://javahispano.org/contenidos/es/comparativa_de_frameworks_web
- The Server Side
  http://www.theserverside.com