



**Universidad Autónoma del Estado de México  
Centro Universitario UAEM Valle de México**



# **Ingeniería en Computación**

**UNIDAD DE APRENDIZAJE:**

**LENGUAJE DE PROGRAMACIÓN VISUAL**

**TEMA:**

**MODELO VISTA CONTROLADOR (MVC) Y WEB  
ARCHIVES (WARS)**

**Elaboró: Dr. en C. Héctor Rafael Orozco Aguirre  
Agosto de 2019**



PROGRAMA DE ESTUDIO POR COMPETENCIAS  
**LENGUAJE DE PROGRAMACIÓN VISUAL**

**I. IDENTIFICACIÓN DEL CURSO**

Espacio Educativo: Facultad de Ingeniería						
Licenciatura: Ingeniería en Computación				Área de docencia: Programación e Ingeniería de Software		
Año de aprobación por el Consejo Universitario:						
Aprobación por los HH. Consejos Académico y de Gobierno		Fecha:		Programa elaborado por: Ing. Ma. Del Consuelo Mañón Salas		Programa revisado por: Integrantes de la Academia de Programación e Ingeniería de Software
				Fecha de elaboración : Enero de 2006		
Clave	Horas de teoría	Horas de práctica	Total de horas	Créditos	Tipo de curso	Núcleo de formación
L41091	2	1	3	5	Curso-Laboratorio	Integral
Unidad de Aprendizaje Antecedente Programación Estructurada				Unidad de Aprendizaje Consecuente Ninguna		
Programas educativos o espacios académicos en los que se imparte: Licenciatura en Ingeniería en Computación (Facultad. de Ingeniería, Centros Universitarios: Atlacomulco, Ecatepec, Texcoco, Valle de Chalco, Valle de México, Valle de Teotihuacán, Zumpango)						



DISTRIBUCIÓN DE LAS UNIDADES DE APRENDIZAJE OPTATIVAS

		PERIODO 1	PERIODO 2	PERIODO 3	PERIODO 4	PERIODO 5	PERIODO 6	PERIODO 7	PERIODO 8	PERIODO 9	PERIODO 10																												
O P T A T I V A S	Núcleo básico			<table border="1"> <tr><td>2</td></tr> <tr><td>0</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> </table> Análisis de Fourier	2	0	2	4	<table border="1"> <tr><td>2</td></tr> <tr><td>0</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> </table> Ética	2	0	2	4																										
		2																																					
		0																																					
		2																																					
		4																																					
2																																							
0																																							
2																																							
4																																							
		<table border="1"> <tr><td>3</td></tr> <tr><td>0</td></tr> <tr><td>3</td></tr> <tr><td>6</td></tr> </table> Cálculo numérico	3	0	3	6	<table border="1"> <tr><td>2</td></tr> <tr><td>0</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> </table> Sociedad e Ingeniería	2	0	2	4																												
3																																							
0																																							
3																																							
6																																							
2																																							
0																																							
2																																							
4																																							
		<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>3</td></tr> <tr><td>4</td></tr> </table> Simulación	1	2	3	4	<table border="1"> <tr><td>0</td></tr> <tr><td>2</td></tr> <tr><td>2</td></tr> <tr><td>2</td></tr> </table> Lectura y redacción	0	2	2	2																												
1																																							
2																																							
3																																							
4																																							
0																																							
2																																							
2																																							
2																																							
		<table border="1"> <tr><td>2</td></tr> <tr><td>1</td></tr> <tr><td>3</td></tr> <tr><td>5</td></tr> </table> Inferencia estadística	2	1	3	5	<table border="1"> <tr><td>3</td></tr> <tr><td>1</td></tr> <tr><td>4</td></tr> <tr><td>7</td></tr> </table> Química general	3	1	4	7																												
2																																							
1																																							
3																																							
5																																							
3																																							
1																																							
4																																							
7																																							
		<table border="1"> <tr><td>3</td></tr> <tr><td>0</td></tr> <tr><td>3</td></tr> <tr><td>6</td></tr> </table> Variable compleja	3	0	3	6																																	
3																																							
0																																							
3																																							
6																																							
	Línea 4: Desarrollo de software de aplicación				<table border="1"> <tr><td>2</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>6</td></tr> </table> Métricas de software	2	2	4	6	<table border="1"> <tr><td>2</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>6</td></tr> </table> Análisis de lenguajes de programación	2	2	4	6	<table border="1"> <tr><td>2</td></tr> <tr><td>1</td></tr> <tr><td>3</td></tr> <tr><td>5</td></tr> </table> Lenguaje de programación estructurado	2	1	3	5	<table border="1"> <tr><td>2</td></tr> <tr><td>1</td></tr> <tr><td>3</td></tr> <tr><td>5</td></tr> </table> Lenguaje de programación visual	2	1	3	5	<table border="1"> <tr><td>2</td></tr> <tr><td>1</td></tr> <tr><td>3</td></tr> <tr><td>5</td></tr> </table> Lenguaje de programación orientado a objetos	2	1	3	5	<table border="1"> <tr><td>2</td></tr> <tr><td>2</td></tr> <tr><td>4</td></tr> <tr><td>6</td></tr> </table> Minería de datos	2	2	4	6	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>3</td></tr> <tr><td>4</td></tr> </table> Desarrollo multimedia	1	2	3	4
2																																							
2																																							
4																																							
6																																							
2																																							
2																																							
4																																							
6																																							
2																																							
1																																							
3																																							
5																																							
2																																							
1																																							
3																																							
5																																							
2																																							
1																																							
3																																							
5																																							
2																																							
2																																							
4																																							
6																																							
1																																							
2																																							
3																																							
4																																							
	Línea 2: Redes y comunicaciones						<table border="1"> <tr><td>2</td></tr> <tr><td>1</td></tr> <tr><td>3</td></tr> <tr><td>5</td></tr> </table> Instalaciones y equipos	2	1	3	5	<table border="1"> <tr><td>2</td></tr> <tr><td>1</td></tr> <tr><td>3</td></tr> <tr><td>5</td></tr> </table> Tipos y configuraciones	2	1	3	5	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>3</td></tr> <tr><td>4</td></tr> </table> Auditoría de redes	1	2	3	4	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>3</td></tr> <tr><td>4</td></tr> </table> Servicios de internet	1	2	3	4	<table border="1"> <tr><td>1</td></tr> <tr><td>2</td></tr> <tr><td>3</td></tr> <tr><td>4</td></tr> </table> Interconexión de redes	1	2	3	4								
2																																							
1																																							
3																																							
5																																							
2																																							
1																																							
3																																							
5																																							
1																																							
2																																							
3																																							
4																																							
1																																							
2																																							
3																																							
4																																							
1																																							
2																																							
3																																							
4																																							

# Propósito de la Unidad de Aprendizaje

---

- El alumno conocerá la estructura de un lenguaje de programación orientado a objetos, el cual explotará como herramienta para el diseño y elaboración de páginas WEB.

# Contenido

---

- Modelo Vista Controlador (MVC):
  - Definición, flujo de control y ventajas
  - API en el J2SE y J2EE
  - Ejemplo

# Contenido

---

- Web-Archives (WARS):
  - Definición y estructura
  - Descriptor de despliegue de una aplicación Web
  - Elemento de contexto y creación
  - Ejemplo

# Guion explicativo

---

- Esta presentación tiene como fin dar a conocer a los alumnos los siguientes aspectos:
  - ¿Qué es el Modelo Vista Controlador (MVC)?
  - Uso del API J2SE para una aplicación de ejemplo
  - Creación de WARS en Java
  - Ejemplo de una aplicación Web con WARS

# Guion explicativo

---

- El contenido de esta presentación contiene elementos de competencia (temas) de interés contenidos en la Unidad de Aprendizaje de Lenguaje de Programación Visual, en específico de las Unidades de Competencia II y III.
- Las diapositivas deben explicarse en orden, y deben revisarse aproximadamente en 6 horas, además de realizar preguntas y dejar prácticas al grupo sobre el contenido mostrado.



# ¿Qué es el MVC?

---

- **El Modelo Vista Controlador (MVC)** es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos:
  - **Modelo:** representación específica del dominio de la información sobre la cual funciona una aplicación.
  - **Vista:** presenta un formato adecuado para interactuar, usualmente un elemento de interfaz usuario.
  - **Controlador:** interacciona con el usuario y con la aplicación, ya que interpreta la información que el usuario provee (usualmente acciones) y provoca cambios en el modelo y probablemente en la vista.

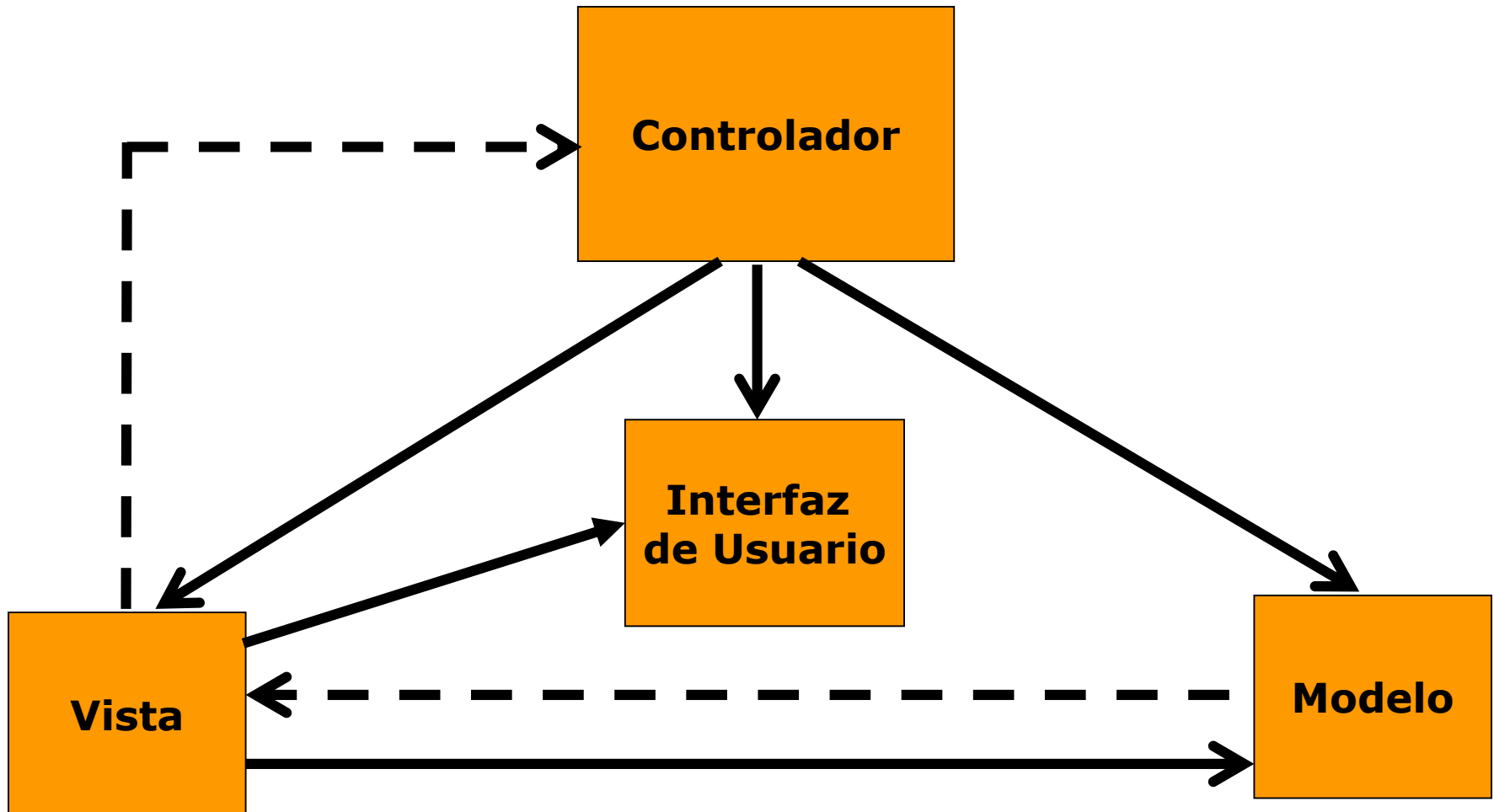
# ¿Qué es el MVC?

---

- ❑ La arquitectura MVC fue introducida como parte de la versión Smalltalk-80 del lenguaje de programación Smalltalk. Fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos.
- ❑ Sus características principales son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas.

# Flujo de Control del MVC

---



# Flujo de Control del MVC

---

- Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:
  - El usuario interactúa con la interfaz de usuario de alguna forma.
  - El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

# Flujo de Control del MVC

---

- El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario. Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

# Flujo de Control del MVC

---

- ❑ El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo.
- ❑ El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, el patrón de observador puede ser utilizado para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio.

# Flujo de Control del MVC

---

- Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista.
- El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. Un modelo puede tener diversas vistas, cada una con su correspondiente controlador.

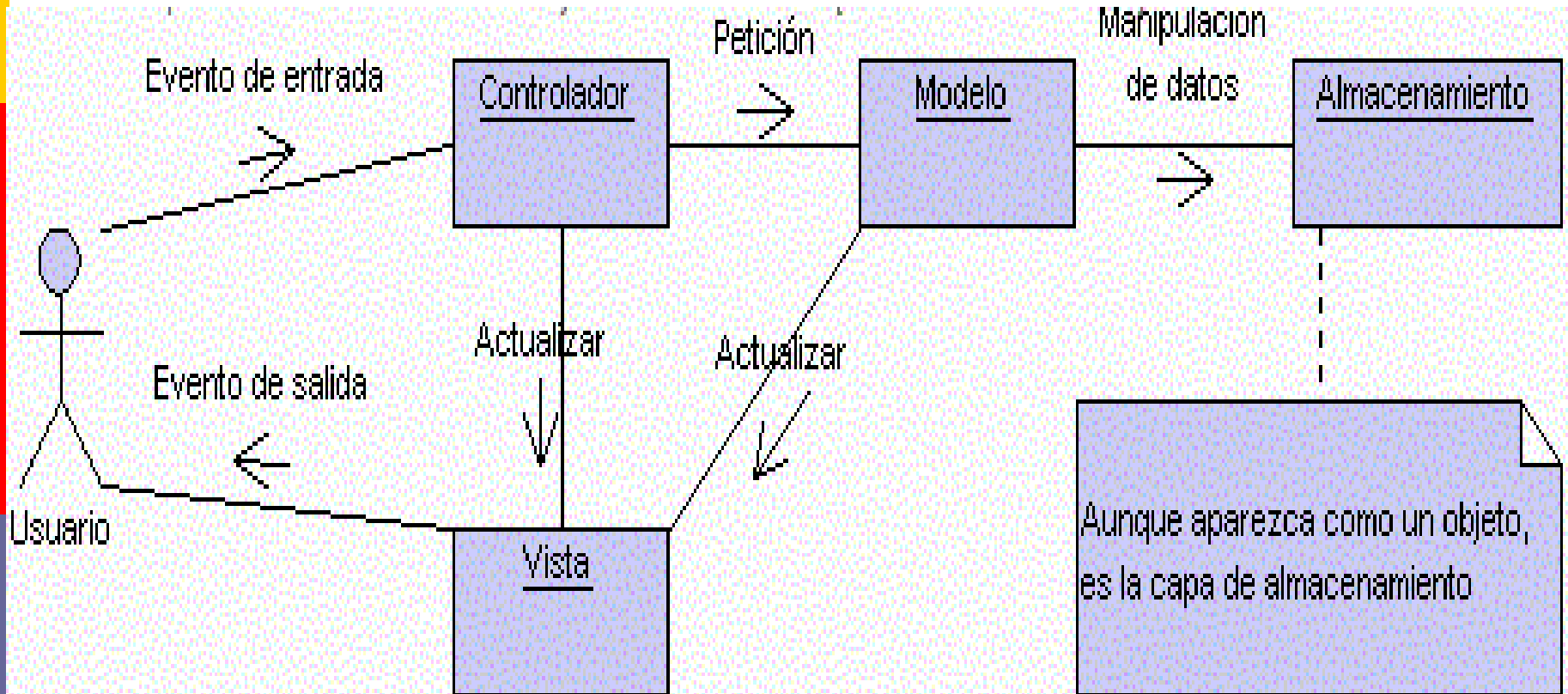
# Flujo de Control del MVC

---

- En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.
- La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

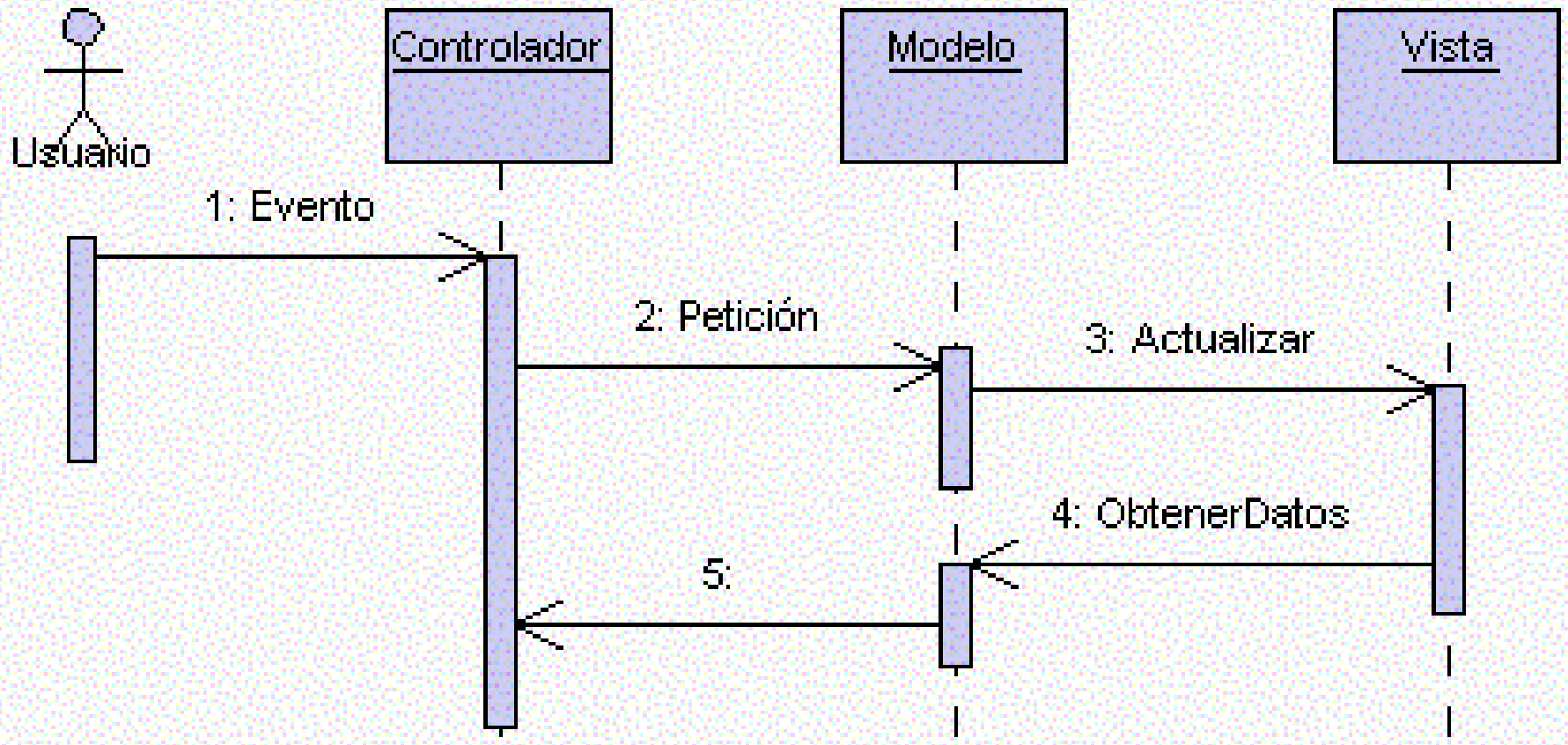


# Flujo de Control del MVC



MVC para el manejo de información de una Base de Datos.

# Flujo de Control del MVC



MVC para la navegación Web (modelo pasivo que no notifica cambios en los datos del servidor).

# Ventajas del MVC

---

- ❑ Clara separación entre los componentes de un programa; lo cual permite implementarlos por separado.
- ❑ El diseño queda más simple para los encargados del diseño gráfico y la implementación más entendible para los programadores.
- ❑ Reutilización de mucho más código, ya que las partes a implementar quedan mejor definidas y casi siempre suelen ser idénticas a las de otros proyectos.

# Ventajas del MVC

---

- Aislamiento entre las diferentes capas. Por ejemplo, si nuestra vista es una aplicación Web basada en JSP y queremos cambiar nuestro modelo, para que acceda a otra base de datos, la vista no se verá afectada por el cambio.

# Ventajas del MVC

---

- Utilizar los mismos objetos del modelo para diferentes vistas. Por ejemplo podemos hacer que la aplicación tenga dos tipos de presentación: una en HTML para visualizarla en un navegador y otra en XML para exportarla. El controlador podrá decidir qué vista presentar.

# Ventajas del MVC

---

- Hay un API muy bien definido; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.
- La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

# Ventajas del MVC

---

- Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unir las en tiempo de ejecución.
- Si uno de los Componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas.

# API del MVC en el J2SE

---

- El lenguaje de programación Java proporciona soporte para la arquitectura MVC mediante dos clases:
  - **Observer**: Es cualquier objeto que desee ser notificado cuando el estado de otro objeto sea alterado
  - **Observable**: Es cualquier objeto cuyo estado puede representar interés y sobre el cual otro objeto ha demostrado ese interés
- Estas dos clases se pueden utilizar para muchas más cosas que la implementación de la arquitectura MVC. Serán útiles en cualquier sistema en que se necesite que algunos objetos sean notificados cuando ocurran cambios en otros objetos.



# API del MVC en el J2SE

---

- El Modelo es un subtipo de **Observable** y la Vista es un subtipo de **Observer**. Estas dos clases manejan adecuadamente la función de notificación de cambios que necesita la arquitectura MVC.
- Proporcionan el mecanismo por el cual las Vistas pueden ser notificadas automáticamente de los cambios producidos en el Modelo. Referencias al objeto Modelo tanto en el Controlador como en la Vista permiten acceder a los datos de ese objeto Modelo.

# API del MVC en el J2SE

---

## □ **Observer**

- public void update( Observable obs, Object obj )

Llamada cuando se produce un cambio en el estado del objeto Observable

# API del MVC en el J2SE

---

## □ **Observable**

- `public void addObserver(Observer obs)`

Añade un observador a la lista interna de observadores

- `public void deleteObserver(Observer obs)`

Borra un observador de la lista interna de observadores

# API del MVC en el J2SE

---

- `public void deleteObservers()`

Borra todos los observadores de la lista interna

- `public int countObserver()`

Devuelve el número de observadores en la lista interna

- `protected void setChanged()`

Levanta la bandera interna que indica que el Observable ha cambiado de estado

# API del MVC en el J2SE

---

- `protected void clearChanged()`

Baja la bandera interna que indica que el Observable ha cambiado de estado

- `protected boolean hasChanged()`

Devuelve un valor booleano indicando si el Observable ha cambiado de estado

- `public void notifyObservers()`

Comprueba la bandera interna para ver si el Observable ha cambiado de estado y lo notifica a todos los observadores

# API del MVC en el J2SE

---

□ `public void notifyObservers( Object obj )`

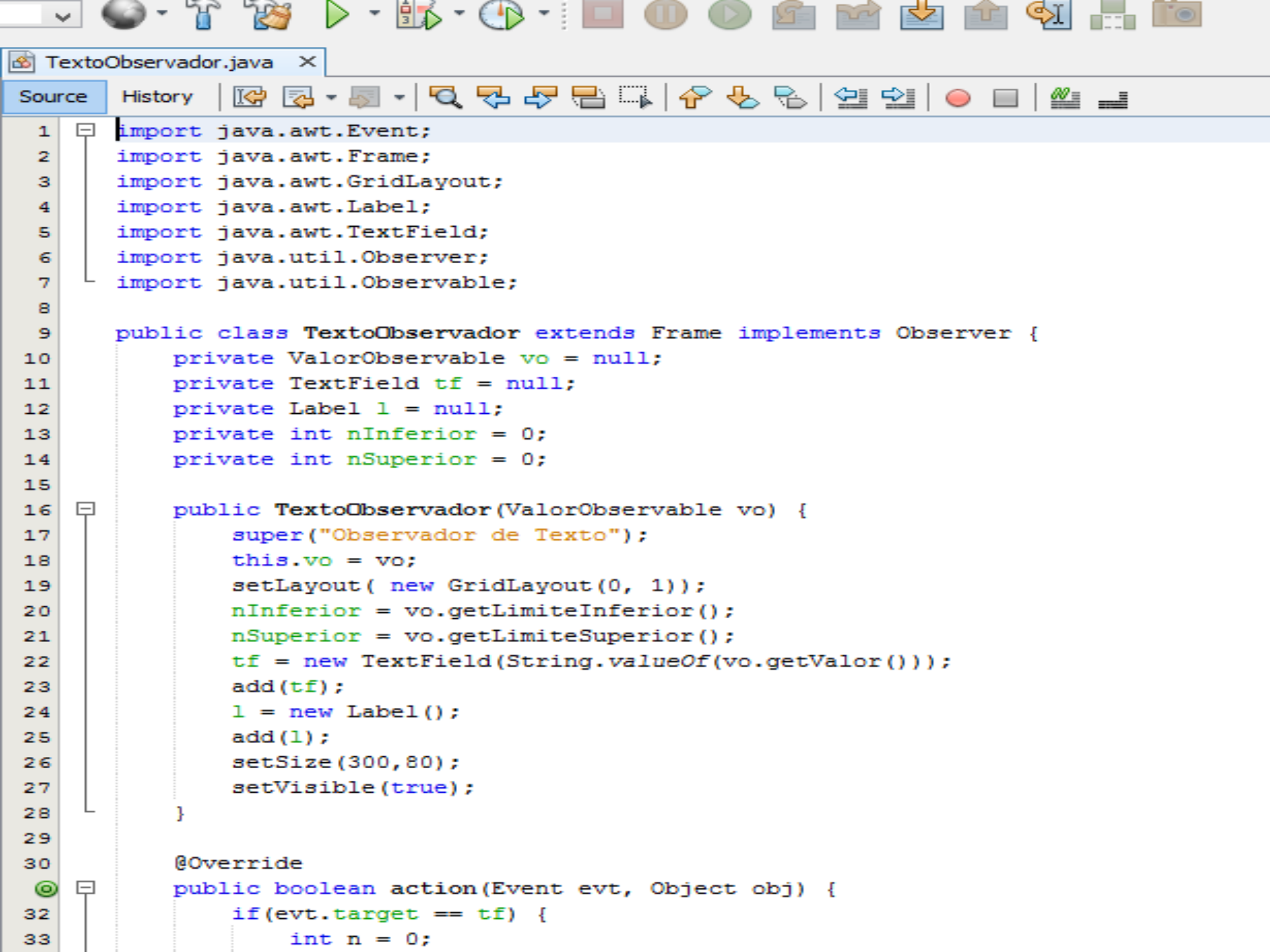
Comprueba la bandera interna para ver si el Observable ha cambiado de estado y lo notifica a todos los observadores. Les pasa el objeto especificado en la llamada para que lo usen los observadores en su método *notify()*.

---

# Ejemplo del MVC en J2SE

```
1  import java.util.Observable;
2
3  public class ValorObservable extends Observable {
4      private int nValor = 0;
5      private int nInferior = 0;
6      private int nSuperior = 0;
7
8      public ValorObservable(int nValor, int nInferior, int nSuperior) {
9          this.nValor = nValor;
10         this.nInferior = nInferior;
11         this.nSuperior = nSuperior;
12     }
13
14     public void setValor(int nValor) {
15         this.nValor = nValor;
16
17         setChanged();
18         notifyObservers();
19     }
20
21     public int getValor() {
22         return nValor;
23     }
24
25     public int getLimiteInferior() {
26         return nInferior;
27     }
28
29     public int getLimiteSuperior() {
30         return nSuperior;
31     }
32 }
```





TextoObservador.java

Source History

```
1 import java.awt.Event;
2 import java.awt.Frame;
3 import java.awt.GridLayout;
4 import java.awt.Label;
5 import java.awt.TextField;
6 import java.util.Observer;
7 import java.util.Observable;
8
9 public class TextoObservador extends Frame implements Observer {
10     private ValorObservable vo = null;
11     private TextField tf = null;
12     private Label l = null;
13     private int nInferior = 0;
14     private int nSuperior = 0;
15
16     public TextoObservador(ValorObservable vo) {
17         super("Observador de Texto");
18         this.vo = vo;
19         setLayout( new GridLayout(0, 1));
20         nInferior = vo.getLimiteInferior();
21         nSuperior = vo.getLimiteSuperior();
22         tf = new TextField(String.valueOf(vo.getValor()));
23         add(tf);
24         l = new Label();
25         add(l);
26         setSize(300,80);
27         setVisible(true);
28     }
29
30     @Override
31     public boolean action(Event evt, Object obj) {
32         if(evt.target == tf) {
33             int n = 0;
```



```
34         boolean bValido;
35         try {
36             n = Integer.parseInt(tf.getText());
37             bValido = true;
38         } catch(NumberFormatException nfe) {
39             bValido = false;
40             System.err.println("Error: " + nfe.getMessage());
41         }
42         if(n < nInferior || n > nSuperior)
43             bValido = false;
44         if(bValido) {
45             vo.setValor(n);
46             l.setText("");
47         }
48         else
49             l.setText("Valor no valido, intentelo de nuevo");
50         return true;
51     }
52     return false;
53 }
54
55 @Override
56 public boolean handleEvent(Event evt) {
57     if(evt.id == Event.WINDOW_DESTROY) {
58         vo.deleteObserver(this);
59         dispose();
60         return true;
61     }
62     return super.handleEvent(evt);
63 }
64
65 @Override
66 public void update(Observable obs, Object obj) {
```



```

48         else
49             l.setText("Valor no valido, intentelo de nuevo");
50         return true;
51     }
52     return false;
53 }

54
55 @Override
56 public boolean handleEvent(Event evt) {
57     if(evt.id == Event.WINDOW_DESTROY) {
58         vo.deleteObserver(this);
59         dispose();
60         return true;
61     }
62     return super.handleEvent(evt);
63 }

64
65 @Override
66 public void update(Observable obs, Object obj) {
67     if(obs == vo)
68         tf.setText(String.valueOf(vo.getValor()));
69 }
70 }
    
```

```
1 import java.awt.Event;
2 import java.awt.Frame;
3 import java.awt.GridLayout;
4 import java.awt.Scrollbar;
5 import java.util.Observer;
6 import java.util.Observable;
7
8 public class BarraObservador extends Frame implements Observer {
9     private ValorObservable vo = null;
10    private Scrollbar sb = null;
11
12    public BarraObservador(ValorObservable vo) {
13        super("Observador de Barra");
14        this.vo = vo;
15        setLayout(new GridLayout(0, 1));
16
17        sb = new Scrollbar(Scrollbar.HORIZONTAL, vo.getValor(), 10,
18            vo.getLimiteInferior(), vo.getLimiteSuperior());
19        add(sb);
20        setSize(300, 60);
21        setVisible(true);
22    }
23
24    @Override
25    public boolean handleEvent(Event evt) {
26        if(evt.id == Event.WINDOW_DESTROY) {
27            vo.deleteObserver(this);
28            dispose();
29            return true;
30        }
31        else if(evt.id == Event.SCROLL_LINE_UP) {
32            vo.setValor(sb.getValue());
33            return true;
34        }
35    }
36 }
```



```
28         dispose();
29         return true;
30     }
31     else if(evt.id == Event.SCROLL_LINE_UP) {
32         vo.setValor(sb.getValue());
33         return true;
34     }
35     else if(evt.id == Event.SCROLL_LINE_DOWN) {
36         vo.setValor(sb.getValue());
37         return true;
38     }
39     else if(evt.id == Event.SCROLL_PAGE_UP) {
40         vo.setValor(sb.getValue());
41         return true;
42     }
43     else if(evt.id == Event.SCROLL_PAGE_DOWN) {
44         vo.setValor(sb.getValue());
45         return true;
46     }
47     else if(evt.id == Event.SCROLL_ABSOLUTE) {
48         vo.setValor(sb.getValue());
49         return true;
50     }
51
52     return(super.handleEvent(evt));
53 }
54
55 @Override
56 public void update(Observable obs, Object obj) {
57     if(obs == vo)
58         sb.setValue(vo.getValor());
59 }
60 }
```



```
1 public class ControlValor {
2     public ControlValor() {
3         ValorObservable vo = new ValorObservable(100, 0, 10000);
4         TextoObservador to = new TextoObservador(vo);
5         BarraObservador bo = new BarraObservador(vo);
6         vo.addObserver(to);
7         vo.addObserver(bo);
8     }
9
10    public static void main(String[] args) {
11        new ControlValor();
12    }
13 }
```

ControlValor.java

Source History

```
1 public class ControlValor {
2     public ControlValor() {
3         ValorObservable vo = new ValorObservable(100, 0, 10000);
4         TextoObservable
5         BarraObservable
6         vo.addObserver
7         vo.addObserver
8     }
9
10    public static void
11        new ControlV
12    }
13 }
```

Observador de Barra

Observador de Texto

2701

ControlValor.java

Source History

```
1 public class ControlValor {
2     public ControlValor() {
3         ValorObservable vo = new ValorObservable(100, 0, 10000);
4         TextoObservable
5         BarraObservable
6         vo.addObserver
7         vo.addObserver
8     }
9
10    public static void
11        new ControlV
12    }
13 }
```

Observador de Barra

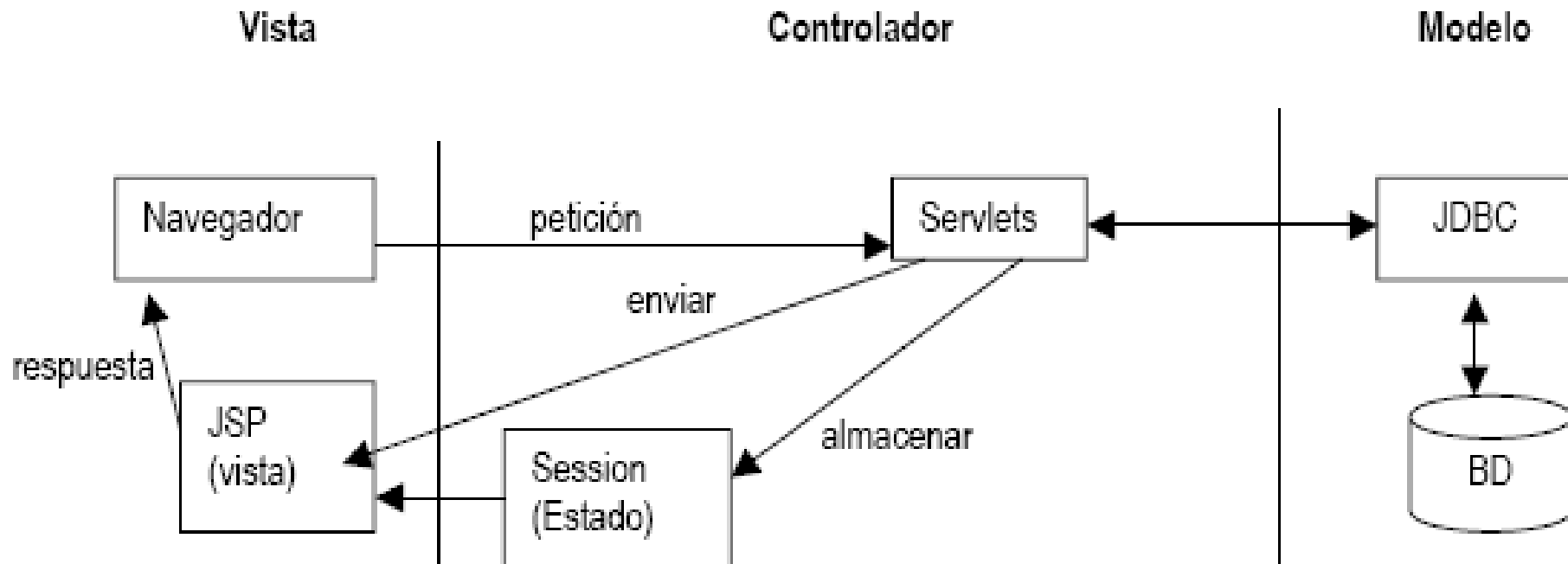
Observador de Texto

80000

Valor no valido, intentelo de nuevo

# MVC y J2EE

- Las aplicaciones Web pueden diseñarse e implementarse mediante el MVC, utilizando HTML y JSP para las vistas, servlets como controladores y JDBC para el modelo.





# ¿Qué son los WARS?

---

- Toda aplicación Web residente en un servidor de aplicaciones Web se encuentra agrupada en Web-Archives (WARS), la estructura de un WAR es definida por *Sun*, la cual debe ser implementada en cualquier producto de Servlet Engine (Web-Container).

# ¿Qué son los WAR'S?

---

- Cuando se instala un servidor de aplicaciones Web, éste ya contiene varios WAR'S, los cuales contienen ejemplos y documentación, estos WAR'S se encuentran alojados en el directorio `$SERVER_HOME/webapps`, donde `$SERVER_HOME` es el directorio raíz del servidor de aplicaciones Web.
- Si se desciende a este directorio se observará que además de los archivos `*.war` existe además un directorio con el mismo nombre del WAR, esto es, si existen WAR's llamados `ROOT.war` y `examples.war` también existen directorios llamados `ROOT` y `examples`.
- Estos directorios son los que precisamente contienen la estructura del archivo WAR, esto es, el archivo WAR en sí no es legible sino que tiene que ser expandido/descomprimido para poder ser leído.

# ¿Qué son los WARS?

---

- Con el lanzamiento de la especificación 2.2 de Java Servlet, el concepto de aplicación Web fue introducido. Según esta especificación, una aplicación Web es una colección de servlets, páginas HTML, clases y otros recursos que puedan ser utilizados y ejecutados en múltiples contenedores de múltiples vendedores”.
- Una aplicación Web para nosotros sera cualquier cosa que resida en la capa Web de una aplicación.

# ¿Qué son los WARS?

---

- Una de las principales características de una aplicación Web es la relación con el **ServletContext**. Cada aplicación Web tiene uno y sólo un ServletContext. Esta relación está controlada por el contenedor de servlets y garantiza que la aplicación Web no chocara cuando se almacenan objetos en el ServletContext. Los siguientes elementos pueden existir en una aplicación Web:
  - Servlets
  - JavaServer Pages
  - Clases útiles
  - Documentos estáticos incluyendo XHTML, imágenes, etc.
  - Clases del lado del cliente
  - Meta información que describe la aplicación Web.
  - Entre otros.

# Estructura de un directorio WAR

---

- La estructura de un directorio WAR es la siguiente:
  - **/:** Este directorio base contiene los elementos que comúnmente son utilizados en un sitio: **\*.html \*.jsp \*.css** y otros elementos.
  - **/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 (clases hechas por el programador) que serán empleadas en los archivos JSP y en los Servlets.
  - **/WEB-INF/lib/ :** Contiene los JAR'S que serán utilizados por su aplicación.

# Descriptor de despliegue de una aplicación Web

---

- En el corazón de toda aplicación Web hay un descriptor de despliegue. El descriptor de despliegue es un archivo XML llamado **web.xml**, este se encuentra en el directorio `/<SERVER_HOME>/WAR/WEB-INF/` y describe la información de configuración para toda la aplicación Web.
- Por ejemplo, supongamos que la información que se contiene en el descriptor de despliegue incluye los elementos siguientes:

# Descriptor de despliegue de una aplicación Web

---

- Parámetros de inicialización del ServletContext.
- Contenido localizado.
- Configuración de la Sesión.
- Definiciones de Servlets / JSP's.
- Mapeos Servlets / JSP's.
- Mapeos de tipo Mime.
- Listado del archivo de bienvenida.
- Paginas de error.
- Seguridad.

# Descriptor de despliegue de una aplicación Web

---

```
<web-app>
  <display-name>Nombre WebApp</display-name>
  <session-timeout>30</session-timeout>
  <servlet>
    <servlet-name>TestServlet</servlet-name>
    <servlet-class>TestServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
    <init-param>
      <param-name>name</param-name>
      <param-value>value</param-value>
    </init-param>
  </servlet>
</web-app>
```



# El elemento de contexto

---

- El elemento de Contexto (**Context**) representa una aplicación Web, la cual se ejecuta en un host virtual particular. Cada aplicación Web está basada en un WAR. Podemos definir el contexto cómo deseemos.
- El elemento de contexto que al final nos interesa definir lo haremos en el archivo que se encuentra en la ruta `$SERVER_HOME/conf/server.xml`.
- Por ejemplo:

```
<Context path="/CONTEXT_NAME" docBase="CONTEXT_NAME"  
  debug="0" crossContext="true" reloadable="true" />
```

# Creación de un WAR

---

- **Ejecutar el siguiente comando en consola:**

```
jar cvf nombre.war directorio_de_aplicacion_war
```

---

# Ejemplo del MVC con WARS en J2EE



Navigator

Projects X Files Services

- Luchadores
  - Web Pages
    - META-INF
      - context.xml
    - WEB-INF
      - web.xml
    - index.html
  - Source Packages
    - <default package >
      - AccesoDatos.java
      - Luchador.java
  - Libraries
    - MySQL JDBC Driver - mysql-connector-java-5.1.23-bin.jar
    - JDK 1.8 (Default)
    - Apache Tomcat 8.0.3.0
  - Configuration Files
    - MANIFEST.MF
    - context.xml
    - web-fragment.xml
    - web-fragment.xml
    - web.xml



```
1 <!DOCTYPE html>
2 <html>
3   <head><title>Fanometro de Luchadores</title></head>
4   <body>
5     <center><H1>Mejores Luchadores Mexicanos</H1></center>
6     <p align="center"><font color="#002424" size="7">
7       <u>VOTE POR EL MEJOR LUCHADOR</u>
8     </font></p>
9     <form action="Luchador" method="post">
10      <p align="left">
11        Nombre del Visitante:
12          <input type="text" size="20" name="txtNombre">
13        E-Mail:
14          <input type="text" size="20" name="txtMail">
15      </p>
16      <p align="left">
17        <input type="radio" name="R1" value="Mistico">Mistico
18      </p>
19      <p align="left">
20        <input type="radio" name="R1" value="Rey Misterio">Rey Misterio
21      </p>
22      <p align="left">
23        <input type="radio" name="R1" value="El Santo">El Santo
24      </p>
25      <p align="left">
26        <input type="radio" name="R1" value="Blue Demon">Blue Demon
27      </p>
28      <p align="left">
29        <input type="radio" name="R1" value="Mil Mascaras">Mil Mascaras
30      </p>
31      <p align="left">
32        <input type="radio" name="R1" value="Huracan Ramirez">Huracan Ramirez
33    </p>
```



```
29 <input type="radio" name="R1" value="Mi Mascota" />Mi Mascota
30
31 <p align="left">
32     <input type="radio" name="R1" value="Huracan Ramirez">Huracan Ramirez
33 </p>
34 <p align="left">
35     <input type="radio" name="R1" value="Perro Aguayo">Perro Aguayo
36 </p>
37 <p align="left">
38     <input type="radio" name="R1" value="Rayo de Jalisco">Rayo de Jalisco
39 </p>
40 <p align="left">
41     <input type="radio" name="R1" value="Otros">Otros
42     <input type="text" size="20" name="txtOtros">
43 </p>
44 <p align="left">
45     <input type="submit" name="B1" value="Votar">
46     <input type="reset" name="B2" value="Reset">
47 </p>
48 </form>
49 </body>
50 </html>
```



```
1  import java.io.IOException;
2  import java.io.PrintWriter;
3  import javax.servlet.ServletConfig;
4  import javax.servlet.ServletException;
5  import javax.servlet.http.HttpServlet;
6  import javax.servlet.http.HttpServletRequest;
7  import javax.servlet.http.HttpServletResponse;
8
9  public class Luchador extends HttpServlet {
10     private AccesoDatos bd;
11
12     @Override
13     public void init(ServletConfig cfg) throws ServletException {
14         bd= new AccesoDatos();
15         bd.abrirConexion();
16     }
17
18     @Override
19     public void doPost(HttpServletRequest req, HttpServletResponse res)
20         throws ServletException, IOException {
21         String nombreP=(String) req.getParameter("txtNombre");
22         String nombre=(String) req.getParameter("R1");
23         if (nombre.equals("Otros"))
24             nombre=(String) req.getParameter("txtOtros");
25         if (bd.existeLuchador(nombre))
26             bd.actualizarLuchador(nombre);
27         else
28             bd.insertarLuchador(nombre);
29         req.getSession(true).putValue("nombreCliente", nombreP);
30         generaEstadisticas(req, res);
31     }
32 }
```
























```
30     generaEstadisticas(req, res);
31 }
32
33 @Override
34 public void doGet(HttpServletRequest req, HttpServletResponse res)
35     throws ServletException, IOException {
36     try {
37         doPost(req, res);
38     } catch (Exception e) {
39         System.err.println("Error: " + e.getMessage());
40     }
41 }
42
43 private void generaEstadisticas(HttpServletRequest req, HttpServletResponse resp)
44     throws ServletException, IOException {
45     String nombre, nombreP, votos;
46     PrintWriter out=null;
47     try
48     {
49         out=resp.getWriter();
50     } catch (IOException io) {
51         System.out.println("Se ha producido una excepcion");
52     }
53     resp.setContentType("text/html");
54     out.println("<html>");
55     out.println("<head>");
56     out.println("<title>Votos de cada luchador</title>");
57     out.println("</head>");
58     out.println("<body>");
59     out.println("<font size=10><h1>Tabla de Votación</h1>");
60     out.println("<table border=1>");
61     out.println("<tr><td><b>Luchador</b></td><td><b>Votos</b></td></tr>");
```























```
58     out.println("<body>");
59     out.println("<font size=10><h1>Tabla de Votación</h1>");
60     out.println("<table border=1>");
61     out.println("<tr><td><b>Luchador</b></td><td><b>Votos</b></td></tr>");
62     nombreP =(String) req.getSession(true).getValue("nombreCliente");
63     bd.consultarLuchadores();
64     while (bd.siguiente()) {
65         nombre=bd.getCampo("Nombre");
66         votos=bd.getCampo("Votos");
67         out.println("<tr><td>" + nombre + "</td><td>" + votos + "</td></tr>");
68     }
69     bd.cerrarConsulta();
70     out.println("</table>");
71     out.println("<h3>Muchas gracias " + nombreP + " por su visita</h3>");
72     out.println("</font>");
73     out.println("</body>");
74     out.println("</html>");
75     out.flush();
76     out.close();
77 }
78
79 @Override
80 public void destroy() {
81     bd.cerrarConexion();
82     super.destroy();
83 }
84 }
```

```
1  import java.sql.Connection;
2  import java.sql.DriverManager;
3  import java.sql.ResultSet;
4  import java.sql.SQLException;
5  import java.sql.Statement;
6
7  public class AccesoDatos {
8      private Connection con;
9      private Statement set;
10     private ResultSet rs;
11
12     public void abrirConexion() {
13         String urlBD = "jdbc:mysql://localhost:3306/luchadores";
14         try
15         {
16             Class.forName("com.mysql.jdbc.Driver").newInstance();
17             con = DriverManager.getConnection(urlBD,
18                 "root", "root");
19             System.out.println("Se ha conectado");
20         } catch(ClassNotFoundException | InstantiationException
21             | IllegalAccessException | SQLException e) {
22             System.err.println("Error: " + e.getMessage());
23         }
24     }
25
26     public boolean existeLuchador(String nombre) {
27         boolean existe = false;
28         String cad;
29         try {
30             set = con.createStatement();
31             rs = set.executeQuery("SELECT * FROM Luchadores");
32             while (rs.next()) {
33                 cad = rs.getString("nombre");
```

```
Source History |                       
```

```
32     while (rs.next()) {
33         cad = rs.getString("nombre");
34         cad = cad.trim();
35         if (cad.equals(nombre.trim()))
36             existe = true;
37     }
38     rs.close();
39     set.close();
40 } catch(Exception e) {
41     System.err.println("Error: " + e.getMessage());
42 }
43 return existe;
44 }
45
46 public void actualizarLuchador(String nombre) {
47     try {
48         set = con.createStatement();
49         set.executeUpdate("UPDATE Luchadores SET votos = votos + 1"
50             + " WHERE nombre " + " LIKE '%" + nombre + "%'");
51         rs.close();
52         set.close();
53     } catch(Exception e) {
54         System.err.println("Error: " + e.getMessage());
55     }
56 }
57
58 public void insertarLuchador(String nombre) {
59     try {
60         set = con.createStatement();
61         set.executeUpdate("INSERT INTO Luchadores "
62             + " (nombre, votos) VALUES ('" + nombre + "',1)");
63         rs.close();
64         set.close();
65     } catch(Exception e) {
```

```
65     } catch(Exception e) {
66         System.out.println("No se puede insertar en la Base de Datos");
67     }
68 }
69
70 public void consultarLuchadores() {
71     try {
72         set = con.createStatement();
73         rs = set.executeQuery("SELECT * FROM Luchadores");
74     } catch(Exception e) {
75         System.err.println("Error: " + e.getMessage());
76     }
77 }
78
79 public boolean siguiente() {
80     try {
81         return (rs.next());
82     } catch(Exception e) {
83         System.err.println("Error: " + e.getMessage());
84         return false;
85     }
86 }
87
88 public String getCampo(String nombre) {
89     try {
90         return rs.getString(nombre);
91     } catch(Exception e) {
92         System.err.println("Error: " + e.getMessage());
93         return "";
94     }
95 }
96
```

```
Source History |                  
```

```
88     public String getCampo(String nombre) {
89         try {
90             return rs.getString(nombre);
91         } catch (Exception e) {
92             System.err.println("Error: " + e.getMessage());
93             return "";
94         }
95     }
96
97     public void cerrarConsulta() {
98         try {
99             rs.close();
100            set.close();
101        } catch (Exception e) {
102            System.err.println("Error: " + e.getMessage());
103        }
104    }
105
106     public void cerrarConexion() {
107         try {
108             con.close();
109         } catch (Exception e) {
110             System.err.println("Error: " + e.getMessage());
111         }
112     }
113 }
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5         http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
6     <servlet>
7         <servlet-name>Luchador</servlet-name>
8         <servlet-class>Luchador</servlet-class>
9     </servlet>
10    <servlet-mapping>
11        <servlet-name>Luchador</servlet-name>
12        <url-pattern>/Luchador</url-pattern>
13    </servlet-mapping>
14    <session-config>
15        <session-timeout>
16            30
17        </session-timeout>
18    </session-config>
19 </web-app>
```



Limit to 100 rows

1

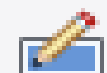
```
SELECT * FROM luchadores.luchadores;
```

Result Grid



Filter Rows:

Edit:



	nombre	votos
▶	Blue Demos	350
	El Santo	400
	Huracan Ramirez	200
	Mil Mascaras	100
	Mistico	5
	Otros	4
	Perro Aguayo	297
	Rayo de Jalisco	336
	Rey Misterio	2



## Mejores Luchadores Mexicanos

# VOTE POR EL MEJOR LUCHADOR

Nombre del Visitante:

E-Mail:

- Místico
- Rey Misterio
- El Santo
- Blue Demon
- Mil Mascaras
- Huracán Ramírez
- Perro Aguayo
- Rayo de Jalisco

Otros





# Tabla de Votación

Luchador	Votos
Blue Demos	350
El Santo	401
Huracan Ramirez	200
Mil Mascaras	100
Mistico	5
Otros	4
Perro Aguayo	297
Rayo de Jalisco	336
Rey Misterio	2

**Muchas gracias Héctor Orozco  
Aguirre por su visita**



Limit to 100 rows

1 •

```
SELECT * FROM luchadores.luchadores;
```



Result Grid



Filter Rows:

Edit:



	nombre	votos
	Blue Demos	350
	El Santo	401
	Huracan Ramirez	200
	Mil Mascaras	100
	Mistico	5
	Otros	4
	Perro Aguayo	297
	Rayo de Jalisco	336
	Rey Misterio	2

# Referencias

---

- Wolf, D., & Henley, A. J. (2017). *Java EE Web Application Primer: Building Bullhorn: A Messaging App with JSP, Servlets, JavaScript, Bootstrap and Oracle*. Apress.
- Kurniawan, B. (2015). *Servlet & JSP: A Tutorial*. Brainy Software Inc.
- Sierra, F. J. C. (2015). *JAVA. Interfaces gráficas y aplicaciones para Internet* (Vol. 14). Grupo Editorial RA-MA.
- Morales, M. S. (2012). *Manual de Desarrollo Web basado en ejercicios y supuestos prácticos*. Lulu. com.
- Groussard, T. (2010). *Java enterprise edition: desarrollo de aplicaciones web con JEE 6*. Ediciones Eni.