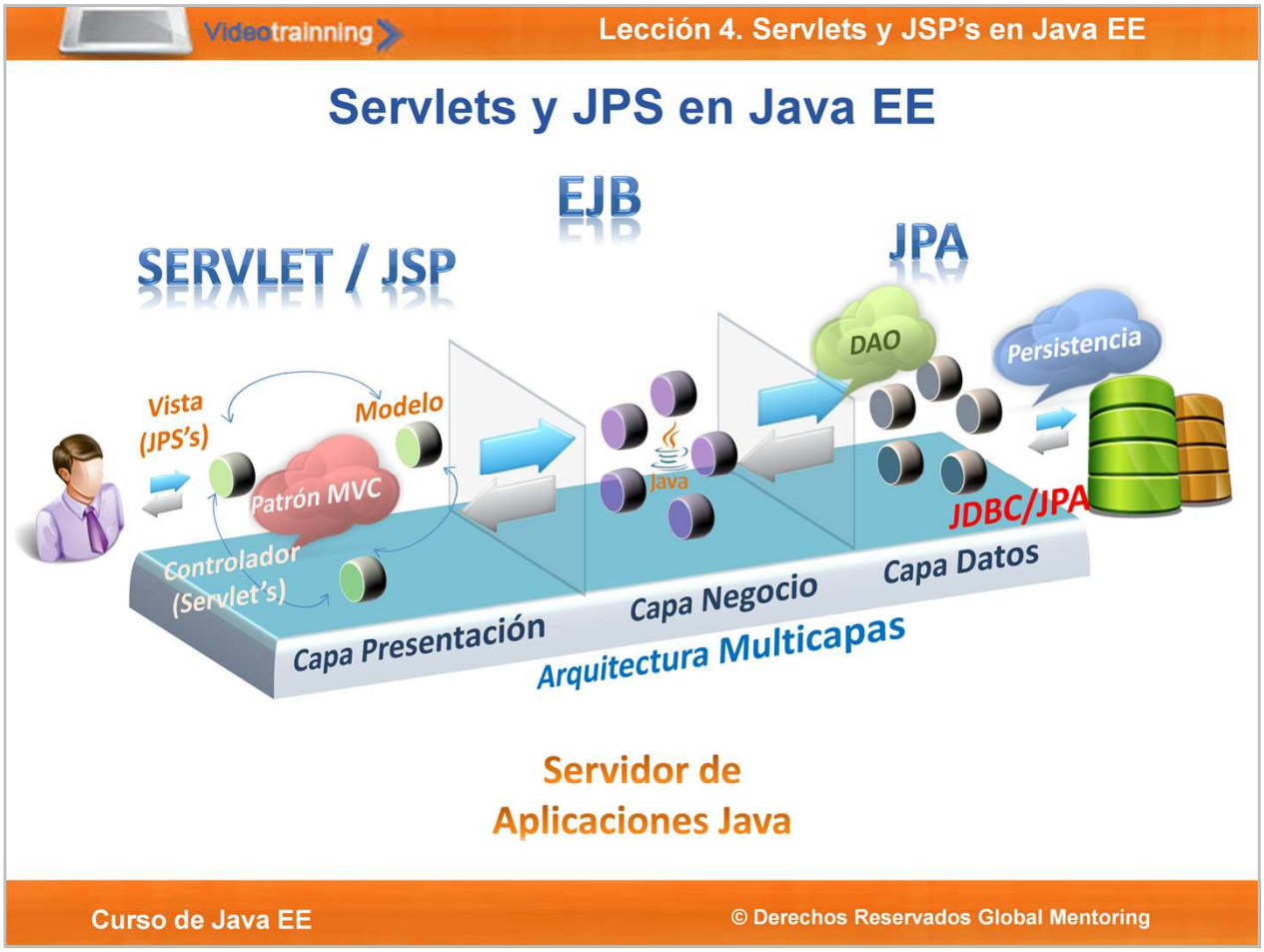

Videotrainning
Lección 4. Servlets y JSP's en Java EE
Global Mentoring
Java EE
Lección 4
Servlets y JPS's en Java EE
www.globalmentoring.com.mx
© Derechos Reservados Global Mentoring



Las razones por las cuales creamos arquitecturas empresariales multicapas son muchas, a continuación mencionamos algunas:

- **Abstracción y Encapsulamiento**: Las capas abstraen los detalles del trabajo que realiza cada capa. Así, cada una será responsable de ciertos procesos, sin tener que involucrar a las demás capas para realizar sus tareas fundamentales.
- **Funcionalidad claramente definida:** Cada capa define sus tareas, permitiendo delegar adecuadamente el trabajo entre el grupo de programadores.
- **Alta Cohesión:** Esto significa que una capa realiza solamente la tarea para la cual fue creada, y las demás funciones las delega o se apoya de las demás capas.
- **Bajo Acoplamiento**: Debido a que cada capa tiene aplicado el concepto de abstracción, solamente existen los puntos de comunicación mínimo necesarios entre las distintas capas, permitiendo acoplarse y desacoplarse sin problemas.
- **Reutilizable**: Como resultado del bajo acoplamiento y las demás características, es posible reutilizar capas completas para distintos proyectos.

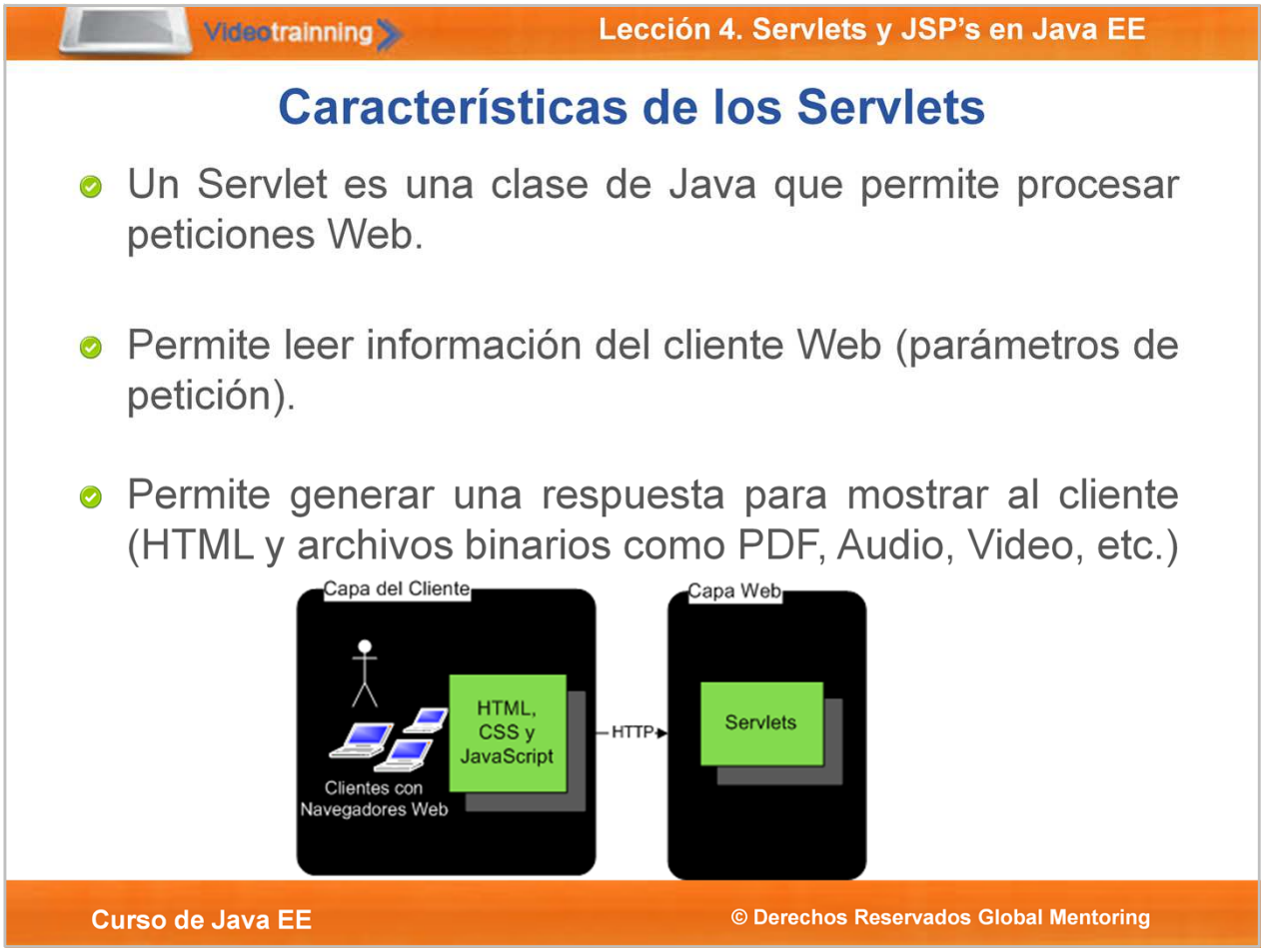
Además, podemos tener los siguientes beneficios al utilizar capas en una arquitectura empresarial:

- **Aislamiento**: Permite reducir el impacto para cambiar la tecnología, ya que podemos ir paulatinamente migrando una capa a una nueva tecnología, sin impactar demasiado en el sistema total.
- **Rendimiento**: Si hay necesidad de distribuir las capas entre diferentes servidores es posible realizarlo, con el objetivo de agregar conceptos como escalabilidad, fiabilidad, tolerancia a fallos y en conclusión mejorar el rendimiento completo del sistema.
- **Mejoras en pruebas**: Entre más claras sean las responsabilidades de cada capa, podemos realizar pruebas de y descartar fallas de software asilando el problema de manera muy precisa.

En la figura podemos observar el rol de las tecnologías Servlets y JSPs en una arquitectura Java Empresarial. Estas tecnologías aplican directamente en la capa de presentación, la cual se encarga de tareas tales como:

- Generación de Código de Presentación, con tecnologías como HTML, CSS y JavaScript.
- Procesamiento de peticiones HTTP provenientes del cliente a través de formularios HTML.
- Validación de parámetros recibidos en la petición HTTP.
- Recuperación de la información del Modelo, apoyándose de la Capa de Servicio (EJB's) para ello.
- Procesamiento de la respuesta y selección de la vista a mostrar al cliente. Entre varias tareas más.

En esta lección revisaremos cómo integrar las tecnologías de los Sevlets y JPSs a una arquitectura Java EE.



Un Servlet es el componente más básico de la tecnología Java para el desarrollo de aplicaciones Web. Incluso, un JPS, al ser compilado se convierte en un Servlet.

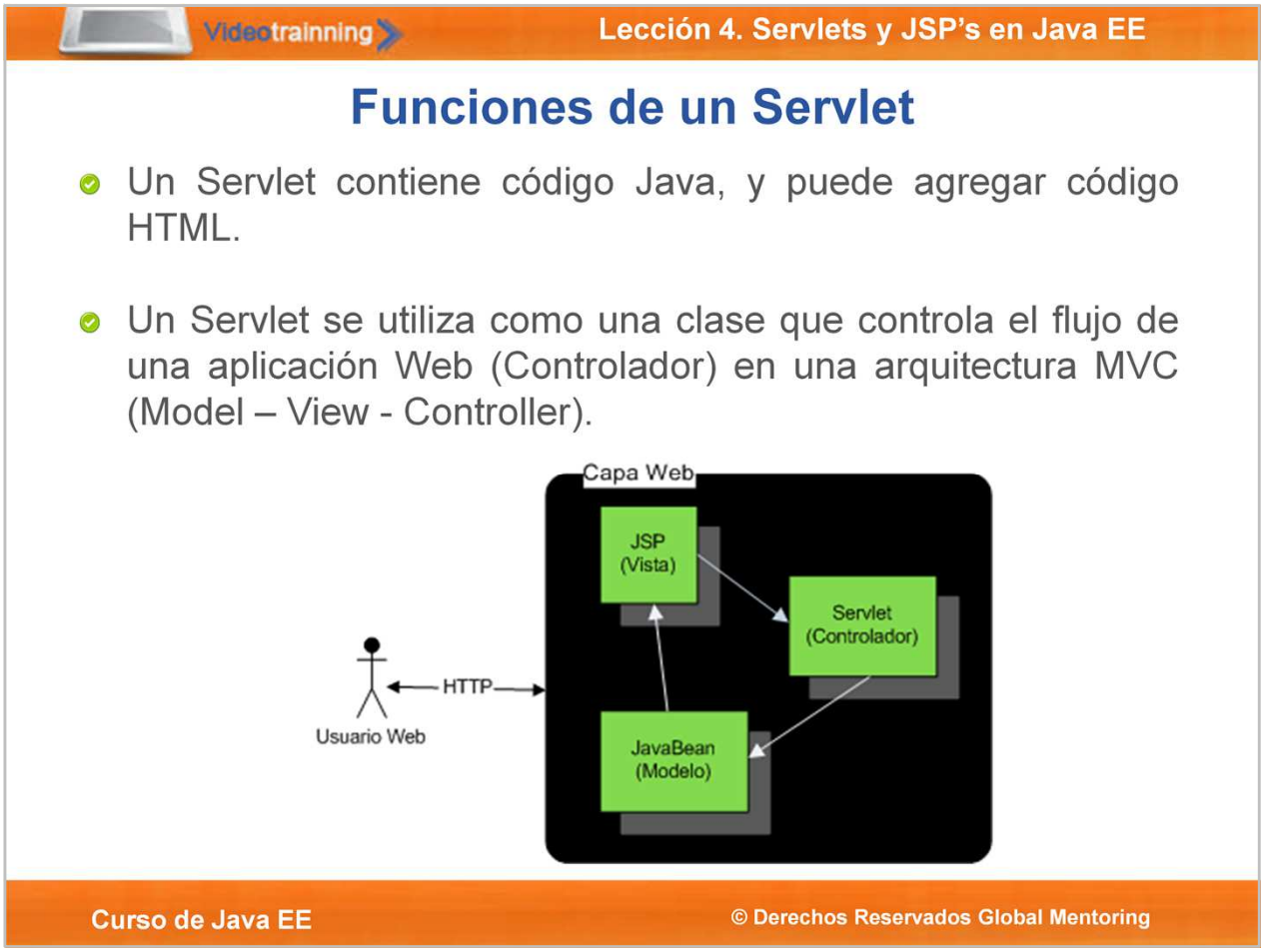
Esta clase Java permite procesar una petición basada en el protocolo HTTP (Hipertext Transfer Protocol). Este es el protocolo que revolucionó en su momento el uso de Internet, ya que a través de él es posible solicitar páginas Web a Servidores de distintas tecnologías como Java, PHP, .Net, entre muchas tecnologías más.

Los Servlets, son clases Java que tienen un ciclo de vida bien definido, y que son administrador por un contenedor de aplicaciones Web Java, por lo tanto, no basta con un JDK para ser ejecutados, sino que debemos tener como mínimo un servidor Tomcat o, si se requiere de integración de más tecnologías, un servidor completo de aplicaciones Java como Glassfish, WebSphere, Weblogic o JBoss.

Como podemos observar en la figura, una clase Servlet permite procesar la petición del cliente y puede generar una respuesta de manera dinámica. El contenido de respuesta puede ser código HTML o archivos binarios, tales como PDF, Audio, Video, etc)

Los Servlets están orientados a procesar la petición del cliente, y por lo tanto el código que contienen es código Java, pudiendo incrustar código HTML para generar la respuesta al cliente, sin embargo, precisamente para evitar esto último es que vamos a utilizar la tecnología de los JSPs, los cuales sí están orientados a tener principalmente código HTML e incrustar código Java.

Hace ya más de una década que los Servlets aparecieron. Al día de hoy es la versión 3.0 la que está liberada, y se ha renovado para soportar de manera nativa varias características muy interesantes que revisaremos más adelante.



En la figura podemos observar las responsabilidades de un Servlet y el rol que juega en el patrón de diseño MVC. El Servlet es el controlador de la aplicación y por lo tanto procesa la petición del cliente, recupera la información del modelo y envía de vuelta la respuesta al cliente, pudiéndose apoyar de los JPS's para esta última tarea, o también lo puede hacer directamente el Servlet (aunque no es recomendable).

Pasos generales que ejecuta un Servlet Controlador:

a) Procesamos y validamos los parámetros (si aplica)

```
int idPersona = request.getParameter("idPersona");
```

b) Realizamos la lógica de presentación almacenando el resultado en JavaBeans

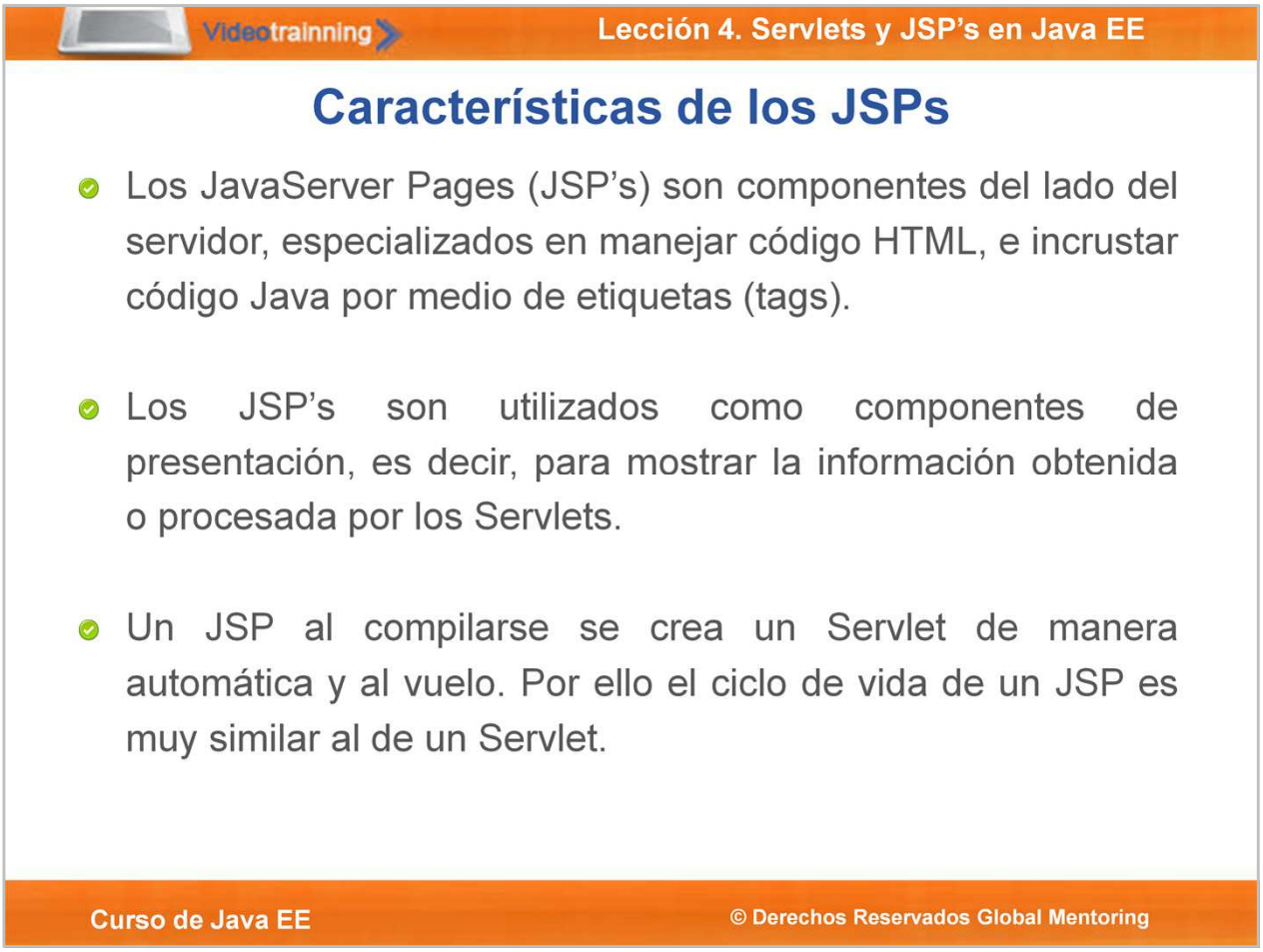
```
Persona persona = personaService.encontrarPersonaPorId( idPersona );
```

c) Compartimos el objeto bean a utilizar en algún alcance (scope)

```
request.setAttribute("persona", persona );
```

d) Redireccionamos al JSP seleccionado

```
RequestDispatcher dispatcher =request.getRequestDispatcher("listar.jsp");
dispatcher.forward( request, response );
```



Los JSPs son componentes Java del lado del servidor, los cuales se encargan de desplegar la información de respuesta hacia el cliente, generando de manera dinámica el contenido HTML necesario. Estos son algunos de los beneficios de utilizar los JSP's:

- Nos podemos enfocar en escribir código HTML, haciendo más fácil el mantenimiento de la capa de presentación
- Podemos utilizar herramientas de diseño para crear visualmente las páginas HTML e incrustar las etiquetas dinámicas de los JSP's
- Separa el código de presentación del código de Java
- Cada miembro del equipo de desarrollo se puede enfocar en distintas tareas, al separar las responsabilidades.

La tecnología de los JSP's se apoya a su vez de otras tecnologías para simplificar el despliegue de información.

- Expression Language: El lenguaje de expresión (EL) simplifica el acceso a la información compartida por los Servlets, según el modelo MVC, y con ello el JPS se enfoca en su tarea primordial, la cual es procesar la información para enviar la respuesta al cliente.
- JSTL (JSP Standard Tag Library): La librería JSTL es un conjunto de etiquetas que podemos utilizar en los JSPs, y nos permiten simplificar tareas en las que comúnmente tendríamos que agregar código Java, permitiendo tener un código más limpio y mantenible.

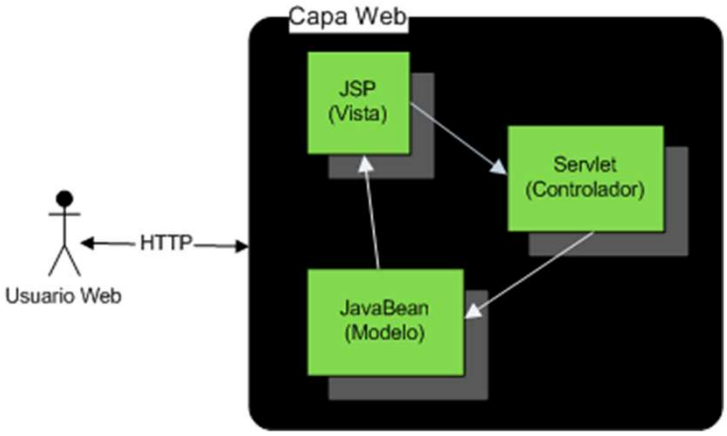
Estas tecnologías las utilizaremos en el ejercicio de integración de tecnologías empresariales que realizaremos más adelante.

Videotrainning

Lección 4. Servlets y JSP's en Java EE

Funciones de un JSP

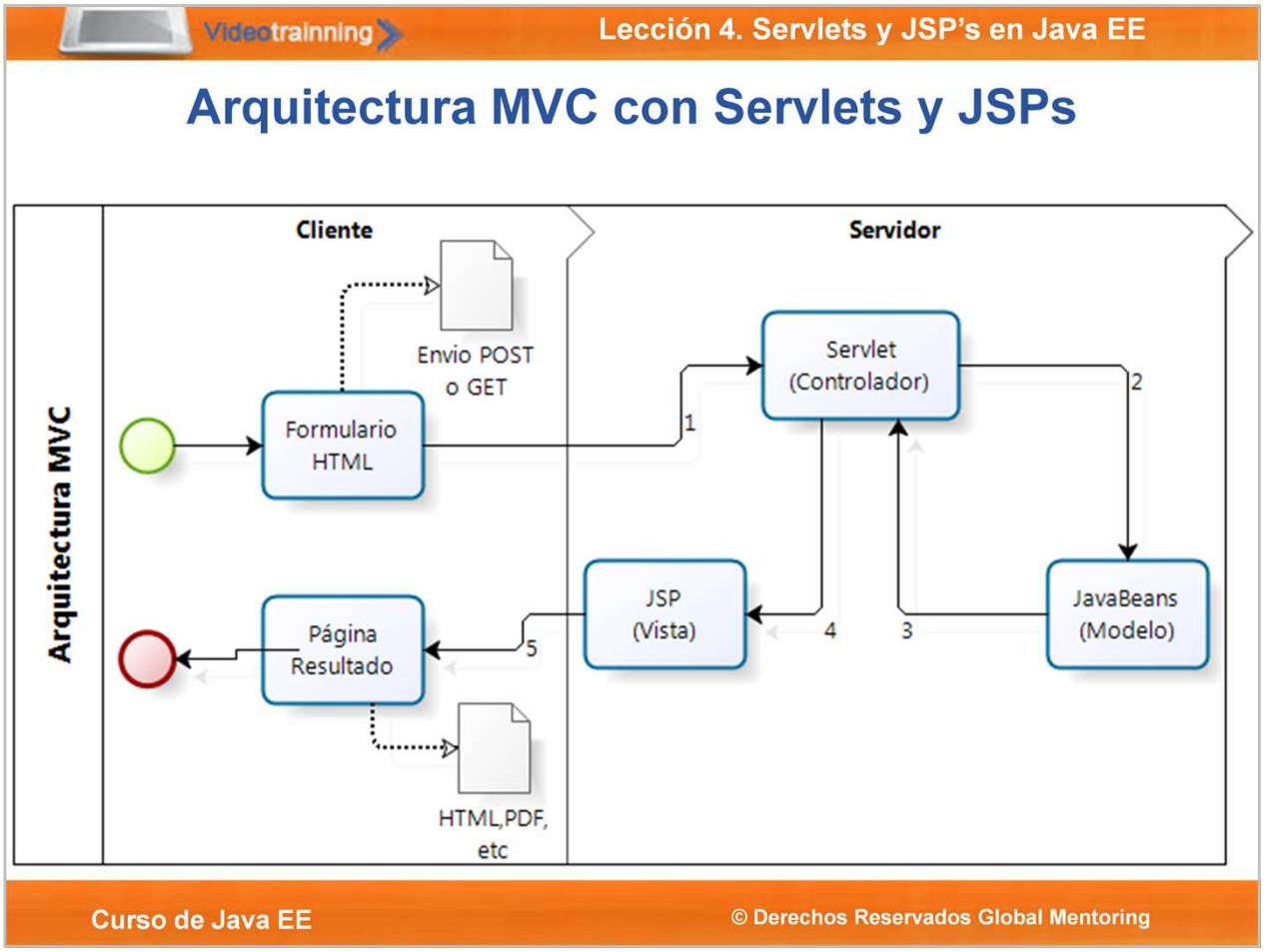
- Un JSP contiene código HTML y puede agregar código Java a través de etiquetas (tags)
- Un JSP se utiliza se debe utilizar como un componente de presentación (Vista) en una arquitectura MVC (Model – View - Controller)



Curso de Java EE

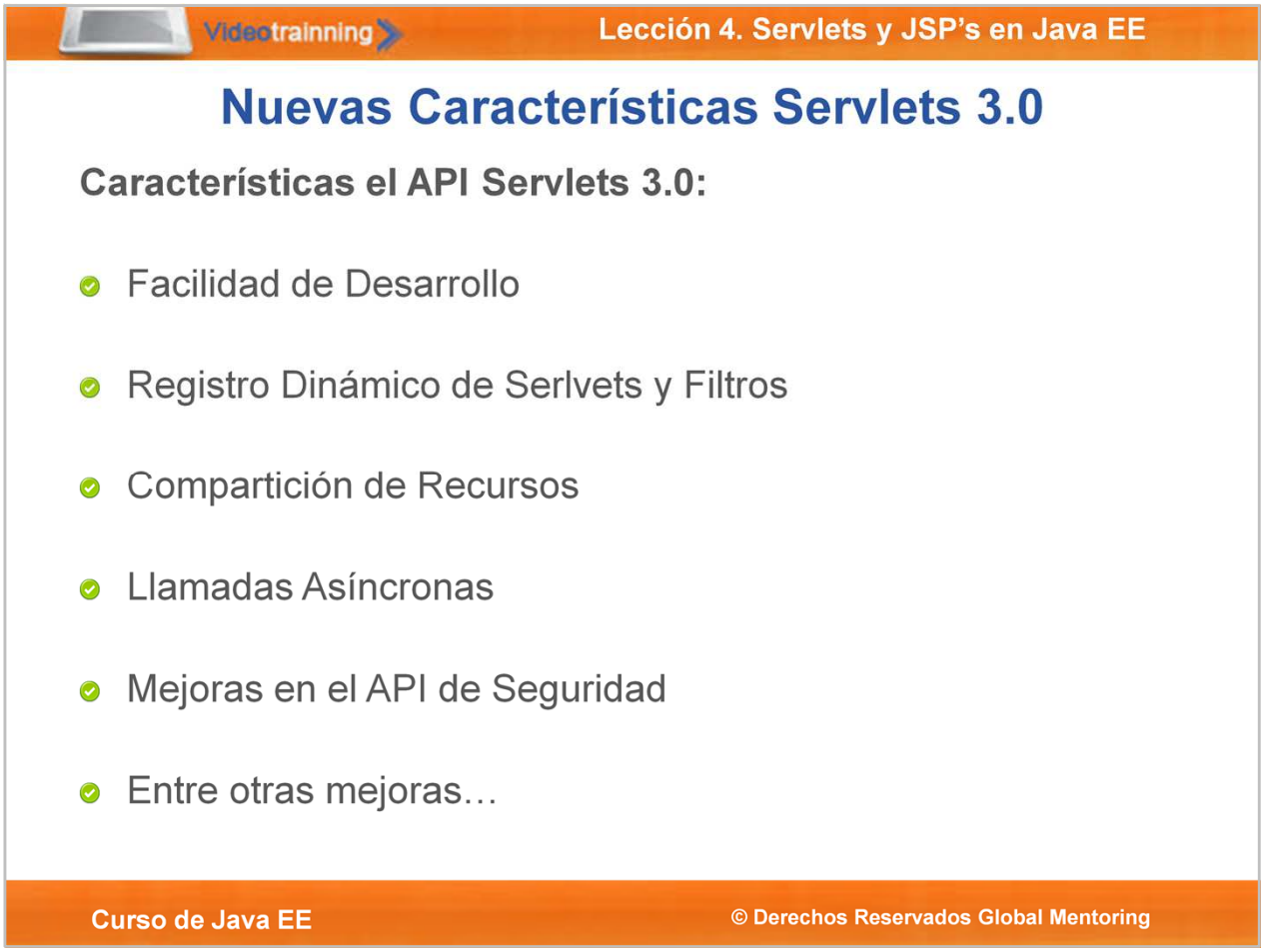
Ejemplo de Código de un JSP listar.jsp que utiliza una lista de objetos Persona como Modelo, el cual fue compartido por un Servlet:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
   <head>
      <title>Listado Personas</title>
   </head>
<body>

   <h1>Listado de Personas</h1>

   <table border="1">
      <tr>
         <th>Nombre</th>
         <th>Apellido Paterno</th>
         <th>Email</th>
      </tr>

   <c:forEach var="persona" items="${personas}">
      <tr>
         <td>${persona.nombre}</td>
         <td>${persona.apePaterno}</td>
         <td>${persona.email}</td>
      </tr>
   </c:forEach>
   </table>

</body>
</html>
```



Ejemplo de Código de un Servlet Controlador aplicando el patrón MVC:

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    //1. Procesamos los parámetros
    String idPersonaString = request.getParameter("idPersona");
    int idPersona = 0;
    if (idPersonaString != null) {

        //2. Creamos/recuperamos el Modelo
        idPersona = Integer.parseInt(idPersonaString);
        Persona persona = new Persona(idPersona);
        persona = this.personaService.encontrarPersonaPorId(persona);

        //3. Compartimos el Modelo con la Vista
        request.setAttribute("persona", persona);

        //4. Redireccionamos a la Vista que mostrará el Modelo
        request.getRequestDispatcher("editarPersona.jsp").forward(request, response);

    }
}
```



El API de los Servlets 3.0 tiene las siguientes mejoras:

Facilidad de Desarrollo:
- Uso de Anotaciones para la declaración de Servlets, Filtros, Listeners, etc.
- El archivo web.xml es opcional (si es utilizado sobreescribe a las anotaciones)
- Soporte del uso de Generics en el API, sin romper compatibilidad
- Mejor soporte de default

Uso de Anotaciones:
- @WebServlet – Define un Servlet
- @WebFilter – Define un filtro
- @WebListener – Define un listener
- @WebInitParam – Define un parametro de inicio
- @MultipartConfig – Define propiedades para subida de archivos
- @ServletSecurity – Define una restricción de seguridad

Además, otras características de los servlets son:
- Registro Dinámico de Servlets y Filtros
- Compartición de Recursos
- Llamadas Asíncronas
- Mejoras en el API de Seguridad
- Entre otras mejoras.

En esta lección nos enfocaremos en realizar un ejercicio que integre la tecnología de los Serlvets, EJB y JPA utilizando las últimas versiones de cada tecnología.

Al día de hoy, la integración entre las distintas tecnologías empresariales es más simple cada vez.

Como podemos observar en el siguiente código, integrar un EJB para ser utilizado en un Servlet es tan simple como utilizar la anotación @EJB y especificar el tipo del EJB a utilizar.

Esto provocará que en automático el servidor de aplicaciones busque una instancia del tipo de EJB especificado, y una vez localizado se realiza una inyección de esta dependencia de manera automática por parte del servidor Java.

```
@WebServlet("/ServletControlador")
public class ServletControlador extends HttpServlet {

   @EJB
   private PersonaService personaService;

   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {

      //Uso de la clase de servicio para recuperar el listado de personas
      List<Persona> personas = personaService.listarPersonas();
      request.setAttribute("personas", personas);

      //Redireccionamos a la vista
      request.getRequestDispatcher("listarPersonas.jsp").forward(request, response);
   }
}
```

Ejercicio 9

- Abrir el documento PDF de Ejercicios del cursos de Java EE.
- Realizar las siguientes prácticas:
- **Ejercicio 9**: Creación Proyecto Web con Servlets/JSPs, EJB y JPA

En Global Mentoring promovemos la *Pasión por la Tecnología Java*.

Te invitamos a visitar nuestro sitio Web donde encontrarás cursos Java Online desde Niveles Básicos, Intermedios y Avanzados.

Además agregamos nuevos cursos para que continúes con tu preparación como consultor Java de manera profesional.

A continuación te presentamos nuestro listado de cursos en constante crecimiento:

- Fundamentos de Java
- Programación con Java
- Java con JDBC
- HTML, CSS y JavaScript
- Servlets y JSP's
- Struts Framework
- Hibernate Framework
- Spring Framework
- JavaServer Faces
- Java EE (EJB, JPA y Web Services)
- JBoss Administration

Datos de Contacto:

Sitio Web: www.globalmentoring.com.mx

Email: informes@globalmentoring.com.mx

Ayuda en Vivo: www.globalmentoring.com.mx/chat.html