Java EE – Arquitectura MVC

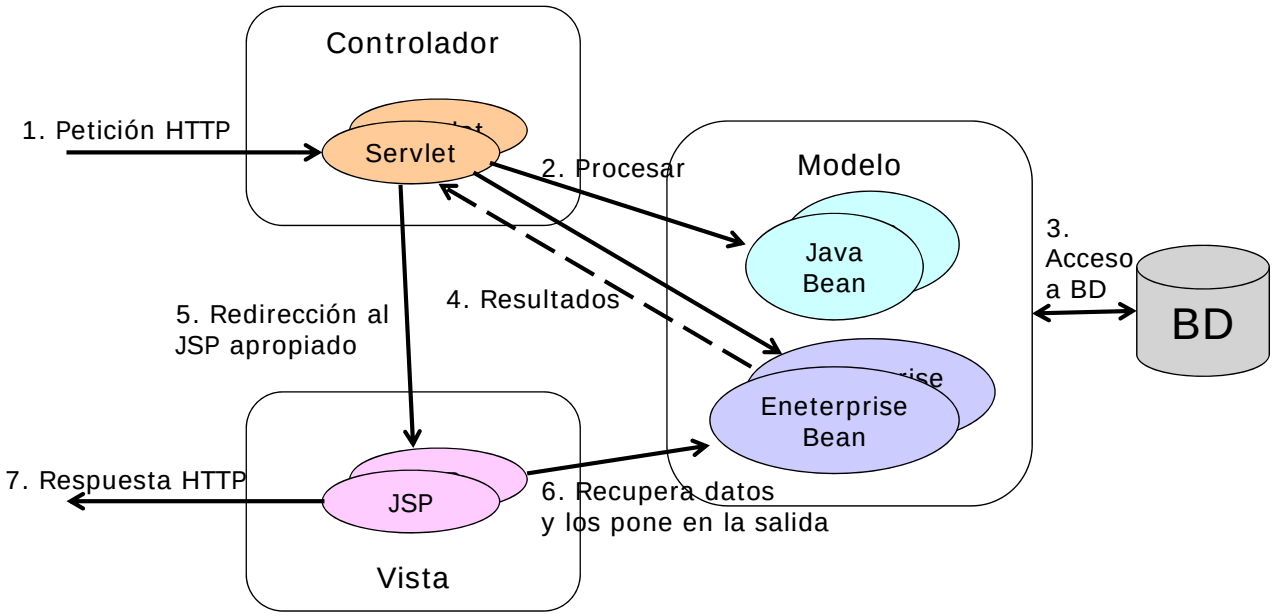
Aplicaciones Web / Sistemas Web

Juan Pavón Mestras
Dep. Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense Madrid

Servlets, JSPs y Enterprise beans

- Los **servlets** facilitan el tratamiento de las peticiones que llegan al servidor
  - Tratamiento de datos de formularios
  - Generación de contenidos de formato variable
  - Permiten redireccionar las peticiones
- Las **JSP** facilitan el desarrollo y mantenimiento del contenido HTML
  - Interesante para páginas de formato establecido (poca variabilidad)
- Los **Java Beans** y **Enterprise Beans** facilitan la implementación de la lógica de negocio
  - Independiente del protocolo de interacción con los clientes
  - Independiente de la presentación de los resultados

Arquitectura MVC en Java EE



Arquitectura MVC en Java EE

- Los beans representan los datos
  - Los resultados pueden ser Java Beans que encapsulan los resultados
  - Beans y Enterprise Beans pueden usarse para la lógica de negocio
- Los servlets gestionan las peticiones de los clientes
  - Reciben las peticiones HTTP
  - Procesan los parámetros (p.ej. de formularios) comprobando que son correctos
  - Invocan operaciones en beans para ejecutar la lógica de negocio
  - Guardan referencias a los beans de resultados en el ServletsContext, en la sesión o en la petición
- Reenvían la petición a una página JSP
  - El servlet determina la página JSP apropiada y usa el método de reenvío del RequestDispatcher para transferirle el control
- La página JSP accede a los beans de resultado
  - La página JSP no crea ni modifica beans, solo los consulta para ver los resultados

Servlets – Reenvío de peticiones

- El servlet determina la página JSP apropiada y usa el método de reenvío del RequestDispatcher para transferirle el control

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
                          throws ServletException, IOException {
// Comprueba los parámetros de la petición (p.ej. de formularios)
// Invoca operaciones en beans para ejecutar la lógica de negocio
// Guarda referencias a los beans de resultados en
// el ServletsContext, en la sesión o en la petición

// Reenvía la petición a un JSP (Vista) para que haga la salida
        String operacion = request.getParameter("operacion");
        if (operacion == null) operacion = "desconocida";
        String jsp;
        if (operacion.equals("consulta"))    jsp = "/WEB-INF/Consulta.jsp";
        else if (operation.equals("cancel")) jsp = "/WEB-INF/Cancel.jsp";
        else     jsp = "/WEB-INF/Desconocida.jsp";

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


Paso de beans

- Se pueden pasar los beans por tres lugares
  - En el objeto request (HttpServletRequest)
    - Visibles para el servlet y la página o servlet a la que se lo reenvía
      - En el servlet:

        ```
        UnBean valor = LookupService.findResult(...);
        request.setAttribute("clave", valor);
        ```

      - En la página JSP:

        ```
        ${clave.propiedad}
        ```

  - En la sesión (HttpSession)
    - Visibles para el servlet y la página o servlet a la que se lo reenvía, y para otras páginas posteriores referentes al mismo usuario
      - En el servlet, similar a antes pero:

        ```
        session.setAttribute("clave", valor);
        ```

  - En el Servlet Context
    - Son visibles para todos los servlets y páginas de la aplicación
      - En el servlet, similar a antes pero:

        ```
        getServletContext().setAttribute("clave", valor);
        ```

      - En otros servlets se tendrá que usar de forma sincronizada (synchronized)

La vista: páginas JSP

- La página JSP no crea beans
  - Para garantizar que no se crean beans, usar

```
<jsp:useBean ... type="paquete.Clase" />
```

    - en vez de

```
<jsp:useBean ... class="paquete.Clase" />
```

- La página JSP no modifica beans, solo los consulta para ver los resultados
  - Usar únicamente jsp:getProperty
  - No usar jsp:setProperty

Conclusiones

- MVC es útil cuando
  - Una petición puede generar resultados de aspecto muy diverso
  - Proceso de información complejo
  - Para repartir el trabajo entre distintos miembros de un equipo de desarrollo
    - Desarrolladores web
    - Lógica de negocio
- Frameworks que usan el MVC para Java EE
  - JavaServer Faces (JFC)
  - Struts