élogos

ANÁLISIS E IMPLEMENTACIÓN DE LOS BLOQUES MVC

élogos

ÍNDICE

ANÁLISIS E IMPLEMENTACIÓN DE LOS BLOQUES MVC

1. El controlador

El controlador es el bloque más crítico de los tres que componen la arquitectura MVC, puesto que es el que se encarga de gobernar el funcionamiento de toda la aplicación.

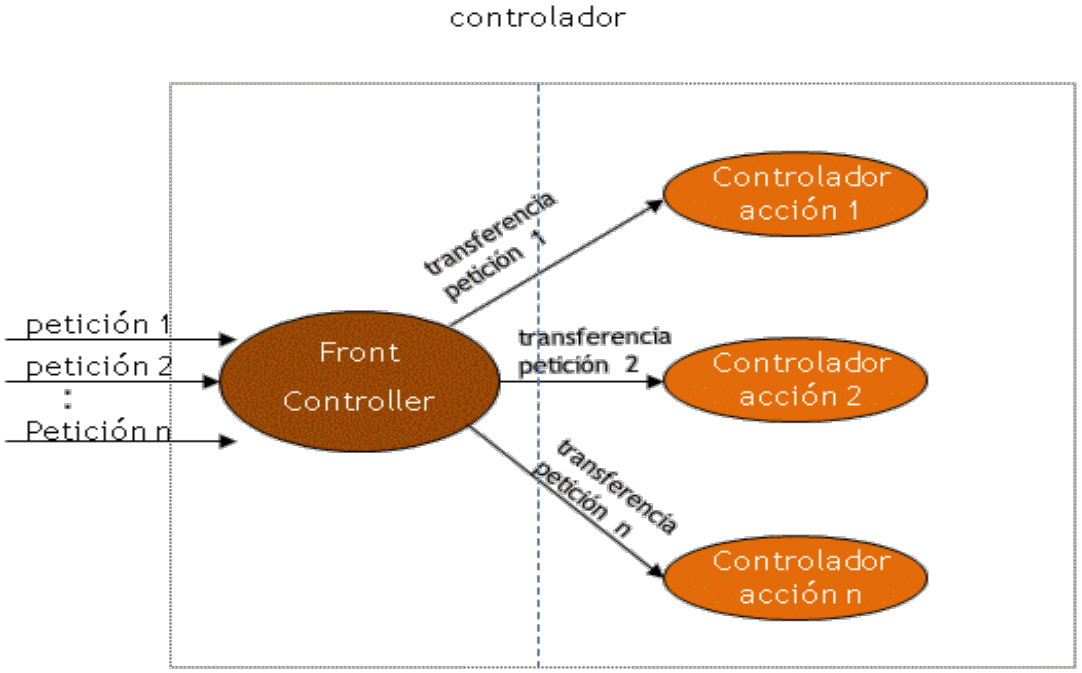
Más concretamente, el controlador es el responsable de:

- Recibir las peticiones de entrada a la aplicación
- Tomar las decisiones sobre las operaciones a ejecutar con cada petición
- Invocar a los métodos del modelo para realizar las operaciones asociadas a la petición en curso.
- Invocar a las vistas que deben ser procesadas para la generación de la respuesta

Diagrama de los bloques internos

El controlador se divide a su vez en dos grandes bloques:

- Front Controller: Es el punto de entrada al que llegan todas las peticiones de la capa cliente. Cada petición es transferida a su correspondiente controlador de acción.
- Controladores de acción: Se encargan de procesar cada petición cliente. Son los responsables de interaccionar con el resto de bloques de la aplicación.



Implementación del Front Controller

Hay diferentes técnicas para que un front controller pueda identificar el tipo de petición entrante y así poder transferirla al controlador de acción apropiado.

Una de estas técnicas consiste en enviar un parámetro en todas las peticiones entrantes, con una valor diferente en cada petición. De esta manera, la estructura del front controller sería similar a la indicada en el siguiente listado:

```
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String op=request.getParameter("op");
        if(op.equals(“valor1")){
                //transferir petición a controlador 1;
        }
        if(op.equals(“valor2")){
                //transferir petición a controlador 2;
        }
        if(op.equals(“valor3")){
                //transferir petición a controlador 3;
        }
        :
}
```

2. La vista

Revisión de funciones

La vista es el bloque encargado de generar las páginas de la aplicación con las que interacciona el usuario. Más concretamente, la vista es el responsable de:

- Realizar la captura de datos de usuario.
- Validar los datos antes de proceder a su envío.
- Generar dinámicamente las páginas de respuesta a una petición, a partir de los datos obtenidos del modelo.
- Dirigir todas las peticiones que parten de la capa cliente al controlador.

Peticiones del controlador

Todas las páginas de la vista que incluyan algún formulario o enlace para acceder a la aplicación, tendrán que apuntar al Servlet front controller.
En la dirección del servlet se añadirá un parámetro que permita identificar el tipo de petición:

Formulario:

```
<form action="Control?operacion=doLogin" method="post">
   Usuario :<input type="text" name="usuario“/>
   :
</form>
```

Enlace:

```
<a href="Control?operacion=doEliminar">
```

Generación de respuestas

Habitualmente, la información generada por el modelo es entregada al controlador. Este deposita los datos en un atributo de petición para que la vista pueda acceder a los mismos una vez que desde el controlador se trasfiera la petición a la página JSP correspondiente.
La página JSP recupera los datos del atributo de petición y les aplica el formato HTML apropiado:

Creación de una tabla HTML a partir de una colección de datos

```
<body>
        <% List<Mensaje> mensajes=
                    (List<Mensaje>)request.getAttribute("mensajes");%>
        <table border="1">
        <tr><th>Remitente</th><th>Texto</th>
        <%for(Mensaje m:mensajes){ %>
           <tr><td><%=m.getRemitente() %></td>
                <td><%=m.getTexto() %></td>
              </tr>
        <%} %>
        </table>
</body>
```

3. El modelo

Revisión de funciones

El modelo encapsula la lógica de negocio de la aplicación. En él, no hay ninguna referencia a operaciones de entrada y salida de datos, ni tampoco a ningún elemento del API Servlet

Más concretamente, la vista es el responsable de:

- Acceder a la capa de datos.
- Encapsular las reglas de negocio de la aplicación.
- Entregar al resto de las capas (normalmente controlador) los datos en un formato Java estándar
- Permitir el intercambio de datos compuestos con el resto de capas, mediante la utilización de JavaBeans.

Mecanismos de implementación

Esta capa puede ser implementada con cualquiera de las siguientes tecnologías:

- Clases estándares Java: La lógica de negocio de la aplicación se implementa en clases normales Java. Las operaciones son expuestas a través de métodos que son invocados desde el controlador.
-
- Enterprise JavaBeans: Se trata de componentes Java especiales que se ejecutan sobre un contenedor que proporciona una serie de servicios extra a los programadores, como gestión automática de transacciones o seguridad declarativa

Dentro del modelo se definen también los JavaBeans, que realizan las funciones del patrón Transfer Object, o lo que es lo mismo, permitir el intercambio de datos compuestos entre las capas.

Ejemplo

El siguiente listado nos muestra un ejemplo de la estructura de una clase utilizada para la implementación del modelo en una aplicación MVC. Como vemos, los métodos de negocio utilizan tipos Java estándar y JavaBeans para el intercambio de datos.

```
public class GestionLibros {
```

```
        public GestionLibros(String nombreJndi){
            //constructor que recibe parámetros
            //de conexión con BD
        }
        public ArrayList<Libro> obtenerlibros() {
                :
        }
        public void registrarusuario(Cliente usuario) {
                :
        }
        public boolean validarusuario(String nombre, String password) {
                :
        }
}
```