Spring Mvc es un **framework** de código abierto para el desarrollo de aplicaciones, basado en el patrón MVC (modelo-vista-controlador).

Desarrollado por Rod Jhonson, *Expert One to One J2EE* Design and Development (Octubre, 2002).

Particularmente, **Spring** identifica malas experiencias en J2EE (complejas, difíciles de probar, etc.) y que no hay suficiente separación entre las capas de Presentación y la Capa de manejo de request, y la Capa de manejo de request y el Modelo.

Módulos principales de Spring

- **Spring Core** (Inversión del control (IoC) / Inyección de dependencias (DI))

  Es el *Núcleo* del Spring, Los componentes no crean (con new) o buscan (p.ej. con JNDI) las referencias a otros componentes que necesitan para realizar su trabajo, sino que simplemente declaran qué dependencias tienen (fichero de configuración, anotaciones) y un contenedor les proporciona (inyecta) estas dependencias.

  El contenedor se encarga de instanciar o localizar los componentes necesarios e inicializarlos antes de inyectar su referencia a quién la necesite

- **Spring AOP** (Programación orientada a aspectos)

  Permite combinar cierto código con otro (aspecto) para añadir cierta funcionalidad al original sin necesidad de modificarlo

  Facilita la implementación de funcionalidades transversales de una aplicación.

  Tipos de AOP

  a. Estática

En tiempo de compilación, se incluye información de aspectos en los bytecodes.

b. Dinámica

En tiempo de ejecución, interrumpe flujo de programa para comprobar si hay que ejecutar aspectos.

- Spring JDBC (Acceso a datos)
- Spring MVC (desarrollo Web según el patrón MVC)
- **Spring Remoting** (distribución)

  Facilitar el desarrollo de aplicaciones distribuidas, reduce el código que se necesita para exponer un bean como servicio o para acceder desde un bean a un servicio remoto.

- Spring Transaction Abstraction (Gestión de transacciones)
- Otros: Spring TestContext (pruebas), Spring JMX (gestión), Spring Security (seguridad), Spring Web Flow (control de flujos en aplicaciones Web)

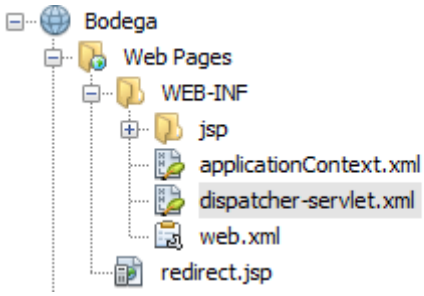
Componentes Spring MVC

Los componentes del MVC son los siguientes:

a. **DispatcherServlet** (org.springframework.web.servlet)

Despacha las solicitudes que recibe al controlador. Lee su configuración dispacher-server.xml, ubicado en /WEB-INF/ de la aplicación, hace referencia al *HandlerMapping*, los *controladores* y el *ViewResolver.*

La clase **DispatcherServlet** está en el front controller y es responsable de delegar y coordinar el control entre varias interfaces en la fase de ejecución durante una petición Http.



```
<bean id="urlMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <props>
            <prop key="index.htm">controlador</prop>
            <prop key="agregar.htm">controlador</prop>
            <prop key="editar.htm">controlador</prop>
            <prop key="delete.htm">controlador</prop>
        </props>

    </property>
</bean>

<bean id="viewResolver"
      class="org.springframework.web.servlet.view.InternalResourceViewResolver"
      p:prefix="/WEB-INF/jsp/"
      p:suffix=".jsp" />

<!--
The index controller.
-->
<bean name="controlador" class="Controller.controlador"/>
```

b. **HandlerMapping** (org.springframework.web.servlet)

Permite manejar peticiones de entrada. El HandlerMapping es una interfaz implementada por objetos para definir la asignación entre los objetos de solicitud y de controlador. Cuando se realiza una solicitud al servlet de despachador de Spring, se entrega la solicitud a la asignación de controlador.

El mapeo de manejadores inspecciona la solicitud, identifica la cadena de ejecución de manejadores apropiada y la entrega a DispatcherServlet.

```
<bean id="urlMapping" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <props>
            <prop key="index.htm">controlador</prop>
            <prop key="agregar.htm">controlador</prop>
            <prop key="editar.htm">controlador</prop>
            <prop key="delete.htm">controlador</prop>
        </props>
    </property>
</bean>
```

c. Controlador

El Controller está entre el modelo y la vista, y permite manejar peticiones entrantes y redirigirlas a la respuesta adecuada.

```
@Controller
public class controlador {
    conexion con= new conexion();
    JdbcTemplate jdbcTemplate = new JdbcTemplate(con.Conectar());
    ModelAndView nav = new ModelAndView();
    int xid;
    List datos;

    @RequestMapping("index.htm")
    public ModelAndView Listar(){
        String sql="select * from vendedor";
        datos=this.jdbcTemplate.queryForList(sql);
        nav.addObject("lista", datos);
        nav.setViewName("index");
        return nav;
    }
```

d. **ModeAndView** (org.springframework.web.servlet)

El ModelAndView está representado por class.org.springframework.web.servlet.ModelAndView y el objeto Controller lo devuelve al DispatcherServlet.

Esta clase es solo una clase contenedora para contener el modelo y la información de vista. El objeto modelo representa cierta información que puede ser utilizada por la Vista para mostrar la información.

e. ViewResolver

Selecciona una vista basada en un nombre lógico de la vista. ViewResolver es una interfaz implementada por objetos para resolver vistas usando el módulo MVC de name.Spring. Encapsula el objeto modelo y el objeto vista en una sola entidad, que está representada por el objeto de la clase ModelAndView.

```
<bean id="viewResolver"
      class="org.springframework.web.servlet.view.InternalResourceViewResolver"
      p:prefix="/WEB-INF/jsp/"
      p:suffix=".jsp" />
```

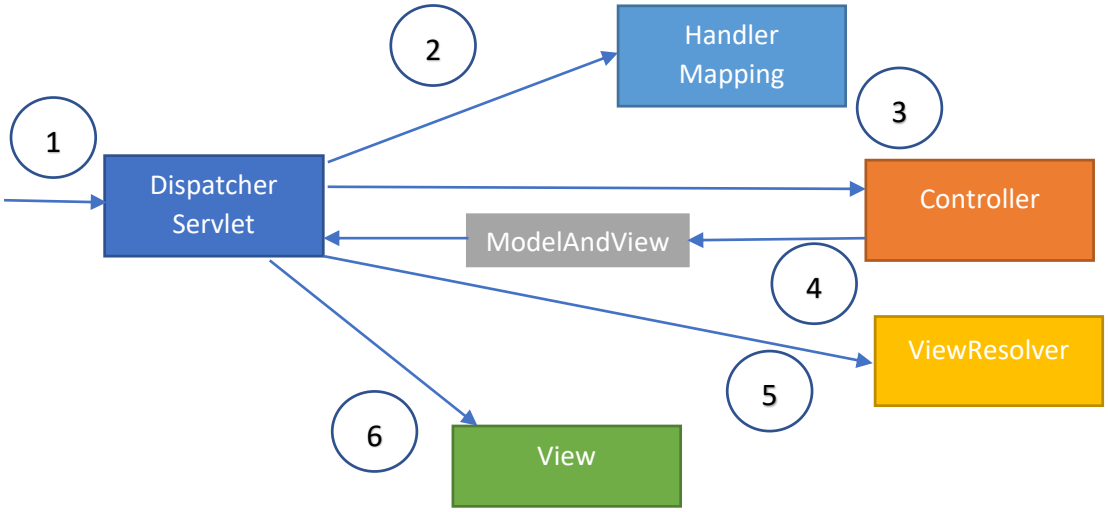


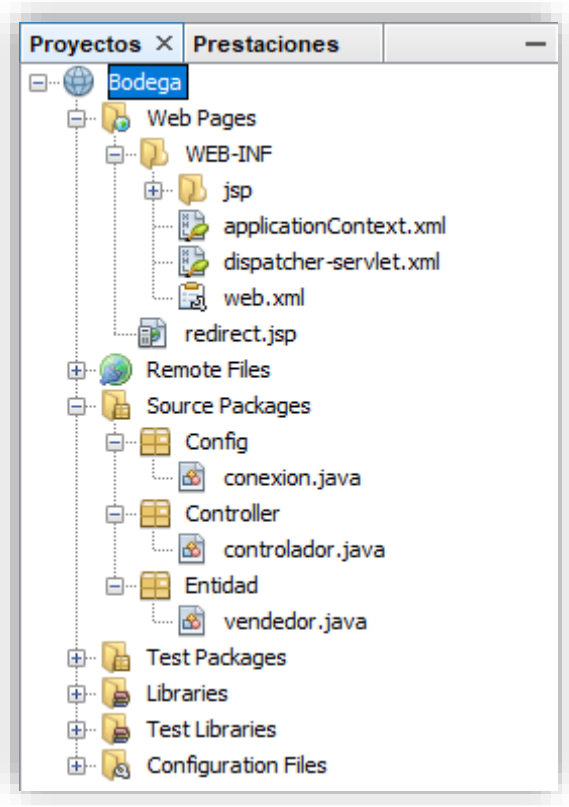
Figura 1. Flujo de proceso de Spring MVC.

Las interfaces importantes definidas en **Spring MVC**, y sus responsabilidades, son las siguientes:

- **HandlerAdapte**r: ejecución de objetos que permiten manejar las peticiones entrantes.
- **View**: responsable de retornar una respuesta al cliente.
- **HandlerInterceptor**: intercepta las peticiones entrantes, es comparable pero no igual a los filtros de Servlet.
- **LocaleResolver**: resuelve y opcionalmente salva el locale de un usuario individual.
- **MultipartResolver**: facilita trabajar con ficheros de subida wrapping peticiones de entrada.

Nota

El modelo son los datos con los que interactúa la vista y el controlador. En este caso, el modelo se pasa mediante un atributo de la clase **ModelAndView**, para posteriormente acceder al mismo desde la parte de la vista.



Explicación técnica del Proceso – Sprint MVC

- Después de recibir una solicitud HTTP, **DispatcherServlet** consulta el **HandlerMapping** para llamar al controlador apropiado.
- El controlador toma la solicitud y llama al método de servicio apropiado en función del método GET o POST utilizado. El método de servicio establecerá los datos del modelo según la lógica empresarial definida y devolverá el nombre de la vista al DispatcherServlet.
- El **DispatcherServlet** tomará la ayuda de ViewResolver para recoger la vista definida para la solicitud.
- Una vez que se finaliza la vista, **DispatcherServlet** pasa los datos del modelo a la vista; esto finalmente se representa en el navegador.

Explicación Flujo de trabajo - Spring MVC -

1. El controlador o Servlet recibe solicitudes del navegador y captura la entrada
2. El controlador invoca el método de negocio del modelo o bean java.

3. El modelo se conecta con la base de datos y obtiene datos de negocios.
4. El modelo envía la respuesta al controlador (mantiene los datos de proceso en la solicitud de memoria del montón, la sesión y el contexto de Servlet)
5. Los controladores cambian el control a la vista apropiada de la aplicación.

```
public class conexion {
    public DriverManagerDataSource Conectar(){
        DriverManagerDataSource ds = new DriverManagerDataSource();
        ds.setDriverClassName("com.mysql.jdbc.Driver");
        ds.setUrl("jdbc:mysql://localhost:3306/Bodega");
        ds.setUsername("root");
        ds.setPassword("");
        return ds;

    }
}
```