MVC y MVVM (3-5)

Por **Usuario Anónimo** - 7 octubre, 2012

Framework ZK

MVC y MVVM (3-5)

¿Qué son y en qué se diferencian?

Contenido

Introducción

Este tutorial pertenece a la siguiente serie:

1. Configurar el entorno
2. Empezar a programar
3. MVC y MVVM ¿Qué son y en qué se diferencian?
4. MVC en ZK
5. MVVM en ZK

Está dirigido a desarrolladores con experiencia en la creación de aplicaciones web en Java. A continuación aprenderás a manejar los componentes de la vista **manualmente** e implementar la lógica necesaria para que funcione nuestra aplicación.

A lo largo de este documento hacemos referencias a fuentes externas, las encontrarás todas en la sección de Referencias

¿Qué son MVC y MVVM?

Son patrones de diseño o modelos de abstracción utilizados para definir y estructurar los componentes necesarios en el de desarollo de software.

Una manera muy simple de describirlos sería:

MVC significa **Modelo Vista Controlador**, porque en este patrón de diseño se separan los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. Cuando la lógica de negocio realiza un cambio, es necesario que ella sea la que actualiza la vista.

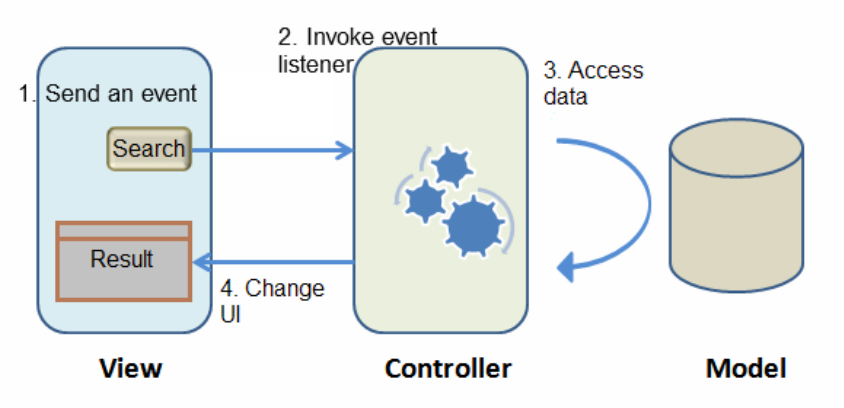
MVVM significa **Modelo Vista VistaModelo**, porque en este patrón de diseño se separan los datos de la aplicación, la interfaz de usuario pero en vez de controlar manualmente los cambios en la vista o en los datos, estos se actualizan directamente cuando sucede un cambio en ellos, por ejemplo si la vista actualiza un dato que está presentando se actualiza el modelo automáticamente y viceversa.

Comparación entre MVC y MVVM

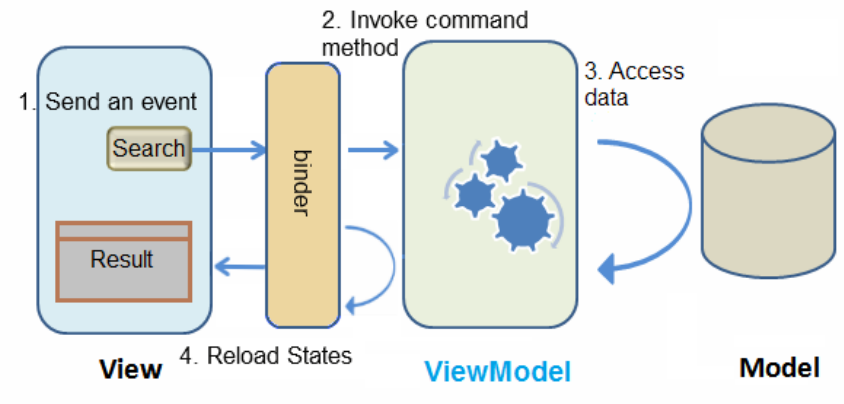
La primera imagen describe la interacción que se produce con MVC en ZK, y la siguiente es la correspondiente al enfoque MVVM que hace ZK.

Las principales diferencias entre MVC y MVVM son que en MVVM el «**Controller**» cambia a «**ViewModel**» y hay un «**binder**» que sincroniza la información en vez de hacerlo un controlador «**Controller**» como sucede en MVC.

MVC



MVVM



Los dos enfoques tienen muchas cosas en común, pero también hay claras diferencias entre ellos. Cada uno de los 2 enfoques tiene su razón de ser/existir.

¿Cuál debo usar?

Construir una aplicación mediante **MVC** puede resultar más intuitivo, porque directamente controlas lo que ves en la vista y su comportamiento, es decir manualmente. MVC se caracteriza porque tienes control total de los componentes, por lo tanto puedes crear componentes hijo dinámicamente («child»), controlar componentes propios personalizados, o realizar cualquier cosa sobre el componente que este pueda hacer por sí mismo.

En el patrón **MVVM**, puesto que la capa «ViewModel» esta débilmente acoplada con la vista (no está referenciada a los componentes de la vista), podemos usarla con múltiples vistas sin tener que modificarla. Por lo tanto los diseñadores y programadores pueden trabajar en paralelo. Si la información y comportamiento no cambian, un cambio en la vista no provoca que se tenga que modificar la capa «ViewModel». Además, como la capa «ViewModel» es un POJO, es fácil realizar tests unitarios sobre ella sin ninguna configuración ni entorno especial. Lo que significa que la capa «ViewModel» tiene una mejor reusabilidad, testabilidad, y los cambios en la vista le afectan menos.

Para sumarizar, comparamos los 2 patrones de diseño en una tabla:

	MVC	**MVVM**
Acoplamiento con la vista	Muy poco con plantilla	Muy poco
Acoplamiento con el componente	Un poco	Muy poco
Codificar en la vista	Mediante el ID del componente	A través de una expresión Data binding
Implementación de un controlador	Extendemos ZK's Composer	Es un POJO
Acceso a la información de la UI	Acceso directo	Automático
Acceso a la información desde el backend	Acceso directo	Acceso directo
Actualización de la interfaz de usuario	Manipulamos directamente los componentes	Automático (@NotifyChange)
Nivel de control del componente	Elevado, control total	Normal
Rendimiento	Alto	Normal

Referencias

- Página web oficial de ZK
- MVC en Wikipedia
- MVVM en Wikipedia