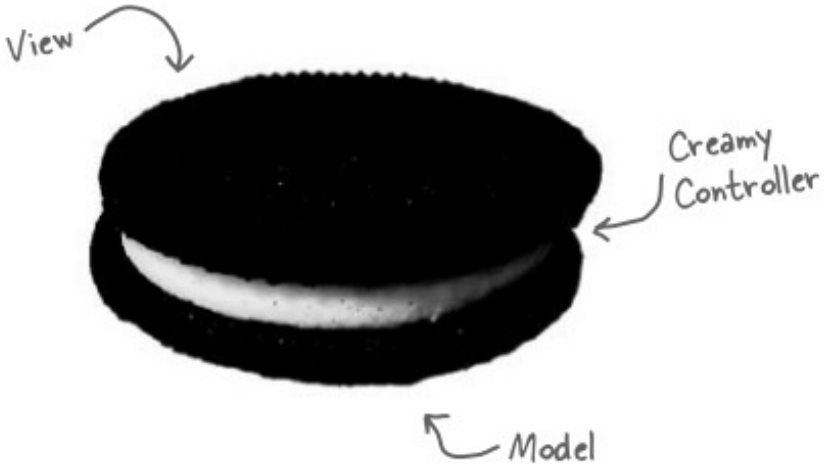
El Rey de los Patrones Compuestos MVC

- Define la organización independiente de los datos, la lógica de negocio, la gestión de eventos y la interfaz de usuario de una aplicación.
- Construcción tres componentes: Modelo, Vista y Controlador.

- MVC es un paradigma para factorizar tu código en segmentos funcionales, para que tu cerebro no explote.
- Para lograr la reutilización, debe mantener esos límites limpios, Modelo por un lado, Vista en el otro, el Controlador en el medio.

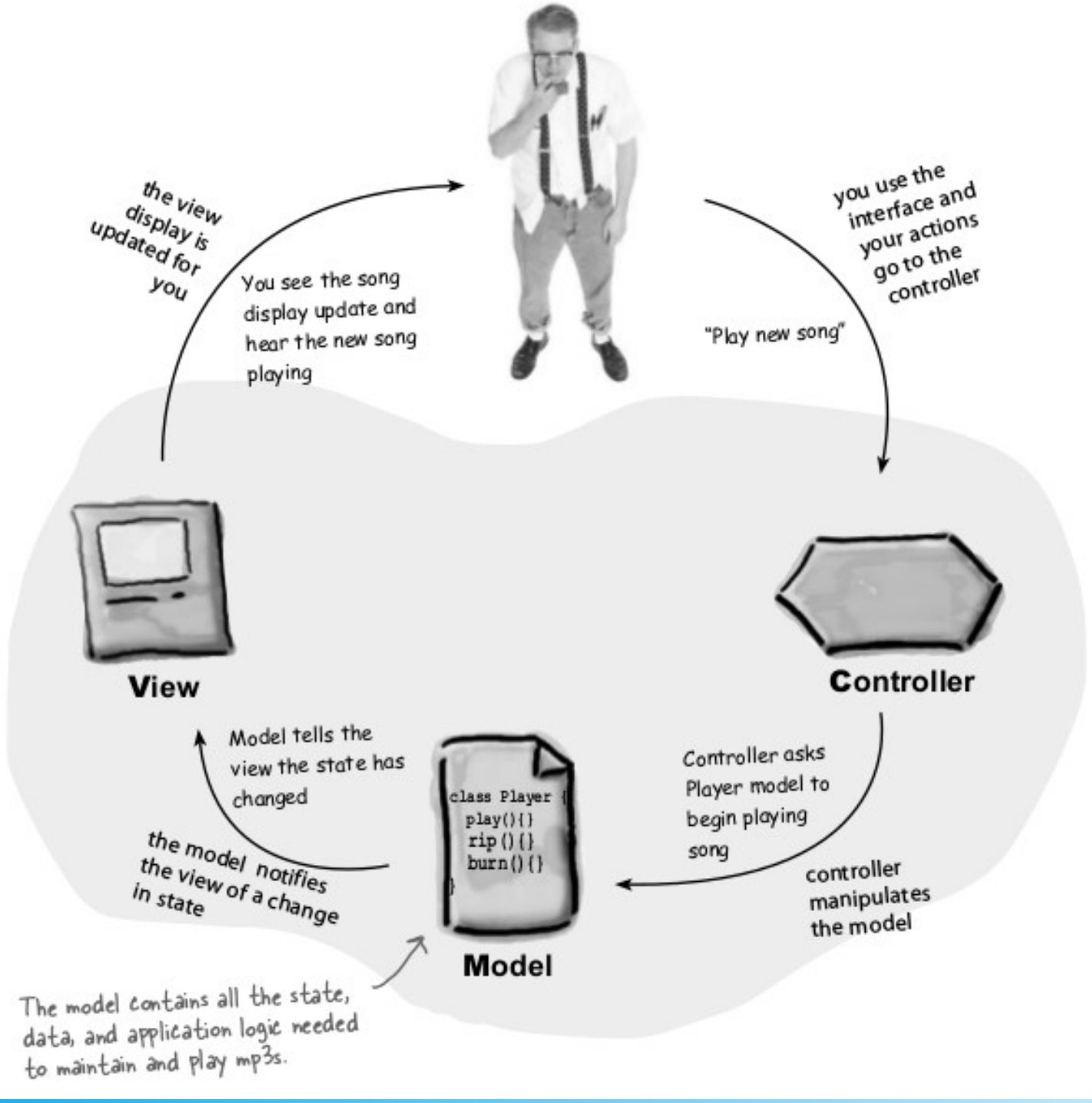


MP3

- Imagina que tu reproductor MP3 favorito. Puede utilizar su interfaz para agregar nuevas canciones, gestionar listas de reproducción y cambiar el nombre de las pistas

MP3

- El reproductor se encarga de mantener una base datos con los nombres de las canciones y los datos asociados.
- Al reproducir las canciones, la interfaz del usuario se actualiza constantemente con el título de la canción actual y el tiempo de ejecución.


the view display is updated for you
You see the song display update and hear the new song playing
you use the interface and your actions go to the controller
"Play new song"
View
Controller
Model tells the view the state has changed
the model notifies the view of a change in state
class Player {
play(){}
rip(){}
burn(){}
}
Controller asks Player model to begin playing song
controller manipulates the model
Model
The model contains all the state, data, and application logic needed to maintain and play mp3s.

VISTA

Obtiene el estado y los datos que necesita del Modelo para visualizar.

- Le da una presentación adecuada al Modelo para interactuar (interfaz de usuario).

CONTROLADOR

Toma la entrada del usuario y se da cuenta de lo que significa para el modelo.

- Responde a eventos (acciones del usuario).
- Invoca al Modelo cuando se hace alguna solicitud sobre la información, y puede enviar notificación a la Vista si se solicita un cambio en la forma de presentar el Modelo.
- Es el intermediario entre la Vista y el Modelo

MODELO

El modelo contiene todos los datos, el estado y la lógica de la aplicación. El modelo es ajeno a la vista y al controlador.

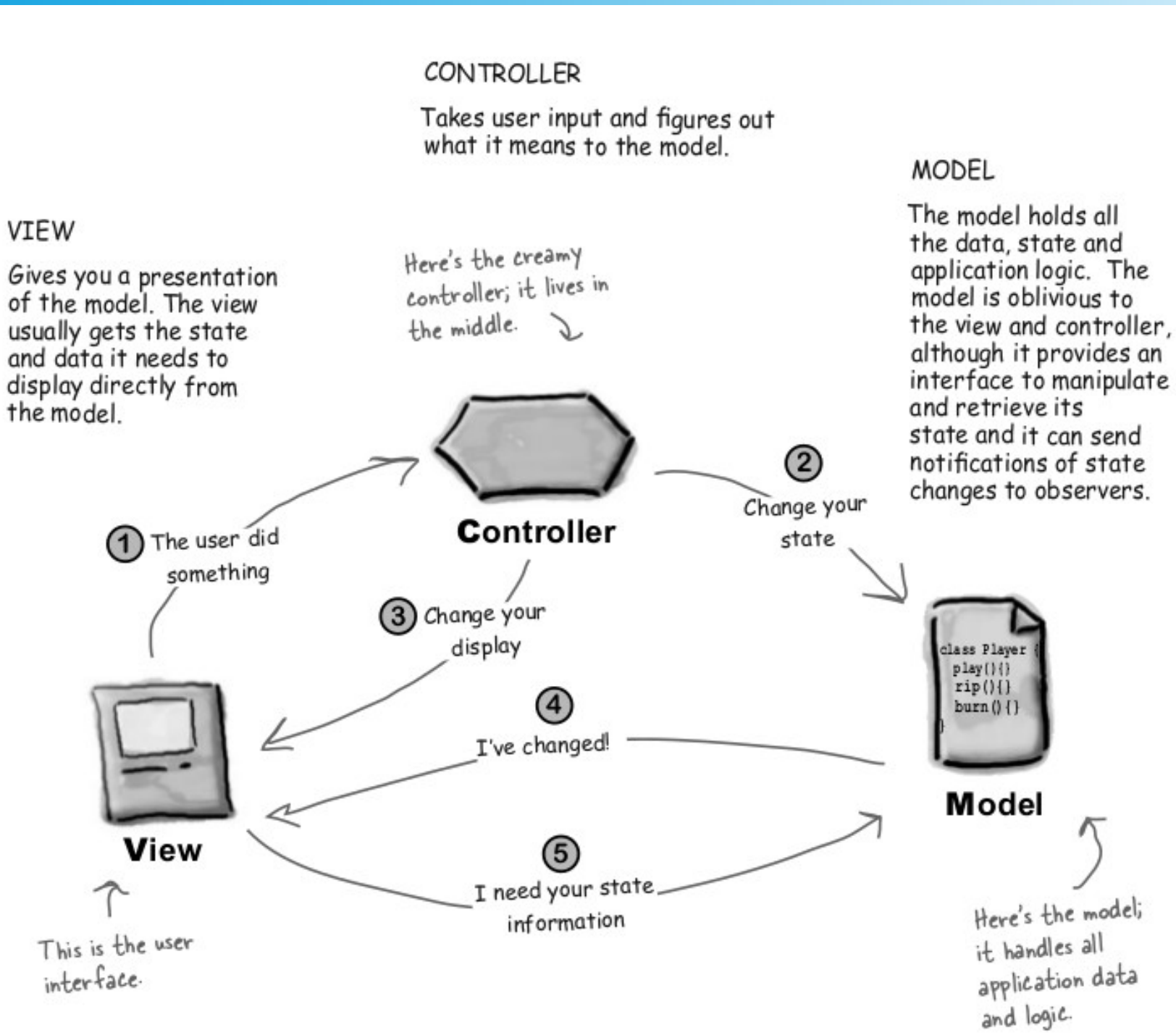
Sin embargo, proporciona una interfaz para manipular y recuperar su estado y puede enviar notificaciones de cambios de estado a los observadores.

MODELO

- Es la representación de la información con la cual el sistema opera.
- Gestiona todos los accesos a la información (actualizaciones y consultas).
- Envía a la Vista los datos solicitados para que sean mostrados.
- Las peticiones de acceso llegan a través del Controlador.

¿De que esta compuesto el MVC?

- Un modelo
- Varias vistas
- Varios controladores
- Las vistas y los controladores suelen estar muy relacionados
- Los controladores tratan los eventos que se producen en la interfaz gráfica (vista)


CONTROLLER
Takes user input and figures out what it means to the model.
MODEL
The model holds all the data, state and application logic. The model is oblivious to the view and controller, although it provides an interface to manipulate and retrieve its state and it can send notifications of state changes to observers.
VIEW
Gives you a presentation of the model. The view usually gets the state and data it needs to display directly from the model.
Here's the creamy controller; it lives in the middle.
Controller
1 The user did something
2 Change your state
3 Change your display
4 I've changed!
5 I need your state information
View
Model
class Player {
play(){}
rip(){}
burn(){}
}
This is the user interface.
Here's the model; it handles all application data and logic.

Interacción de los componentes:

1) El usuario: interactúas con la vista.

   Cuando haces algo con la vista (como hacer clic en el botón Reproducir), la vista le dice al controlador lo que hiciste. El trabajo del controlador es manejarlo.

2) El controlador le pide al modelo que cambie su estado.

   El controlador toma sus acciones y las interpreta. Si hace clic en un botón, es el trabajo del controlador para descubrir qué significa y cómo se debe manipular el modelo en función de esa acción.

Interacción de los componentes:

3) El controlador también puede pedirle a la vista que cambie.

   Por ejemplo, el controlador puede habilitar o deshabilitar ciertos botones o elementos de menú en la interfaz.

4) El modelo notifica a la vista cuando su estado ha cambiado.

   Cuando algo cambia en el modelo, en función de alguna acción que haya realizado (como hacer clic en un botón) u otro cambio interno (como la siguiente canción en la lista de reproducción), el modelo notifica a la vista que su estado ha cambiado.

Interacción de los componentes:

5) La Vista obtiene el estado (actualizado) que será mostrado a partir del Modelo.

   La vista le pregunta al modelo por el estado.

   La vista obtiene el estado que muestra directamente desde el modelo.

   Por ejemplo, cuando el modelo notifica a la vista que una nueva canción ha comenzado a reproducirse, la vista solicita el nombre de la canción al modelo y lo muestra.

   La vista también podría preguntarle al modelo el estado como resultado de que el controlador solicito algún cambio en la vista.

Ventajas

- Reutilización de código.
- Facilita las tareas de desarrollo de aplicaciones.
- Facilita el mantenimiento del software.

Framework

¿Qué es un framework?

Un framework es un entorno o marco de trabajo, un conjunto de prácticas, conceptos y criterios a seguir estandarizados.

El objetivo del framework es intentar ahorrarnos trabajo.

¿Para qué sirve un Framework?

Es un conjunto de:

- Convenciones, estándares o paradigmas y buenas prácticas
- Funcionalidades costosas (tiempo o seguridad) ya desarrolladas

Ventajas de usar Framework

- Proceso de depuración más sencillo
- Fácil reutilización de código
- Desarrollo más rápida: el programador ahorra tiempo ya que dispone de un esqueleto
- Seguridad mejorada
- Facilita los desarrollos colaborativos
- Es más fácil encontrar herramientas
- Existe una comunidad detrás dispuesta a ayudar

Algunos Frameworks

.NET	Apache	Spring.NET
Ruby	MIT	Ruby on Rails
Java Apache		Struts
Java Apache		Spring
Java Apache		JavaServerFaces
Python	BSD	Django
PHP		Laravel
PHP	BSD	Zend
PHP	MIT	CakePHP

Most Popular Backend Frameworks