PATRONES DE DISEÑO EMPRESARIALES – TERCERA PARTE

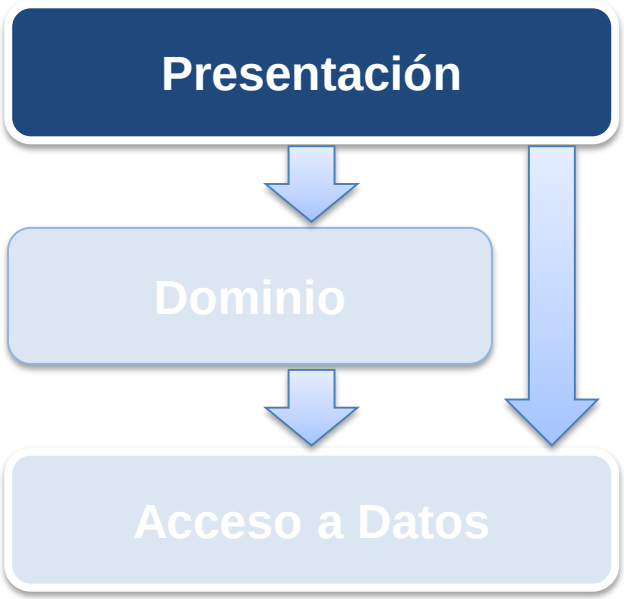
ELSA ESTEVEZ
UNIVERSIDAD NACIONAL DEL SUR
DEPARTAMENTO DE CIENCIAS E INGENIERIA DE LA COMPUTACION

CONTENIDO

PATRONES DE PRESENTACIÓN

MOTIVACION –

Una de los cambios mas importantes en las aplicaciones empresariales en los últimos tiempos fue la utilización de las interfaces de usuario (IU) basadas en la Web.



Ventajas de IU basadas en la Web:

- No se necesita instalar software en el cliente
- Se puede adoptar un enfoque común de IU
- Se provee un fácil acceso universal

PRESENTACION WEB – 1

El desarrollo de una aplicacion web comienza con el software en el servidor.

Usualmente,

- se utiliza un archivo de configuracion que indica que URLs deben ser manejadas por que programas
- un servidor web puede manejar varias clases de programas
- el trabajo del servidor web es interpretar el URL de un pedido y pasar el control al programa del servidor Web

PRESENTACION WEB – 2

Hay dos formas de estructurar un programa en un servidor web:

como un script – un programa, usualmente con funciones o metodos para manejar el HTTP call; ejemplos: CGI scripts y Java servlets

- ✓es recomendado cuando se necesita interpretar un pedido

como un page server – un programa estructurado en base a la pagina de texto resultado, se insertan scriptlets de codigo HTML para ser ejecutados en determinados puntos; ejemplos: PHP, ASP, y JSP

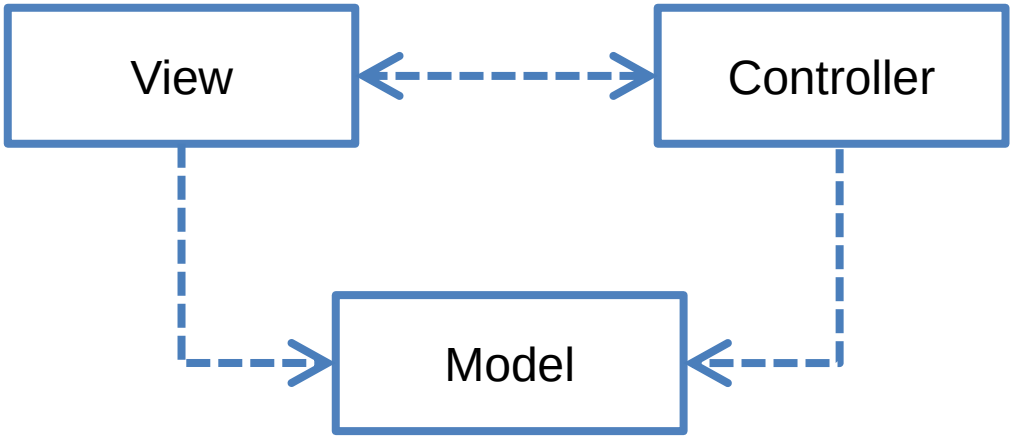
- ✓es recomendado cuando se necesita formatear una respuesta

DOS TIPOS DE PATRONES

PATRONES DE CONTROLADOR DE INPUT (INPUT CONTROLLER PATTERNS)	PATRONES DE VISTAS (VIEW PATTERNS)
○ Model View Controller	○ Template View
○ Page Controller	○ Transform View
○ Front Controller	○ Two Steps View

MODEL-VIEW-CONTROLLER

DESCRIPCION – Separa la interacción de la interfaz de usuario en tres roles distintos



Uno de los patrones de diseño más conocidos , teniendo influencia desde su creación (1970) en muchos frameworks de UI y en la forma de pensar sobre el diseño de interfaces de usuario.

MODEL-VIEW-CONTROLLER – COMO TRABAJA?

ROLES	
MODELO	○ es un objeto que representa al dominio, contiene datos y comportamiento distinto a los de la UI ○ en su forma más pura de OO, sería un objeto del Domain Model ○ también podría ser un Transaction Script del Dominio
VISTA	○ representa la visualización del modelo en la interfaz de usuario ○ sólo conoce como mostrar la información
CONTROLADOR	○ toma información del request, invoca al objeto del modelo correspondiente y, en base a los resultados obtenidos, determina qué vista es la apropiada para mostrar. Luego le pasa el control a la vista, junto con los datos de respuesta. ○ toma la entrada del usuario, manipula el modelo y refresca la vista apropiadamente.

MODEL-VIEW-CONTROLLER – PRINCIPALES SEPARACIONES

SEPARAR LA PRESENTACIÓN DEL MODELO

- Es uno de los más importantes principios de diseño.
- Generalmente tienen intereses distintos
  - ✓ Al desarrollar una vista, nos preocupamos por mecanismos de UI y por cómo diseñar una buena UI.
  - ✓ Al trabajar con el modelo, se piensa en términos de reglas de negocio y, quizás, de interacciones con la base de datos.
- Se podría querer ver la misma información del modelo de distintas formas.
- Objetos no visuales son más fáciles de testear que objetos visuales.
- La presentación depende del modelo pero el modelo no depende la presentación.

SEPARAR EL CONTROLADOR DE LA VISTA

- No es tan importante, pero igualmente brinda beneficios. Permitiría tener más de un controlador por vista, o distintas vistas usar el mismo controlador.
- Ejemplo: Soportar comportamiento de edición y visualización con una vista.
  - ✓ Podríamos tener 2 controladores, uno para cada caso, donde los controladores son Strategies (GoF) de la vista.

MODEL VIEW CONTROLLER – CUANDO UTILIZARLO?

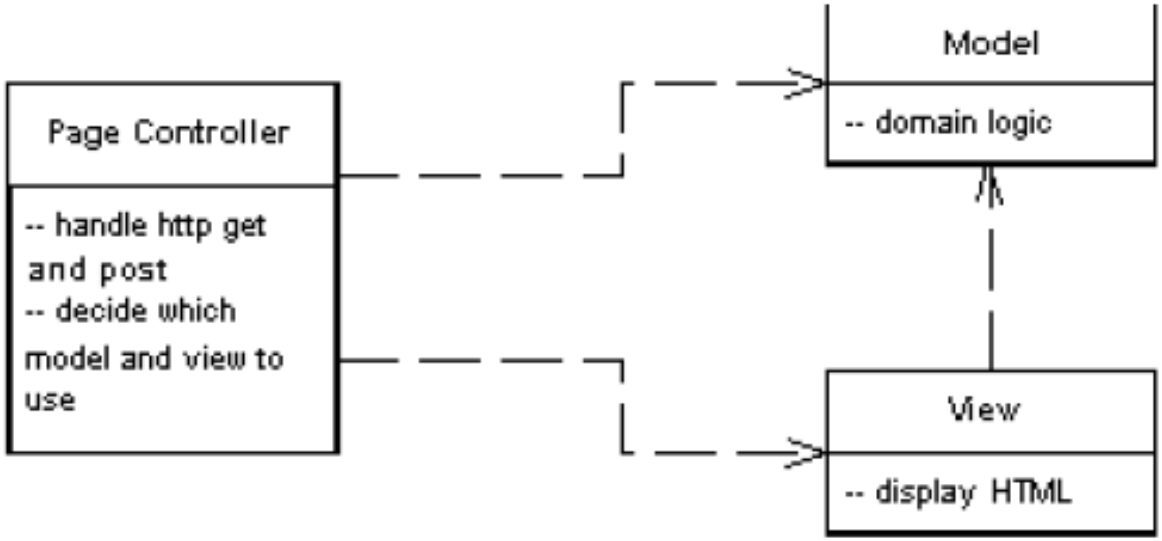
- Cuando las separaciones indicadas anteriormente (especialmente la separación de la presentación y el modelo) son útiles.
- Si se tiene una aplicación muy simple donde el modelo no tiene comportamiento.
- Si las tecnologías a utilizar no brindan la infraestructura necesaria.

MODEL VIEW CONTROLLER – EJEMPLOS

TECNOLOGÍAS	FRAMEWORKS
Java	Spring MVC Struts JSF Seam Tapestry
.Net	ASP.NET MVC
PHP	CakePHP, Lamplighter (ex Fuse) Symfony
Javascript	Backbone.js, Knockout.js, JavascriptMVC, Angular.js

PAGE CONTROLLER

DESCRIPCION – Un objeto que maneja un request para una página o acción de una aplicación web



Page Controller tiene un controlador para cada página lógica de la aplicación web.

El controlador puede ser la página en sí misma (frecuente en ambiente *server pages*), o puede se un objeto separado que se corresponda con la página. Sin embargo, este patrón de diseño apunta a una alternativa de un path que conduce a un archivo que maneja un requerimiento

PAGE CONTROLLER – COMO TRABAJA?

IDEA BÁSICA

- Un módulo (clase o lo que fuera) actúa como controlador para cada página de la aplicación web.
- En la realidad, existirá un controlador por cada acción (incluidas los eventos)

RESPONSABILIDADES

- Decodificar la URL y extraer cualquier dato del request que sea necesario para realizar la acción requerida.
- Crear e invocar objetos del modelo para procesar los datos de entrada. Los objetos del modelo no necesitan conocer al request.
- Determinar qué vista se debe mostrar y enviarle el modelo que corresponda.

PAGE CONTROLLER – FORMAS DE IMPLEMENTACIÓN

SCRIPT	CGI script, Servlet, etc
	○ Script que actúa como handler y controlador ✓ El web server le pasa el control al script, el script lleva a cabo las responsabilidades de un controlador y finalmente transfiere el control a la vista que corresponda.
SERVER PAGE	ASP, ASP.NET, PHP, JSP,
	○ Generalmente combina los patrones Page Controller y Template View en un mismo archivo. Esto favorece al Template View en detrimento del Page Controller (más difícil estructurar el modulo) ○ Para evitar scriptlets ✓ Utilizar helpers: La server page llama, en primer lugar, al helper para que maneje la lógica. Luego el helper le retorna el control a la página o se lo transfiere a otra.

RECOMENDACIONES

- No necesita ser una única clase
- Se podrían tener ambas alternativas en una misma aplicación

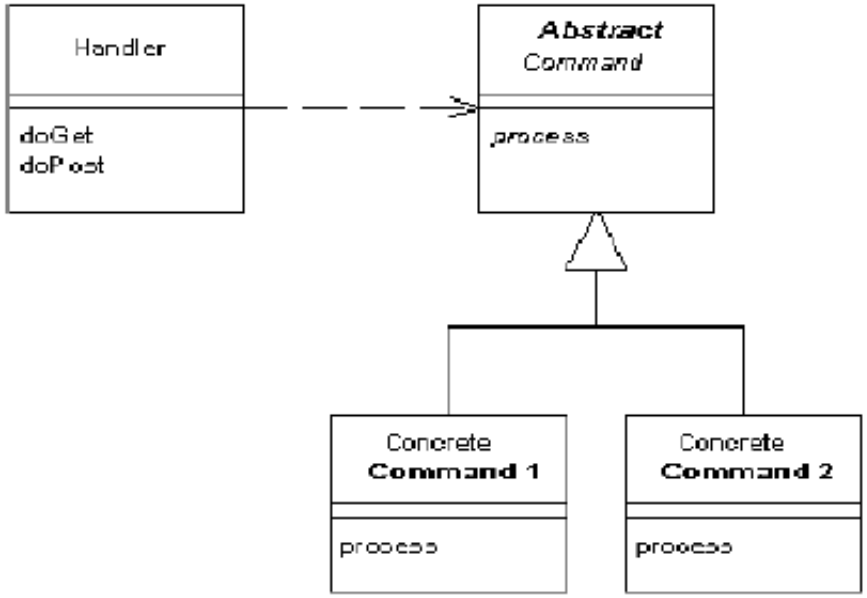
PAGE CONTROLLER – CUANDO UTILIZARLO?

- 👍 Cuando mucha de la lógica del controlador es muy simple, ya que no agrega demasiado overhead.

- 👎 Cuando hay mucha complejidad navegacional, sería preferible usar un *Front Controller*

FRONT CONTROLLER

DESCRIPCION – Un controller que maneja todos los request para una aplicación web.



Front Controller consolida el manejo de requests, canalizándolos a través de un único objeto handler. Este objeto lleva adelante comportamiento común (seguridad, i18n, logging, etc.) y luego despacha el request a un *command* para realizar el comportamiento específico.

FRONT CONTROLLER – COMO TRABAJA?

IDEA BÁSICA

- Un Front Controller maneja todas las llamadas a una aplicación web
- Está estructurado en 2 partes: un web handler y una jerarquía de comandos.
- Web Handler:

  - ✓ Es el objeto que realmente recibe POSTs y GETs del web server
  - ✓ Obtiene información de la URL y del request para decidir qué tipo de acción iniciar
  - ✓ Delega al *command* correspondiente la atención del requerimiento.
  - ✓ Generalmente es implementado como una clase y no con una server page. No produce ninguna respuesta por sí mismo.

- Comandos:

  - ✓ Son clases y no server pages
  - ✓ Generalmente no necesitan conocimiento del ambiente web, aunque muchas veces se les pasa el request HTTP.

FRONT CONTROLLER – SELECCIÓN DEL COMANDO

ESTÁTICA

- Parsea la URL y utiliza lógica condicional
- Tiene las ventajas de la lógica explícita:
  - ✓ Chequeos en compilación
  - ✓ Flexibilidad en el formato de la URL

DINÁMICA

- Utiliza un patrón de URL y utiliza instanciación dinámica para crear comandos.
- Permite agregar nuevos comandos sin cambiar el web handler.
- Implementación
  - ✓ Poner el nombre de la clase comando en la URL
  - ✓ Usar properties files / XMLs para mapear las URLs con los comandos.

Uso con Intercepting Filter

- Es un decorador que wrappea al web handler de un Front Controller permitiendo construir un *pipe & filter*
- Permite manejar autenticación, logging, i18n, etc.

PAGE CONTROLLER – CUANDO UTILIZARLO?

- Si las ventajas pagan el esfuerzo:
  - Si se tiene una aplicación muy simple donde el modelo no tiene comportamiento. Sólo un Front Controller tiene que ser configurado en el web server.
  - Con comandos dinámicos, se podrían agregar comandos sin ningún cambio más que en la configuración.
  - No es necesario que los comandos sean thread-safe, ya que son creados por cada request
  - Permite factorizar código que, de otra forma, estaría duplicándose en las Page Controller.
  - Se puede mejorar el comportamiento del Front Controller, a través de decoradores para autenticación, codificación de caracteres, i18n, etc, y agregarlos a través de un archivo de configuración.

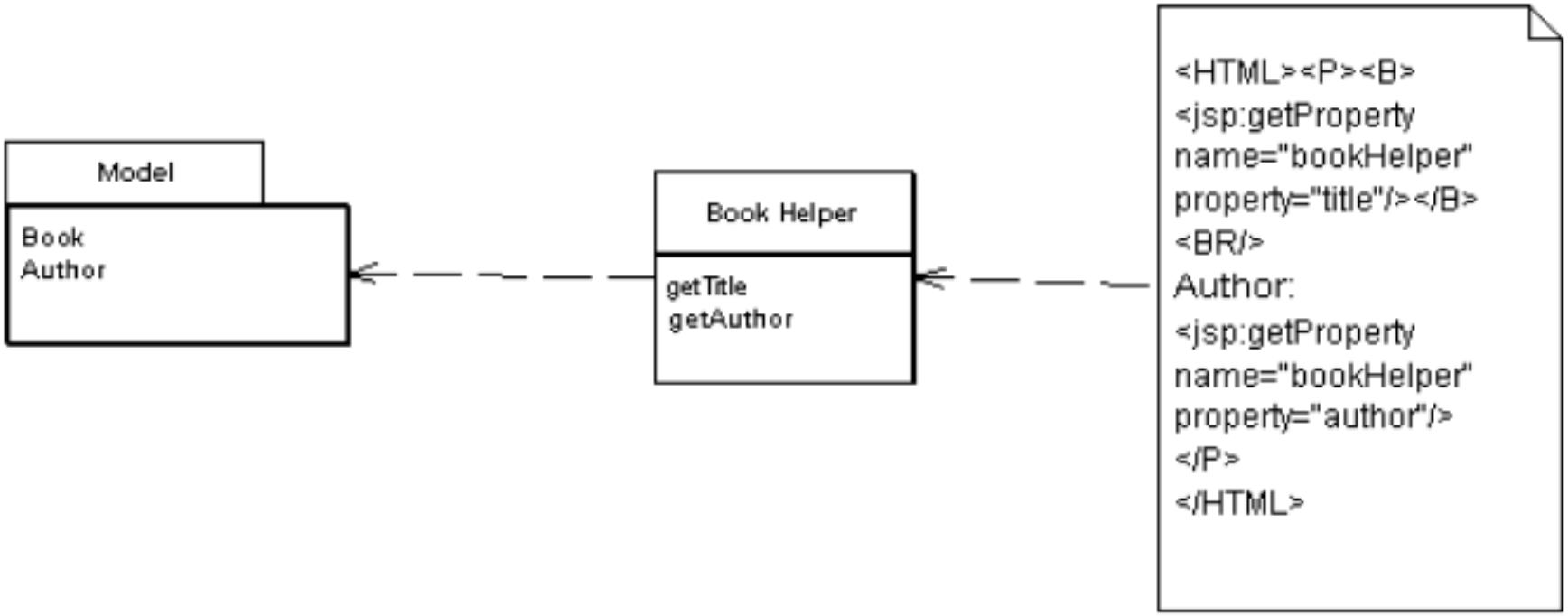
- Es más complejo que un Page Controller

DOS TIPOS DE PATRONES

PATRONES DE CONTROLADOR DE INPUT (INPUT CONTROLLER PATTERNS)	PATRONES DE VISTAS (VIEW PATTERNS)
○ Model View Controller ○ Page Controller ○ Front Controller	○ Template View ○ Transform View ○ Two Steps View

TEMPLATE VIEW

DESCRIPCION – Renderiza información en HTML embebiendo markers en una página HTML.



HTML Estático + Markers Especiales = Página Web Dinámica

TEMPLATE VIEW – COMO TRABAJA?

IDEA BÁSICA

- Embeber markers en una página HTML cuando ésta es escrita.
- La página puede ser diseñada de la manera usual, con editores WYSIWYG.

EMBEBIENDO LOS MARKERS

- Tags del estilo HTML
  - ✓ Trabajan bien con editores WYSIWYG
  - ✓ Con tags XML bien formados, se pueden usar herramientas de manejo de XML para editar las páginas, permitiendo que la página sea diseñada como HTML
  - ✓ Además de proveer un conjunto estándar de tags, muchos entornos permiten definir tags propios que satisfagan las necesidades.
- Text markers
  - Los editores WYSIWYG los ignoran, aunque a veces pueden meter ruido (spell checking)
  - La sintaxis puede ser más simple que los toscos XML / HTML

TEMPLATE VIEW – USO DE SCRIPTLETS

DEFINICIÓN

- Server pages (ASP, JSP, PHP) generalmente permiten embeber lógica de programación arbitraria en las páginas, llamada ***scriptlets.***
- Si bien pueden ser pragmáticos en algún sentido, conviene no abusar, limitándonos al comportamiento estándar de un Template View.

DESVENTAJAS

- Elimina la posibilidad de que los diseñadores gráficos diseñen las páginas.
- La página pierde su estructura y, con ello, su capacidad de modularización.
- Podrían llegar a confundir las diferentes capas de una aplicación empresarial.

TEMPLATE VIEW – NO USO DE SCRIPTLETS

HELPER OBJECTS

- Tienen toda la lógica de programación real.
- Una página solamente lo invoca, lo cual simplifica a la página y la hace más Template View pura.
- Es la manera de minimizar el uso de scriptlets
- Permite que los diseñadores gráficos diseñen la página, mientras que los programadores se concentran en los helpers

TEMPLATE VIEW – ROL

- Template View tiende a ser la *vista* en un MVC.
- En muchos sistemas, sólo jugará ese rol.
- En otros, generalmente más simples, podrá jugar el rol del *controller* y, posiblemente, el del *modelo* (aunque esto último es importante que se evite)
  - ✓ Es importante asegurar que estas responsabilidades sean manejadas por el helper, y no por la página, ya que involucran lógica de programación.

TEMPLATE VIEW – MANEJO DE EXCEPCIONES

CUIDADO!

- Los errores durante la compilación/interpretación de un Template View, generalmente no tienen red de contención.

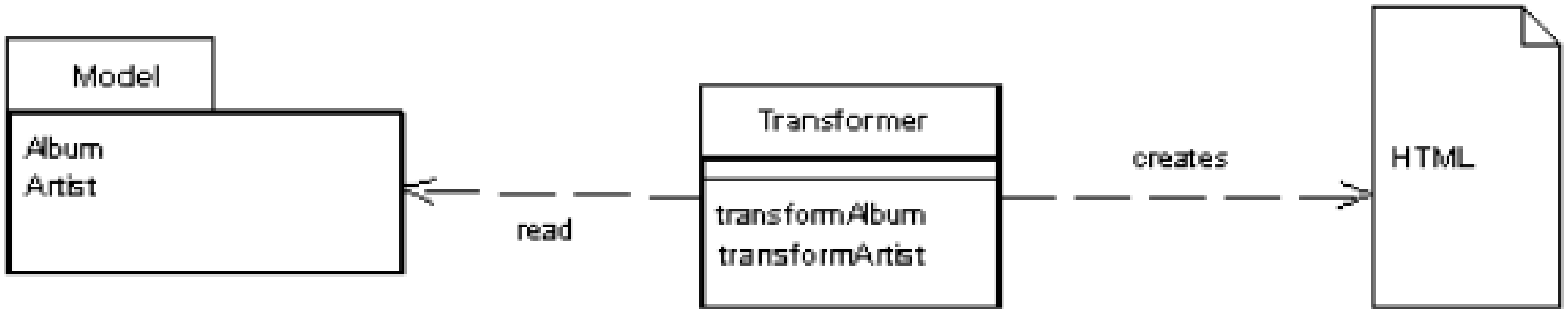
- Asegurar que la aplicación maneje estos errores.

TEMPLATE VIEW – CUANDO UTILIZARLO?

- 👍 Permite comprender el contenido de la página mirando la estructura de la página.
- 👍 Esto permite que diseñadores gráficos puedan trabajar con las páginas, enfocando a los programadores a la resolución de los helpers.
- 👎 La implementación más común hace muy fácil poner lógica complicada en la página, comprometiendo su mantenibilidad.
- 👎 Es más difícil de testear que Transform View

TRANSFORM VIEW

DESCRIPCION – Una vista que procesa datos del modelo elemento por elemento y los transforma en HTML



Usar Transform View para resolver la vista en MVC significa pensarlo como una transformación donde los datos del modelo conforman la entrada de dicha transformación y el HTML la salida.

TRANSFORM VIEW – COMO TRABAJA?

IDEA BÁSICA

- Escribir un programa que recorra los datos devueltos por el dominio y los convierta a HTML:
  - ✓ Atraviesa la estructura de datos del dominio
  - ✓ Reconoce a cada dato
  - ✓ Escribe la sección de HTML particular para el dato

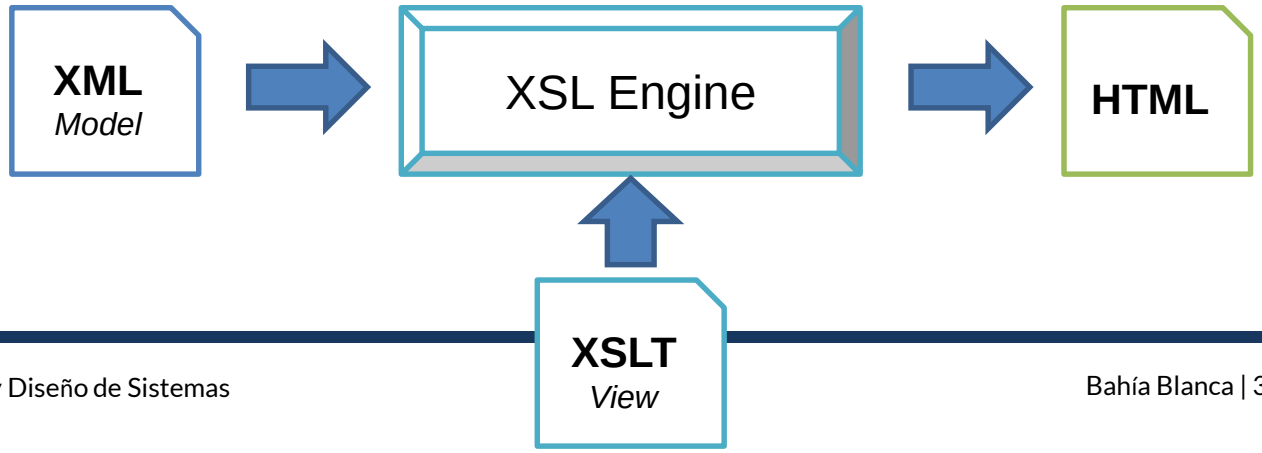
DIFERENCIAS CON TEMPLATE VIEW

- La forma en la cual la vista está organizada
  - ✓ Template View – orientada a la salida
  - ✓ Transform View – orientada a la entrada

TRANSFORM VIEW – IMPLEMENTACIÓN

XSLT es la más frecuente

- Es un lenguaje de programación funcional
- Tiene un tipo diferente de estructura – en lugar de explícitamente llamar a procedimientos, reconoce elementos en los datos del modelo e invoca las transformaciones apropiadas.
- Requiere que la entrada sea un XML
  - ✓ Si el tipo de retorno natural de la lógica de dominio es XML o algo automáticamente transformable a XML (por ej, objetos .NET)
  - ✓ Poblando un Data Transfer Object que sepa serializarse a XML.



TRANSFORM VIEW – CUANDO UTILIZARLO?

- XSLT es portable a casi todas las plataformas web.
- Es más simple de usar si los datos vienen como XML
- Evita dos grandes problemas de Template View:
  - Es más fácil enfocarse solamente en la renderización de HTML, evitando meter lógica en la vista
  - Es más fácil ejecutar la Transform View y obtener la salida para testearla.
- Si bien hay herramientas para trabajar con XSLT, no son tan sofisticadas como las existentes para Template Views.
- XSLT puede ser un lenguaje difícil de aprender por su naturaleza funcional.

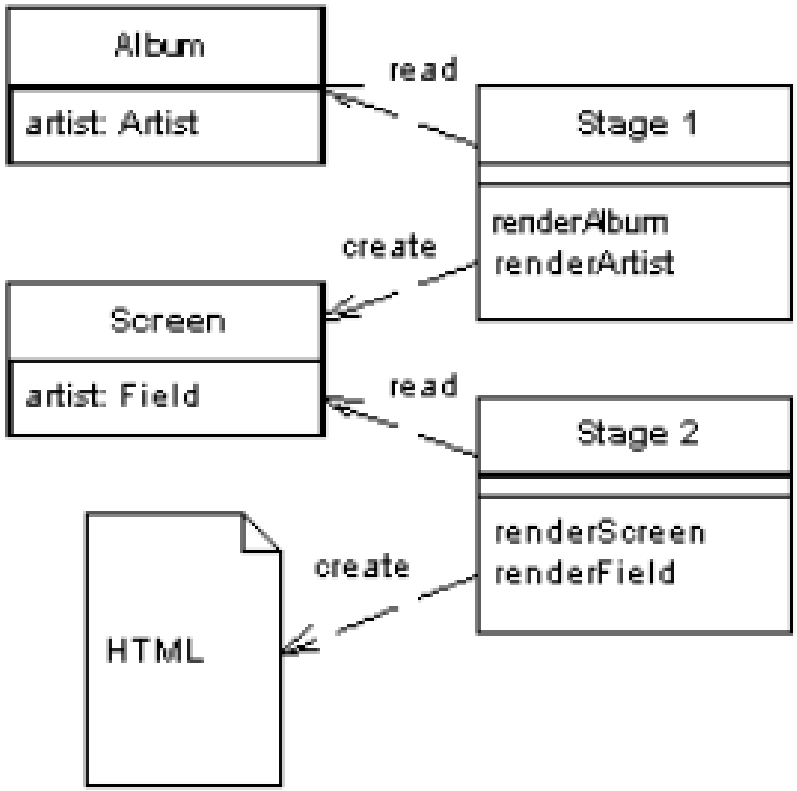
TWO STEP VIEW

DESCRIPCION – Transforma datos del dominio en HTML en dos pasos: primero construyendo algún tipo de página lógica, y luego renderizando la página lógica a HTML

Muchas veces se quiere realizar un cambio en el look & feel de la aplicación. Two Step View resuelve ese problema, separando la transformación en dos etapas:

1. Transforma los datos del modelo en una presentación lógica sin ningún formateo específico.

2. Convierte la representación lógica con el formato apropiado.



TWO STEP VIEW – COMO TRABAJA?

IDEA BÁSICA

- Hacer la transformación en un proceso de 2 etapas:

1) Ensamblar la información en una estructura de pantalla lógica que sea descriptiva de los elementos de pantalla sin usar HTML

    - Su responsabilidad es acceder al modelo orientado al dominio (una base de datos, un modelo de dominio real o un DTO), extraer la información relevante para la pantalla, y poner dicha información en una estructura orientada a la presentación.
    - Dicha estructura podría incluir campos, encabezados, pie de página, tablas, selectores, etc.

2) Tomar la estructura orientada a la presentación y renderizarla en HTML

    - Conoce cada elemento de la estructura orientada a la presentación y genera el HTML correspondiente.

TWO STEP VIEW – COMO TRABAJA?

IMPLEMENTACIONES

1) Two-step XSLT - Hay dos XSLT style-sheets:

   - Transforma el XML orientado al dominio en un XML orientado a la presentación
   - Transforma el XML orientado a la presentación a HTML

2) Clases

   - Se define la estructura orientada a la presentación como un conjunto de clases: una clase tabla, una clase fila, etc.

     ✓ Etapa 1: Toma la información del dominio e instancia estas clases en una estructura que modele una pantalla lógica.

     ✓ Etapa 2: Renderiza las objeto a HTML, haciendo que cada objeto genere su HTML o teniendo un generador separado.

TWO STEP VIEW – CUANDO UTILIZARLO?

- Permite realizar cambios globales con mayor facilidad.
- Resuelve dos situaciones particulares:
  - Aplicaciones web multi-apariencia - se necesita que cada cliente pueda determinar su look & feel
  - Aplicaciones web mono-apariencia - se necesita una look & feel consistente durante toda una aplicación
- Para variaciones de multi-apariencia, donde se pueden generar dos tipos distintos de HTML: para browser desktop y para mobile.
- Si tenemos una aplicación intensiva en diseño, donde cada página se ve distinto.
- Hay muy pocas herramientas para trabajar con esta alternativa.

RESUMEN

PATRONES DE PRESENTACION

INPUT CONTROLLER PATTERNS

- Model View Controller
- Page Controller
- Front Controller

VIEW PATTERNS

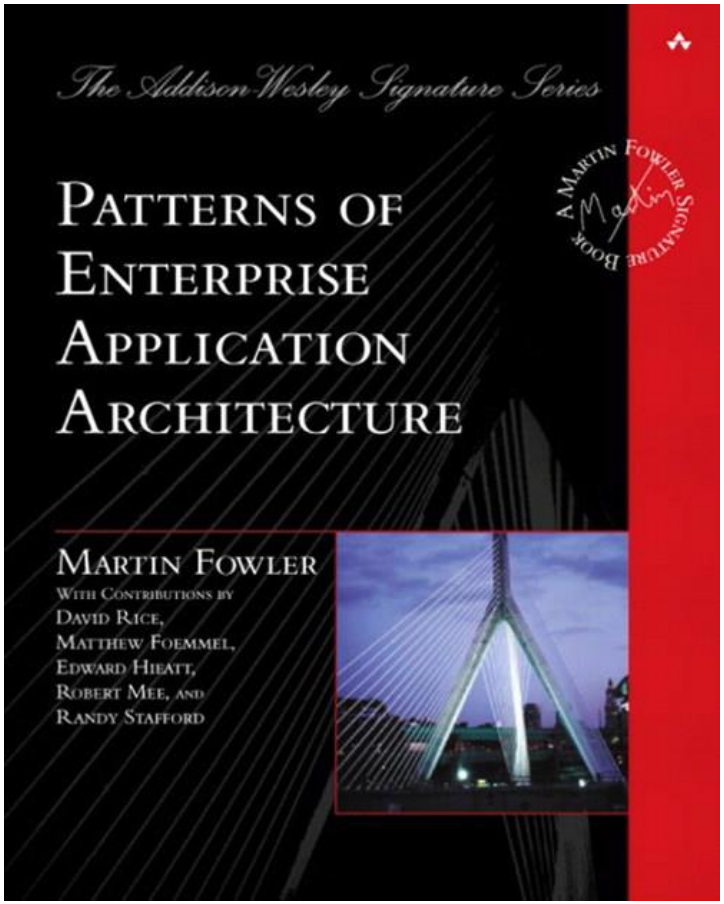
- Template View
- Transform View
- Two Step View

PARA VER

Making Architecture Matter – Martin Fowler

https://www.youtube.com/watch?v=DngAZyWMGR0

BIBLIOGRAFIA



Patterns of Enterprise Application Architecture (2002)

Martin Fowler
www.martinfowler.com

Elsa Estevez
ece@cs.uns.edu.ar