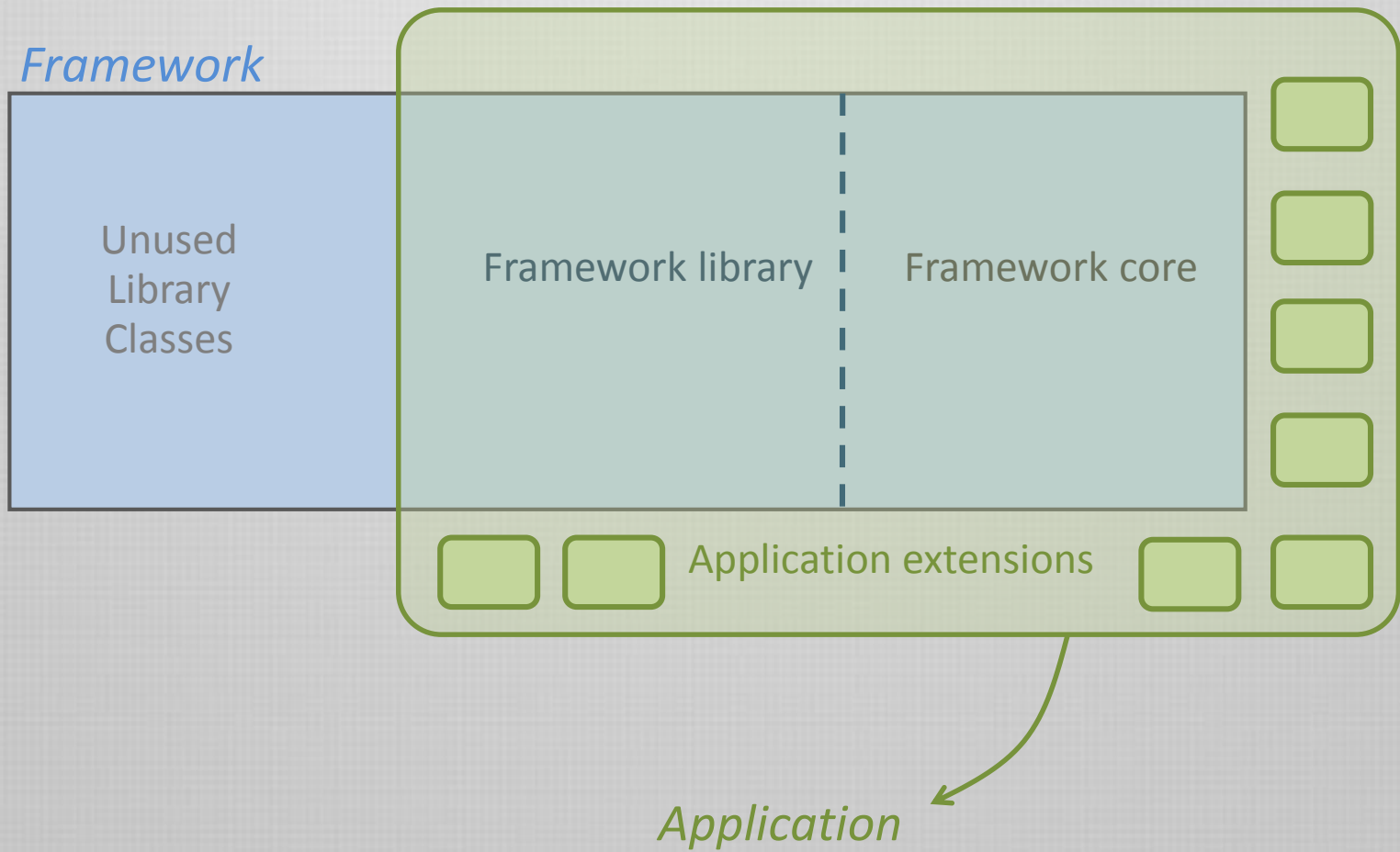


Ingeniería de Aplicaciones Web

Diego C. Martínez

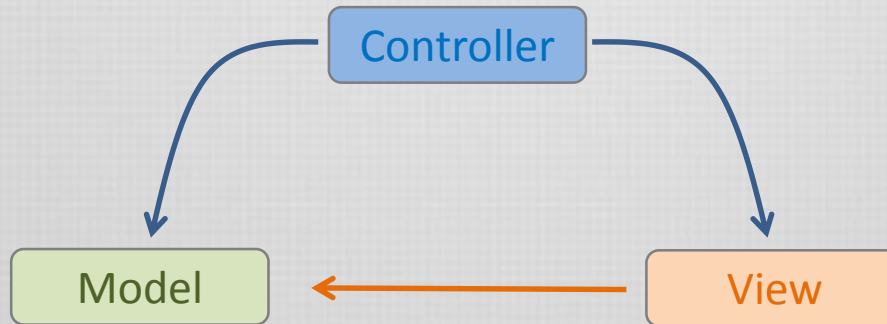
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Frameworks



Patrón general de diseño: MVC

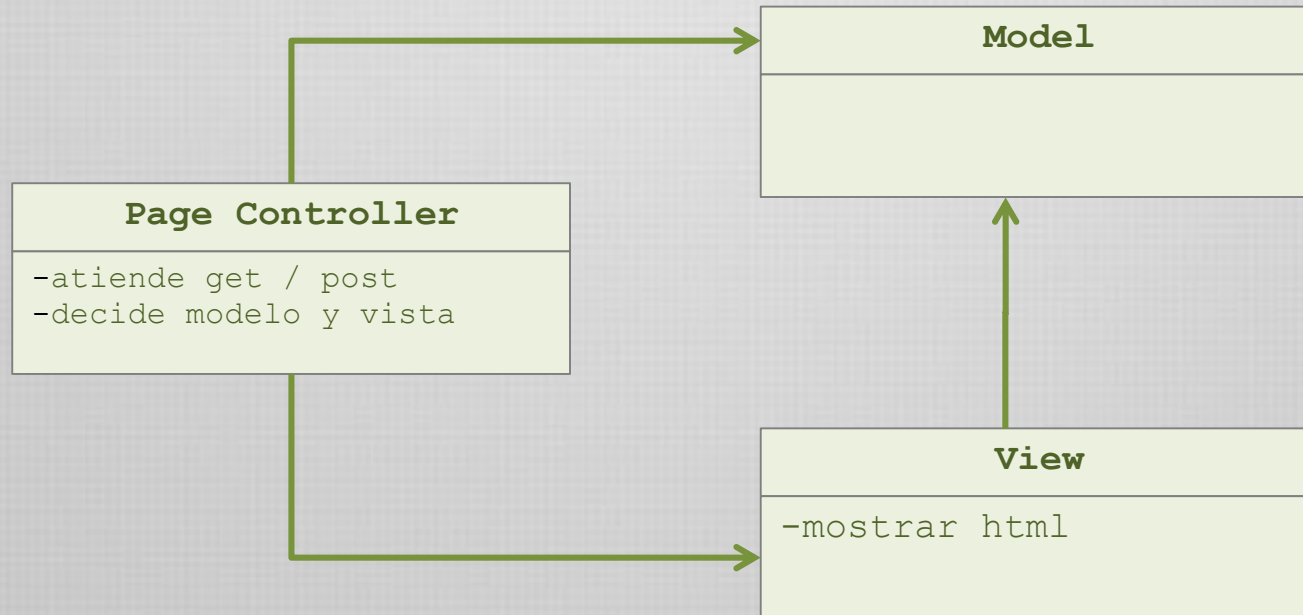
MVC es el patrón arquitectónico predominante en las aplicaciones web



*El **Controlador** administra el **Modelo** y la **Vista**.
La **Vista** es responsable de observar el **Modelo** para exteriorizar los datos.*

Page controller

Page-controller es un patrón de diseño congruente con el patrón arquitectónico MVC. Aquí un objeto controla el *request* de una página específica del sitio (action)

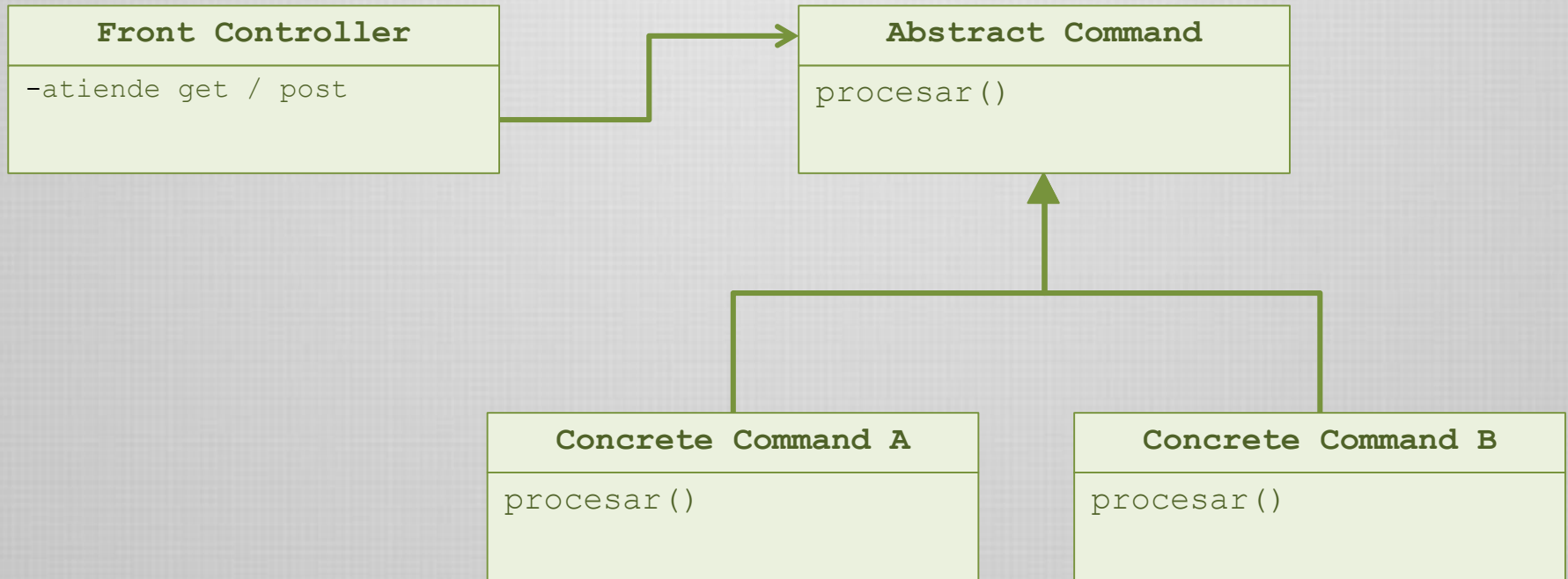


Hay un controlador por cada página lógica del sitio.
En algunos casos es la página misma.

Principalmente vinculado a una acción del sitio
Determina un conjunto de páginas parametrizables.

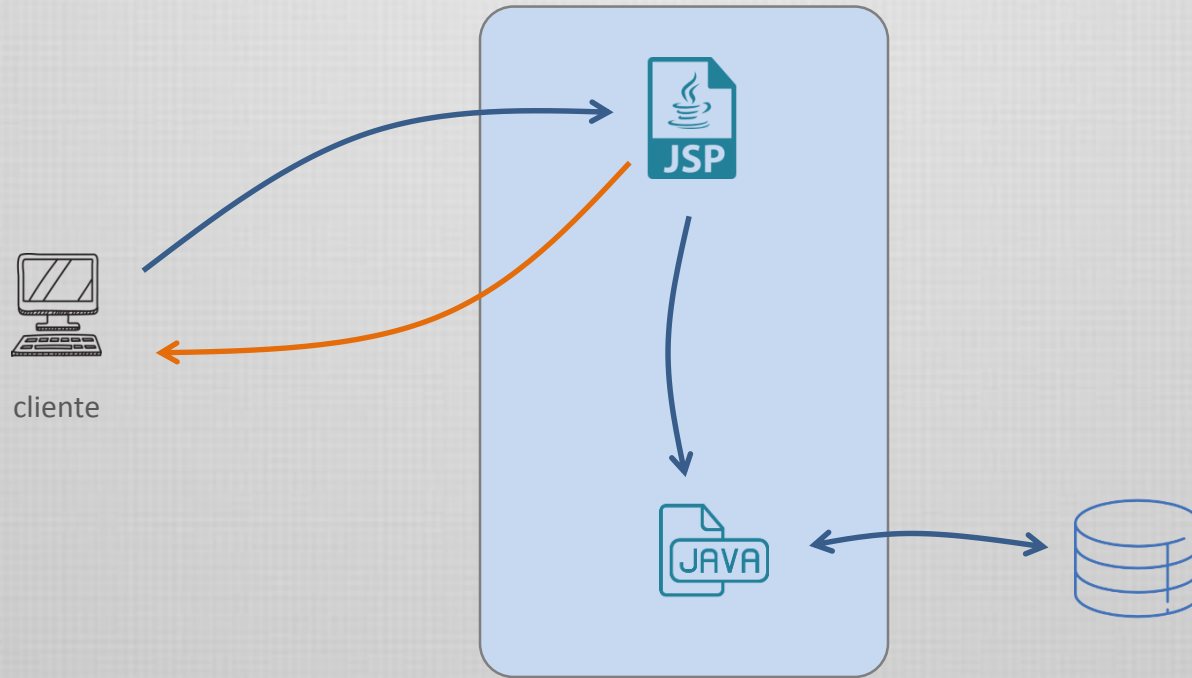
Front controller

Front-controller maneja *todos* los requests de un sitio web.

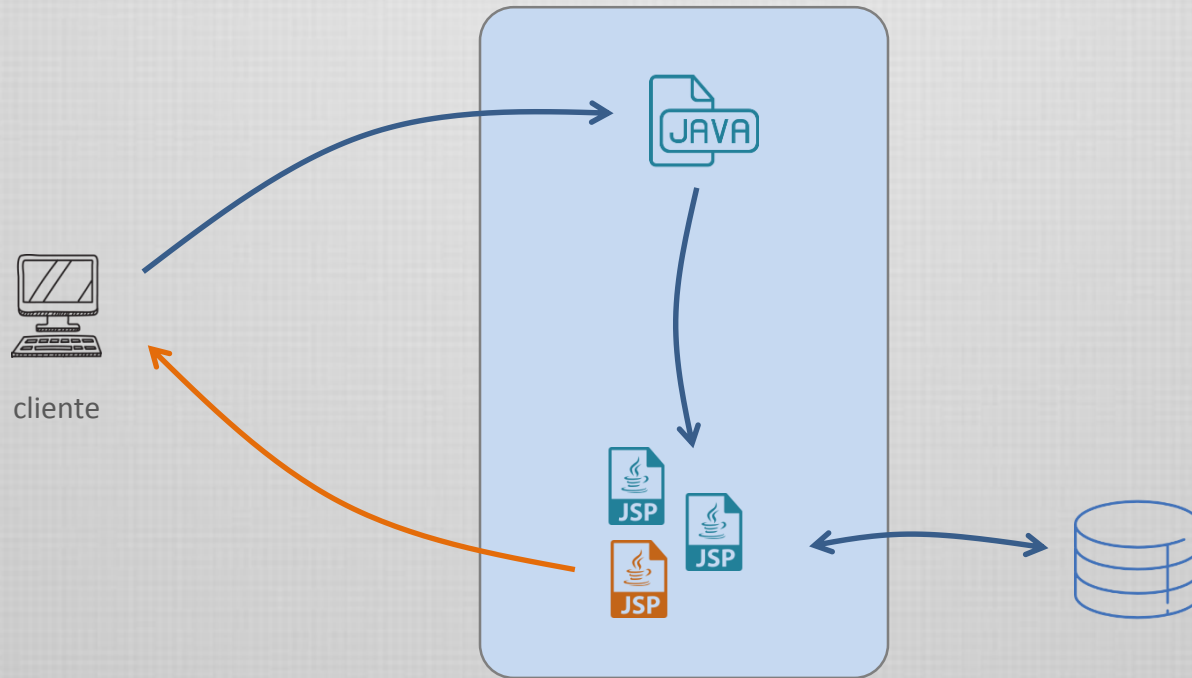


El objeto Front-controller recibe el request del cliente y
Recupera los datos correspondientes.
Prepara los datos para los comandos.
Decide qué comando ejecutar a continuación.

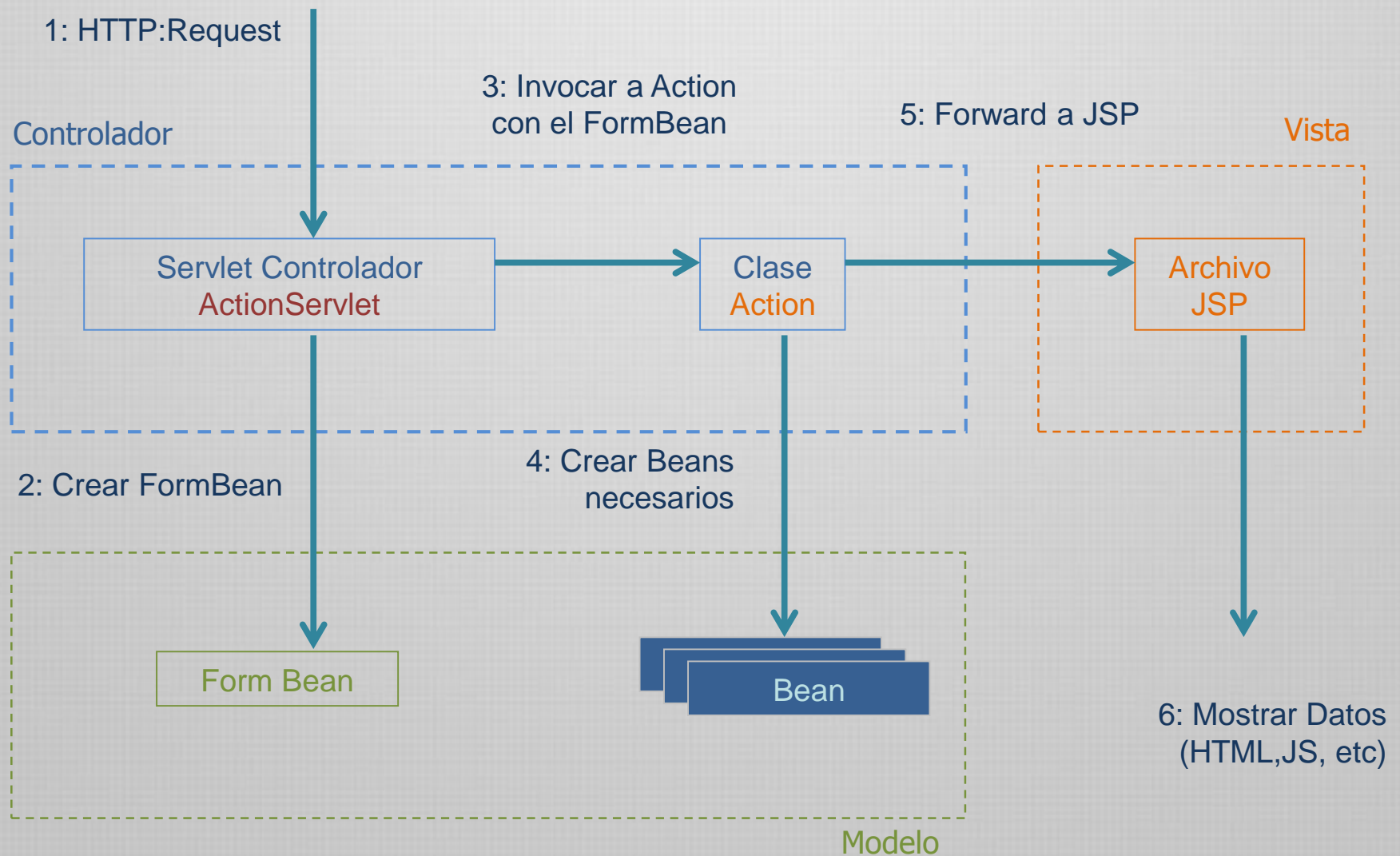
Java – JSP Model 1



Java – JSP Model 2



Struts – Déja Vu



Frameworks para la Web

¿por qué usar frameworks en la web?



menos
tiempo



menos
gastos



organización
del código



distribución
de tareas

¿qué suele ofrecer un framework para la web?

Uso de patrones de diseño efectivos

Arquitecturas definidas

Facilidades para la operatoria web

(formularios, validación, autenticación, soporte para UI, etc)

API

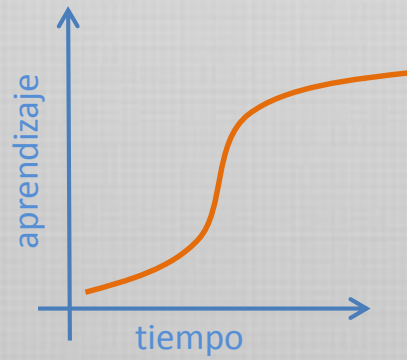
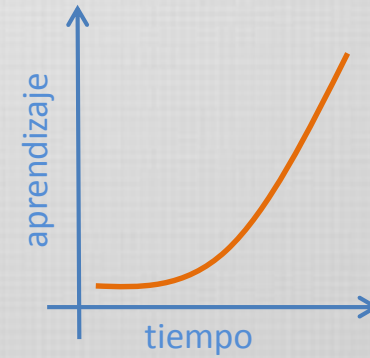
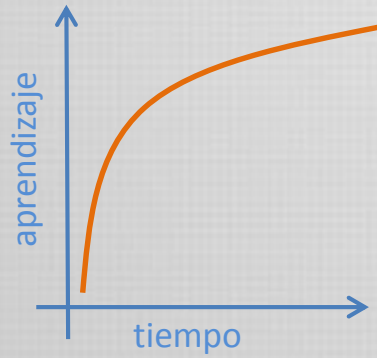
Administración de la capa de datos

Escalabilidad y performance

Comunidad de desarrolladores

Frameworks para la web




















Curva de Aprendizaje



PHP - Frameworks para Web

Existe una gran variedad de frameworks, para diferentes lenguajes.

En el caso puntual de PHP, la oferta es extensa y variada.

PHP Framework	PHP4	PHP5	MVC	Multiple DB's	ORM	DB Objects	Templates	Caching	Validation	Ajax	Auth Module	Modules	EDP
Akelos 	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
ash.MVC 	-	✓	✓	-	-	✓	✓	-	✓	-	✓	✓	-
CakePHP 	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-
CodeIgniter 	✓	✓	✓	✓	-	✓	✓	✓	✓	-	-	-	-
DIY 	-	✓	✓	-	✓	✓	✓	✓	-	✓	-	-	-
eZ Components 	-	✓	-	✓	-	✓	✓	✓	✓	-	-	-	-
Fusebox 	✓	✓	✓	✓	-	-	-	✓	-	✓	-	✓	-
PHP on TRAX 	-	✓	✓	✓	✓	✓	-	-	✓	✓	-	✓	-
PHPDevShell 	-	✓	-	-	-	-	✓	-	-	✓	✓	✓	-
PhpOpenbiz 	-	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	-	-
Prado 	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
QPHP 	✓	✓	✓	✓	-	✓	✓	-	✓	✓	✓	✓	✓
Seagull 	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
Symfony 	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-
WACT 	✓	✓	✓	✓	-	✓	✓	-	✓	-	-	✓	-
WASP 	-	✓	✓	-	-	✓	✓	-	✓	✓	✓	✓	-
Yii 	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Zend 	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
ZooP 	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	-	-

ORM = object recorder mapper

DB Objects = objetos modelo de elementos de bases de datos

Auth Module = incluye módulo de autenticación de usuarios.

Modules = otros modulos de utilidad general: RSS, PDF

EDP = Programación orientada/guiada por eventos

PHP - Frameworks para Web

Lightweight vs Heavyweight

ofrecen funcionalidad básica y estructura simple.

mayor funcionalidad y dependencia del framework

Licencias

GPL, Open Source, BSD, MIT, etc

Estandarización

www.php-fig.org

Framework Interoperability Group

Es un consorcio de desarrolladores PHP que determinan estándares generales

(coding style, logging, autoloading, etc)

Web Frameworks: CakePHP



Framework *open source* para el *desarrollo rápido de aplicaciones* web escritas en lenguaje PHP.

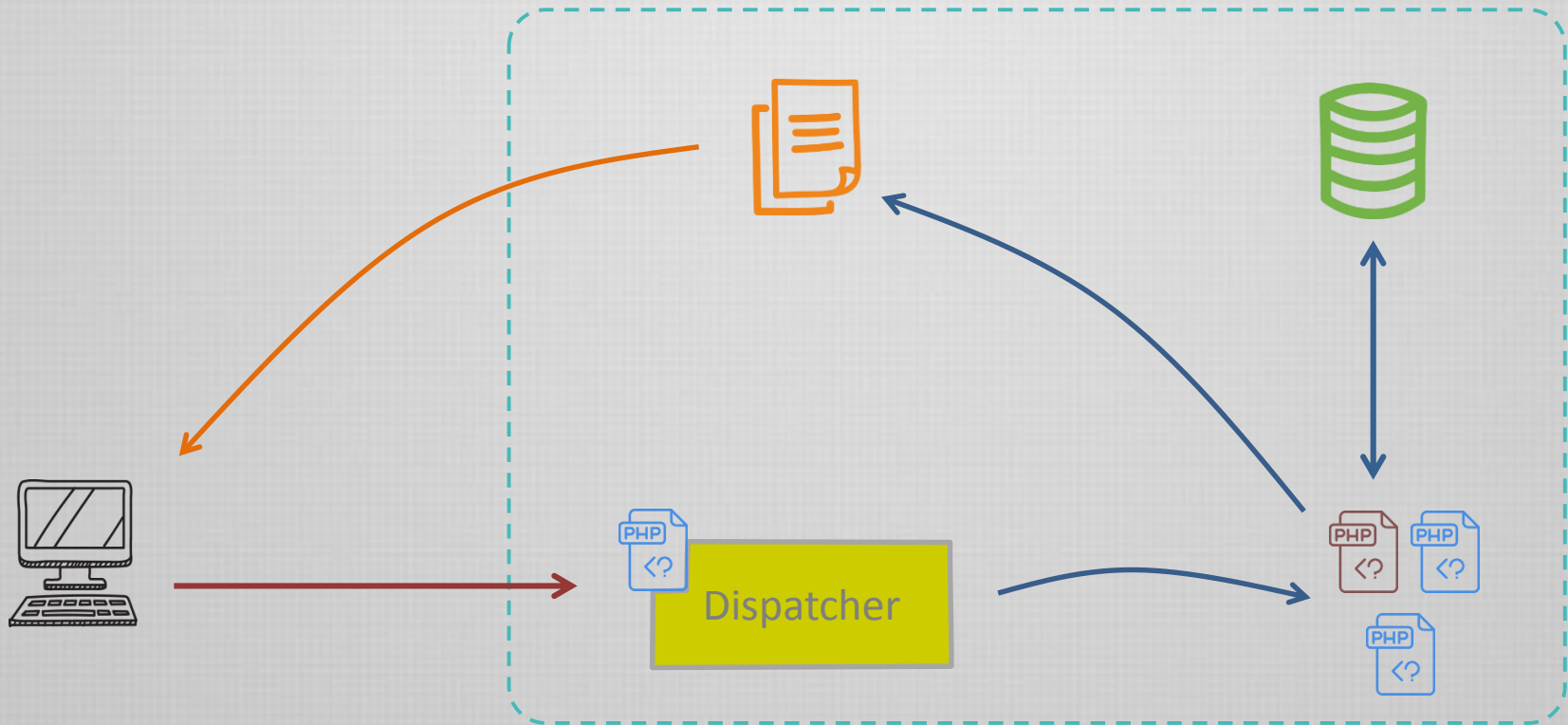
Es una “*estructura de librerías, clases y una infraestructura de ejecución*” basada en el framework *Ruby on Rails* (para Ruby).

Se distribuye bajo *Licencia MIT*

Basado en
MVC

- ▶ **Model:** abstracción de datos. Interfaz con bases de datos y configuración relacionada, como las reglas de validación de los datos.
- ▶ **View:** básicamente HTML con código PHP, pero en archivos que serán procesados por el framework (“view files”).
- ▶ **Controller:** determina el flujo del procesamiento, modifica el modelo, redirecciona a la vista correspondiente.

Web Frameworks: CakePHP

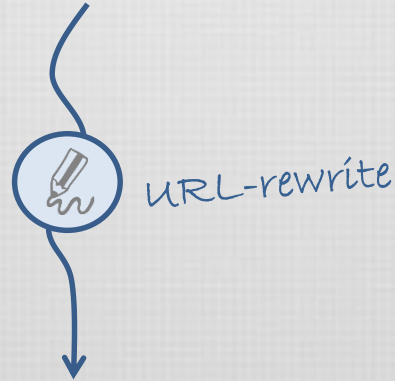


La interacción del cliente con el servidor se interpreta como pedidos de acciones. En la URL se menciona el controlador y la acción deseada.

<http://nuestro.host.com/controladorA/accionZ>

Web Frameworks: CakePHP

`http://nuestro.host.com/controladorA/accionZ`



`http://nuestro.host.com/app/webroot/index.php?url=...`

Los servidores ofrecen la funcionalidad de **reinterpretar una URL** en otra con sentido local

En apache: mod_rewrite

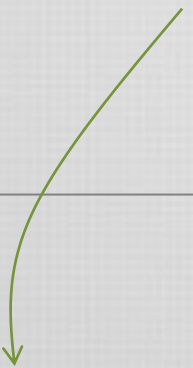
En IIS: Rewrite Module

“Pretty URL”


CakePHP – componentes típicos - *models*

```
class Item extends AppModel
{

}
```



Hereda operaciones predefinidas para crear consultas, agregar y eliminar instancias de datos modelados.



Los modelos representan *modelos de datos*.
Usualmente es una tabla de la base de datos, pero puede modelar datos de otro tipo.

*Se ubican todos en la carpeta **src/Models**.
Deben tener el mismo nombre de la clase*

\$useDbConfig: qué configuración de DB usar.

\$primaryKey: clave primaria de la tabla a utilizar.

\$useTable: tabla a usar

Una vez creado el modelo, puede accederse desde el controlador.

CakePHP garantiza la disponibilidad siguiendo convenciones de nombres.

Pueden “cargarse” explícitamente también dentro de un controlador.

CakePHP puede incluso crear un modelo on-the-fly, siguiendo convenciones de nombres.

CakePHP – componentes típicos - *controllers*

```
class MiController extends AppController
{
    function accion1()
    { ... }

    function accion2()
    { ... }
}
```

Los controladores incluyen operaciones denominadas comúnmente *acciones*.

Se ubican en la carpeta
src/Controllers

ver un ítem de base de datos
listar ítems
loguear un usuario
validar datos de un formulario

Usualmente, un controlador implementa la lógica para *un modelo*.

Llevan entonces el nombre en *plural* del *modelo* que administran.

Al ser los controladores de la aplicación, las acciones pueden accederse via URL directamente:

http :// misitio.com/nombre/accion1

http :// misitio.com/nombre/accion2

nombre_controller.php

CakePHP – componentes típicos - *vistas*

```
<h1>Una Vista</h1>
<table>
  <tr> <th>Campo1</th>          <th>Campo2</th>          ...      </tr>
  <?php foreach ($items as $item): ?>
    ...
  <?php endforeach; ?>
</table>
```

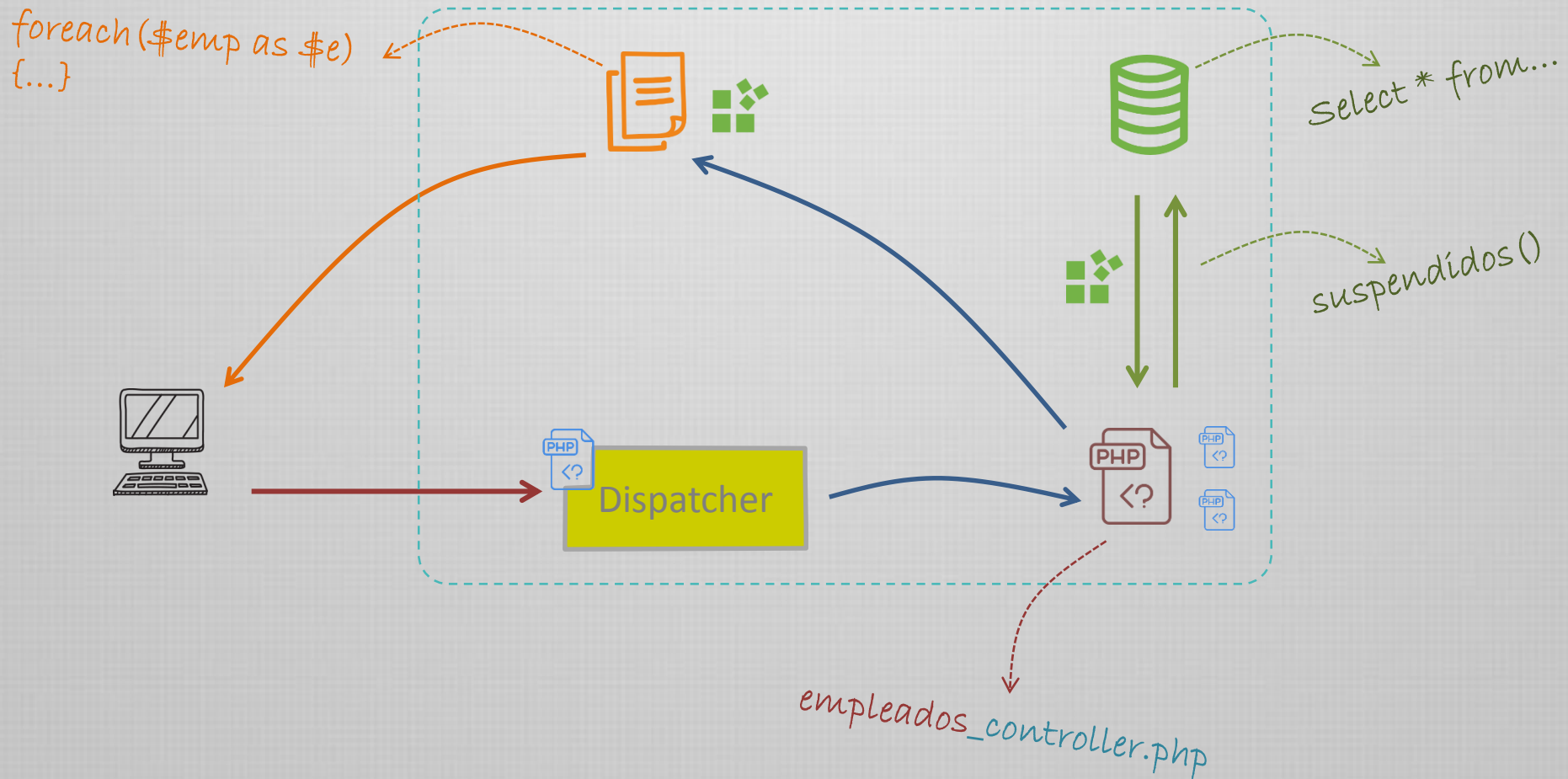
Las vistas generan la visualización de las acciones (sus resultados).

Es la comunicación final con el usuario.

Los archivos de vistas tienen contenido (X)HTML.
Se ubican todas en la carpeta *app/views*.

Web Frameworks: CakePHP

`http://nuestro.host.com/empleados/suspendidos`



Web Framework: CakePHP

“conventions over configurations”



Nombres de archivos, *CamelCase*. Nombres de clases, *CamelCase*:

Los modelos, en singular: *Alumno*, *Materia*, *Curso*, *Profesor*

Los controladores, en plural + "*Controller*". La acción por defecto es *index()*.

Las tablas de bases de datos, en plural y con underscore.

Las vistas son nombradas según la acción del controlador

Tabla de la base de datos: “*alumnos*”

Clase modelo: “*Alumno*”, en el archivo ***alumno.php***

Clase controlador: “*AlumnosController*”, en el archivo ***AlumnosController.php***

Vista: en el archivo ***index.ctp***