6. Cap a un framework MVC

Desenvolupament web en entorn servidor

Vicent Jordà

21/11/2020

Índex de continguts

1 El controlador frontal **3**

1.1 Què posa ahí? URL amigables 3

2 Routing **8**

2.1 La classe Router 8

2.2 Classe Router 8

2.3 Classe Request 8

2.4 Classe FilteredMap 8

2.5 Classe DependencyInjection 9

2.6 Què necessites? La injecció de dependències 9

2.7 Contenidor de serveis 9

2.8 Webgrafia i recursos 10

1 El controlador frontal

El controlador frontal és un patró de disseny que apareix en diversos catàlegs de patrons i està relacionat amb el disseny d'aplicacions web. És "un controlador que gestiona totes les sol·licituds d'un lloc web", que és una estructura útil per als desenvolupadors d'aplicacions web per a aconseguir la flexibilitat i la reutilització sense redundància de codi.

Normalment el controlador frontal s'implementa en la pàgina principal del lloc web: `index.php`.

Perquè el controlador frontal siga funcional és important que qualsevol petició al lloc web siga redirigida al controlador frontal. En eixe sentit les URL amigables tenen un paper esencial.

1.1 Què posa ahí? URL amigables

Les URL semàntiques o URL amigables són aquelles URL que, dins del que cap, poden ser enteses pels usuaris.

Lluny de les clàssiques URL de les pàgines dinàmiques plenes de variables en el *querystring* i nombres difícils de recordar, les URL semàntiques estan formades amb paraules relacionades amb el contingut de la pàgina i fàcils de recordar.

Aquestes s'utilitzen en els llocs web dinàmics (no estàtics). Per això s'estan utilitzant molt més que les URL extenses.

Suposem que vols comprar una càmera fotos. Has trobat dues botigues amb el model que busques.

La direcció de l'producte a la primera botiga és:

```
1 http://www.example.com/B001G5ZTLS/ref=sr_1_5?s=foto&ie=UTF8&qid
     =1365525726&sr=1-5
```

I la direcció de la segona:

```
1 http://www.example.com/camaras/reflex/canon-eos-5d-mark-2/
```

Quan desenvolupem una aplicació web hem de tractar que les url tinguen càrrega semàntica.

Els seus principals avantatges són:

- Fàcils de recordar i de deduir.
- Oculta de la tecnologia usada.
- És conforme a l'estil REST el que li dona consistència.

- Crear URL netes i llegibles que són més fàcils de recordar i més adequades per al Search Engine Optimization (SEO).

En el nostre projecte quina url tindrà més càrrega semàtica?

```
http://movies.local/movies/title/ava
```

o

```
http://movies.local/show-movie.php?id=1488
```

Reescriptura de rutes

Com hem dit abans, per a disposar d'un lloc web en url amigables necessitem que totes les peticions a la la nostra aplicació es redirigisquen al nostre controlador frontal (`index.php`)

En `index.php` obtindrem el *path* (ruta) i cridarem al controlador adient.

Per redirigir totes les cridades al controlador frontal haurem de configurar el mòdul de reescriptura d'Apache (MOD_REWRITE).

Objectius de les regles de reescriptura

Els objectius són diversos:

- Amagar la funcionalitat de PHP i, per tant, exposa menys les parts internes de el lloc.
- Restringir l'accés.
- Crear URL netes i llegibles que són més fàcils de recordar i més adequades per al Search Engine Optimization.

La reescriptura d'URL és la tècnica utilitzada per "traduir" un URL amigable a una altra que el servidor puga entendre.

Directives

Perquè la reescriptura d'URL funcione tenim dues opcions:

- Introduir una sèrie de directives <`Directory`> en els fitxers de configuració d'Apache

- Introduir aquestes directives en un fitxer `.htaccess` dins del directori en el que vulgues que es realitze la reescriptura.

Nosaltres utilitzarem la segona opció.

D'aquesta manera, en producció n'hi haurà prou amb pujar el `.htaccess` al servidor i no necessitarem tocar la configuració.

> Perquè funcione `.htaccess` cal tindre activida la directiva `AllowOverride` d'Apache.

Els passos a realitzar són:

- Comprovar si existeix el `mod_rewrite`
- Totes les directives de `mod_rewrite` les introduirem dins d'una directiva IfModule:

```
<IfModule mod_rewrite.c>
    Directives de reescriptura
</IfModule>
```

DESHABILITAR L'OPCIÓ MULTIVIEWS*

És convenient desactivar aquesta opció per evitar problemes amb fitxers que tenen noms semblants, teniu més informació en What exactly does the multiviews option in htacces

```
<IfModule mod_rewrite.c>
    options -MultiViews
    Directives de reescriptura
</ IfModule>
```

Activar mod_rewrite

Per indicar a Apache que fem servir el `mod_rewrite`:

```
RewriteEngine on
```

Si vols desactivar-lo i que el servidor ignore la resta de la configuració:

```
RewriteEngine off
```

RewriteCond

Ens permet especificar una condició.

Si es compleix, s'executa la regla RewriteRule posterior.

Es poden posar diverses condicions.

Quan es compleixen totes les condicions (llevat que s'indique una altra cosa) s'executa la directiva `RewriteRule` posterior

Sintaxi:

RewriteCond variable_apache expresión_regular banderes

Variables per a condicions

Selecció d'url

Després de la variable indicarem una expressió regular per seleccionar les adreces que es corresponen amb aquest patró.

Existeixen també una sèrie de variables predefinides que es poden utilitzar en comptes d'introduir una expressió regular:

- `-d`: es tracta la URL com si fos una ruta d'el sistema d'arxius i es comprova si existeix i si és un directori.
- `-f`: igual que l'anterior, però comprovant si és un fitxer.
- `-l`: Igual que els anteriors, però comprovant si és un enllaç simbòlic. Etc.

Si indiquem el símbol `!` Davant de l'expressió regular o la variable, l'estem negant, és a dir, que s'avaluarà la condició a cert quan no coincideixi amb l'expressió indicada.

** Flags (banderes) **

Podem indicar una llista de flags per especificar un comportament determinat:

nocase | NC: El testeig és insensible a majúscules i minúscules(no *case-sensitive*). ornext | OR: S'utilitza l'operació OR (en lloc de AND que és l'opció per defecte) per combinar el resta de condicions en les regles

Per exemple:

```
1 RewriteCond% {HTTP_USER_AGENT} ^exemple[OR, NC]
2 RewriteCond% {HTTP_USER_AGENT} ^google[NC]
```

La variable% {HTTP_USER_AGENT} representa el nom del navegador que utilitza l'usuari.

L'expressió regular s'utilitza per determinar quin és el valor de la variable que estem analitzant

^Exemple -> el nom del navegador comença per exemple.

En els flags s'indica que les dues condicions s'agrupen com operacions OR.

El nom de el navegador ha de començar per `exemple` o per `google`.

El valor NC indica que no ha de distingir entre majúscules i minúscules.

RewriteRule

Aquesta directiva serà l'encarregada de realitzar la redirecció (reescriptura) corresponent.

Sintaxi:

```
RewriteRule expresión_regular redirecció flags
```

Mitjançant el caràcter "-" indicarem que no volem cap pàgina de redirecció

El nostre cas

```
<IfModule mod_rewrite.c>
    Options -MultiViews
    RewriteEngine On

    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^(.*)$ /index.php [QSA,L]

</IfModule>
```

`RewriteCond` indica qualsevol sol·licitud que no siga un fitxer.

I en `RewriteRule` l'expressió regular selecciona qualsevol URI.

Ja que:
* ^ comença i $ acaba. * () agrupa * . qualsevol caràcter excepte espai. * * repetició del caràcter anterior.

Els flags indiquen:

- QSA, les *query strings*, d'existir, es combinen en la reescritura.
- L, indica que aplicada la regla, s'ature l'execució.

2 Routing

Una de les tasques que hem de realitzar a l'hora d'analitzar un projecte és com anem a organitzar la taula de rutes.

Una ruta serà la url que ens donarà accés a un controlador.

El que farem serà associar les dades de la URL amb l'script encarregat de resoldre aquesta operació.

2.1 La classe Router

És convenient encapsular l'operativa amb la taula de rutes dins d'una classe que siga la responsable d'aquesta tasca (la classe Router). Aquesta classe tindrà l'array de rutes com a atribut.

També tindrà dos mètodes:

- Un que ens permetrà carregar les rutes de la taula.
- Un altre que ens permetrà tornar el controlador encarregat de gestionar una ruta.

La ubicarem dins de la carpeta core. La seva responsabilitat serà gestionar tot el que tinga a veure amb les rutes de l'aplicació.

2.2 Classe Router

- Carrega les rutes.
- Enruta. Segons el path s'executa el controlador.

```
1 $router = new Router($di);
2 $router->route($request)
```

2.3 Classe Request

La ubicarem dintre de Core. Serà la responsable de gestionar les peticions.

2.4 Classe FilteredMap

Aquest classe és una utilitat que ens facilita d'accés als paràmetres. $id=filer_input(INPUT_GET, 'id', FILTER_VALIDATE_INT); $id =$request→params→getInt('id');

2.5 Classe DependencyInjection

Es tracta d'un contenidor de serveis. Simplement associa una clau amb una dependència (objecte). Després passarem l'objecte DI a l'enrutador (router).

2.6 Què necessites? La injecció de dependències

La injecció de dependències (*dependency injection*) és un patró de disseny orientat a objectes en què es subministren els objectes a una classe en lloc de ser la pròpia classe la que creu aquests objectes.

Aquests objectes compleixen contractes (interfícies, classes abstractes) que necessiten les nostres classes per poder funcionar (d'aquí el concepte de dependència).

Les nostres classes no creen els objectes que necessiten, sinó que se'ls subministra una altra classe 'contenidora' que injectarà la implementació desitjada al nostre contracte.

Simplificant:

> La injecció de dependències proporciona a una classe les seves dependències ja siga mitjançant la injecció del constructor, crides a mètodes o l'establiment de propietats.

Exemple: https://github.com/Seldaek/monolog/blob/master/doc/01-usage.md

Exemple

2.7 Contenidor de serveis

Utilitat que ajuda a la implementació de la injecció de dependències.

La classe App

```
1  class App
2  {
3      private static $container = array();
4      public static function bind($key, $value) {
5          static:: $container [$key] = $value;
6      }
7  }
8  public static function get($key)
9  {
10     if(! array_key_exists ($key, static:: $container )) {
```

```
11          throw new Exception("The $key doesn't in the container");
12      }
13      return static:: $container [$key];
14  }
15  }
```

2.8 Webgrafia i recursos

- WIKIPEDIA CONTRIBUTORS. Front controller [en llínia].https://en.wikipedia.org/w/index.php?title=Front_controller&oldid=965938044. Data de consulta: 20/11/2020.