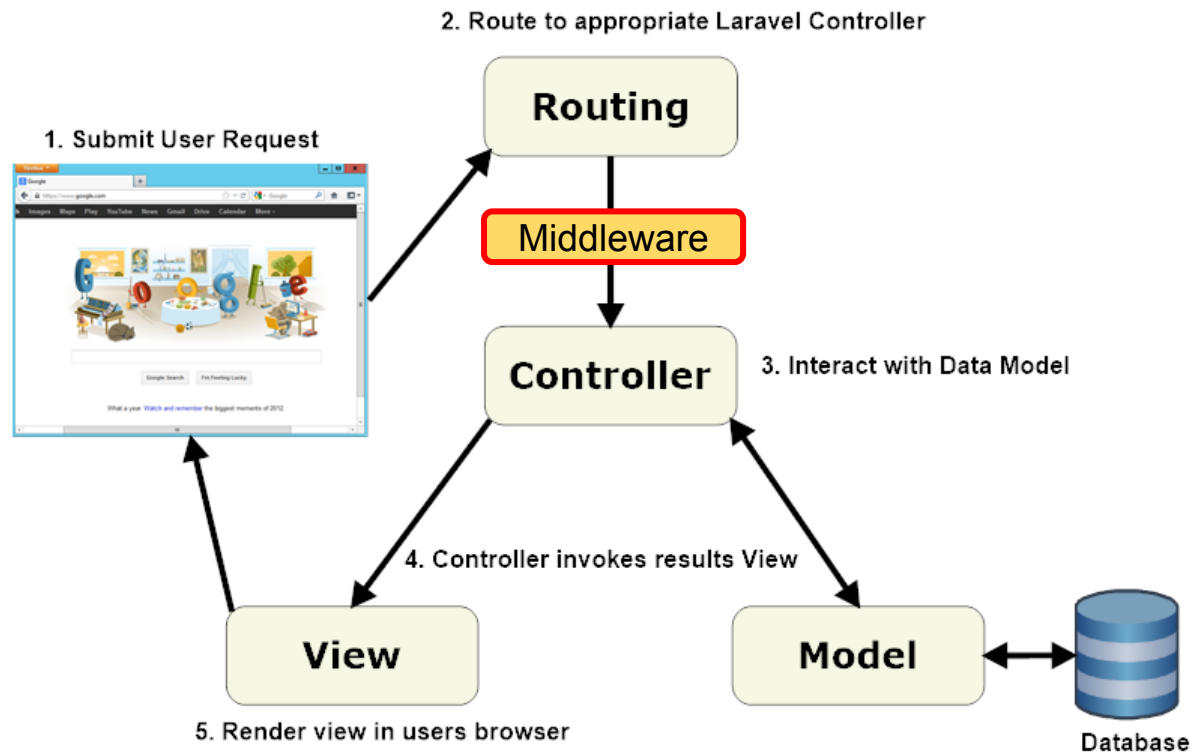


# Desarrollo de aplicaciones web con Laravel 5

**Clase 3. Middleware, providers, containers,  
facades, contracts y Modelos**  
Carlos Ruiz Ruso • @micromante

# HTTP Middleware

# MVC con Ruteo



# Middleware

- app/Http/Middleware
- Evolución de los antiguos filters de Laravel 4.
- Completo mecanismo de filtrado HTTP en la aplicación.
- Uso básico
  - Por defecto, autenticación de usuarios
    - Si el usuario es válido pasa, si no redirige a login
- Extensión
  - Por supuesto no es su única tarea, puede añadir cabeceras a las peticiones, hacer log de las peticiones, mantenimiento, CSRF y mucho más.



# Definición de un Middleware

- Podemos crear un nuevo middleware

```
php artisan make:middleware TestMiddleware
```

# Estructura básica

```
<?php namespace App\Http\Middleware;

use Closure;

class OldMiddleware {

    /**
     * Run the request filter.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        if ($request->input('age') < 200)
        {
            return redirect('home');
        }

        return $next($request);
    }
}
```

Objeto de la petición

Condiciones de filtrado

Salto al siguiente middleware o controller. Al pasarle \$request el controlador puede hacer uso de ello.

# Antes y después de la petición

- El middleware puede ser ejecutado antes o después de la Request.
- Simplemente añadimos After o Before delante del nombre de la clase
  - BeforeTestmiddleware
  - AfterTestMiddleware



# Registrando el Middleware

- Globalmente
  - Añadimos el middleware a `app/Http/Kernel.php`
- Asignación a una ruta
  - `Route::get('usuario/perfil', ['middleware' => 'auth'], function(){ ... });`
  - Como vimos en la clase de rutas, podemos añadir mas de un middleware y el orden influye en la ejecución.
- Constructor o método en Controller
  - `$this->middleware('auth');`



## Middleware de finalización

- En algún caso podremos necesitar lanzar un middleware cuando la respuesta HTTP ya esté en el cliente. En este caso podemos usar un tipo diferente de Middleware.
- Por ejemplo, la “session” se almacena cuando la respuesta se ha realizado, por eso usamos los TerminableMiddleware.
- **TerminableMiddleware** son métodos de terminación y recibe tanto la `$request` como la `$response`. Esta clase debe estar siempre configurada de manera global en el middleware kernel.

# Service Providers

# Service Providers

- Servicios del sistema Laravel
- Parte central e importante para el “Bootstrapped” (registro de arranque)
- Incluye: enlazado (bindings), eventos, listeners (escuchadores), filters (filtros) y Routing.
- Podemos encontrarlos en config/app.php en un array de servicios.
- Estos servicios son cargados por la aplicacion, esto no significa que se cargan en todas las peticiones, son los servicios “necesarios”.
- Podemos crear nuestro Provider básico
  - `php artisan make:provider TestServiceProvider`
- Estan creados bajo App\Providers



# Metodos Service Providers

- **Register**
  - Define una implementación en el service container (Ejemplo Repository)
    - Service container = resolución dependencias clases e inyección
- **Boot**
  - Se lanza cuando el resto de providers estan registrados, podemos acceder a todos los servicios registrados.
  - Podemos registrar eventos
    - `Event::listen('nombredeevento','TestEventHandler');`
    - `$events->listen(...) ← Dispatcher $events //Injection`
    - `protected $listen = ['nombredelvento' => 'TestEventHandler' ,....]`

# Ejemplo mapeo de Rutas

- Binding de las rutas (por defecto)

```
<?php namespace App\Providers;

use Illuminate\Routing\Router;
use Illuminate\Foundation\Support\Providers\RouteServiceProvider as ServiceProvider;

class RouteServiceProvider extends ServiceProvider {

    protected $namespace = 'App\Http\Controllers';


    public function boot(Router $router)
    {
        //
        parent::boot($router);
    }

    public function map(Router $router)
    {
        $router->group(['namespace' => $this->namespace], function($router)
        {
            require app_path('Http/routes.php');
        });
    }
}
```

# Ejemplo implementar interfaz

- Definir implementación de una interfaz de pagos

```
<?php namespace App\Providers;
use Illuminate\Support\ServiceProvider;
class PagosServiceProvider extends ServiceProvider {
    public function boot()
    {
        //
    }
    public function register()
    {
        $this->app->bind(
            'App\Services\Pagos\PagosContract',
            'App\Services\Pago\PagoPaypal'
        );
    }
}
```





# Ejemplo Mapeo comando al inicio

```
<?php namespace App\Providers;

use Illuminate\Bus\Dispatcher;
use Illuminate\Support\ServiceProvider;

class BusServiceProvider extends ServiceProvider {

    public function boot(Dispatcher $dispatcher)
    {
        $dispatcher->mapUsing(function($command)
        {
            return Dispatcher::simpleMapping(
                $command, 'App\Commands', 'App\Handlers\Commands'
            );
        });
    }

    public function register()
    {
        //
    }
}
```

# Contracts

# Contracts

- Los contracts definen las interfaces de los service providers.
- El contract será por tanto una interfaz pendiente de implementar por una clase que mediante el service providers aplica la resolución de la misma.
- Los contracts del sistema
  - <http://laravel.com/docs/5.0/contracts#contract-reference>
- Crear el nuestro propio
  - Veamos un ejemplo....



# Ejemplo UserContract I

```
<?php namespace App\Repositories\Frontend\User;

/**
 * Interface UserContract
 * @package App\Repositories\User
 */
interface UserContract {

    /**
     * @param $data
     * @return mixed
     */
    public function create($data);

    /**
     * @param $data
     * @return mixed
     */
    public function findByUserNameOrCreate($data, $provider);

    /**
     * @param $provider
     * @param $providerData
     * @param $user
     * @return mixed
     */
    public function checkIfUserNeedsUpdating($provider, $providerData, $user);
```

```
/**
 * Class EloquentUserRepository
 * @package App\Repositories\User
 */
class EloquentUserRepository implements UserContract {

    /**
     * @param $data
     * @param bool $provider
     * @return static
     */
    public function create($data, $provider = false) {
        $user = User::create([
            'name' => $data['name'],
            'email' => $data['email'],
            'password' => $provider ? null : $data['password'],
            'confirmation_code' => md5(uniqid(mt_rand(), true)),
            'confirmed' => config('access.users.confirm_email') ? 0 : 1,
        ]);

        return $user;
    }
}
```

# Ejemplo UserContract II

Enlace en el provider.

```
$this->app->bind(
    'App\Repositories\Backend\User\UserContract',
    'App\Repositories\Backend\User\EloquentUserRepository'
);
```

```
/**
 * Class UserController
 */
class UserController extends Controller {

    /**
     * @var UserContract
     */
    protected $users;

    /**
     * @param UserContract $users
     * @param RoleRepositoryContract $roles
     * @param PermissionRepositoryContract $permissions
     */
    public function __construct(
        UserContract $users) {
        $this->users = $users;
    }
}
```

# Facades



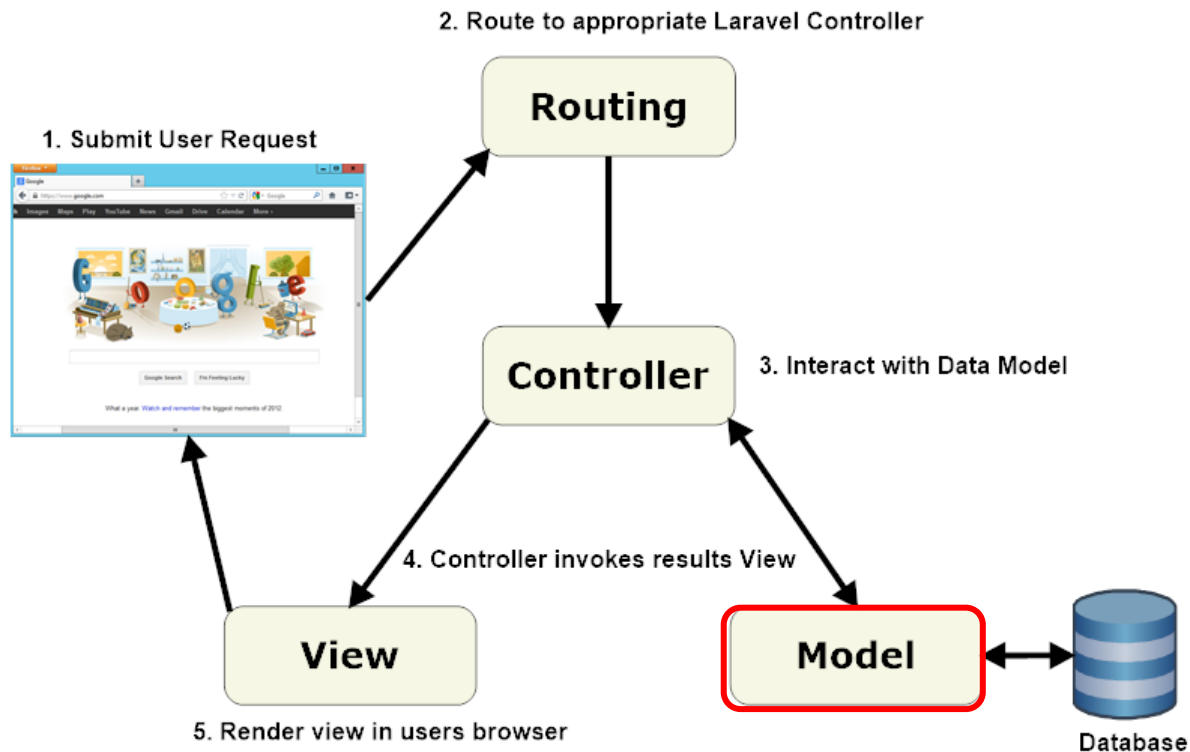
# Facades

- Patrón de diseño
- Facilita un enlace “static” a una clase sin crear una instancia a través del service container de la aplicación.
- Laravel incorpora algunas por defecto Input, Cache,... podemos verlas en config/app.php donde estan los “alias”
- Solo necesitamos implementar el método `getFacadeAccessor` para crear una facade. Se accede internamente con el metodo magico de PHP `__callStatic()`.
- Cuando hacemos `MiClass::metodo` laravel resuelve la operación con el service container.
- Veamos algún ejemplo del sistema...
- Quiero saber más... <http://laravel.com/docs/5.0/facades>

# Models

modelos

# MVC con Ruteo





# Modelos

- Normalmente el Modelo o Entidad suele tener el mismo nombre que el Controller.
- Extendemos de Model (Illuminate\Database\Eloquent\Model)
- Podemos hacer uso de los traits dentro de las clases.
- Un ejemplo de simplicidad es que si creamos el modelo User, automáticamente sabe que tiene que buscar la tabla “users” para traer o insertar contenido. Debemos si, extender las funcionalidades como veremos en el tema de ORM.
- Aquí pondremos las relaciones para que Eloquent ORM entienda como esta mapeada la BD.
- No preocuparos, en el tema de ORM veremos más en profundidad estos temas
- Veamos la estructura...

# Ejemplo de modelo básico

```
<?php namespace laravelsamples;

use Illuminate\Auth\Authenticatable;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Auth\Passwords\CanResetPassword;
use Illuminate\Contracts\Auth\Authenticatable as AuthenticatableContract;
use Illuminate\Contracts\Auth\CanResetPassword as CanResetPasswordContract;

class User extends Model implements AuthenticatableContract, CanResetPasswordContract {

    use Authenticatable, CanResetPassword;

    /**
     * The database table used by the model.
     *
     * @var string
     */
    protected $table = 'users';

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = ['name', 'email', 'password'];

    /**
     * The attributes excluded from the model's JSON form.
     *
     * @var array
     */
    protected $hidden = ['password', 'remember_token'];
}
```

# ¡Ahora os toca a vosotros!

## Ejercicios

1. Crear un middleware que compruebe si existe un token en la cabecera http con Postman para una ruta de tipo delete.
2. Crear un service provider para un repositorio con una interfaz y su implementación, por ejemplo una interfaz Payment y una implementación con tarjeta de credito (no hace falta hacer las operaciones, solo el esqueleto)
3. Crear un modelo Product y obtener de la base de datos 3 columnas, id + nombre + precio



# ¡MUCHAS GRACIAS A TODOS POR VUESTRA PARTICIPACIÓN!

Aquí me teneis para lo que os pueda ayudar.  
Estáis invitados a seguirme en la redes sociales  
como @micromante o Carlos Ruiz Ruso  
[www.micromante.com](http://www.micromante.com)