# Generar Entities y bundle

## Pasos para crear Bundles

Para crear bundles se usa el comando

## php bin/console generate:bundle --namespace=BackendBundle --format=yml

## Pasos para crear Entities

Para crear entities, con este comando se crea un fichero yml con las configuraciones en formato YML

Php bin/console doctrine:mapping:import BackendBundle yml

Generara un archivo yml por cada tabla de la DB. En la ruta bundle/condig/doctrine/

Para crear entities en solo una tabla se crea con el comando

Php bin/console doctrine:mapping:import BackendBundle yml –filter =”NombreTabla”

Para generar las entidades se lanza el comando

Php bin/console doctrine:generate:entities BackendBundle

## Pasos para probar un controlador y la carga de datps

En default controller utilizar lo siguiente:

/\*\*

\* @Route("/pruebas", name="pruebas")

\*/

public function pruebasAction(Request $request)

{

$em = $this->getDoctrine()->getManager();

$users=$em->getRepository('BackendBundle:User')->findAll();

var\_dump($users);

die();

}

# Instalar paquetes con Composer

## Installar JWT y el paginador

Agregar en composer.json:

"require": {

"php": ">=5.5.9",

"symfony/symfony": "3.0.\*",

"doctrine/orm": "^2.5",

"doctrine/doctrine-bundle": "^1.6",

"doctrine/doctrine-cache-bundle": "^1.2",

"symfony/swiftmailer-bundle": "^2.3",

"symfony/monolog-bundle": "^2.8",

"sensio/distribution-bundle": "^5.0",

"sensio/framework-extra-bundle": "^3.0.2",

"incenteev/composer-parameter-handler": "^2.0",

"firebase/php-jwt":"^3.0.0",

"knplabs/knp-paginator-bundle": "2.5.\*"

},

Ejecutar el comando:

composer update

# Rutas YML

## Cambiar las rutas para gestionarlas desde otro método con YML

Crear carpeta en AppBundle que se llame resources

Crear en esta carpeta Resources una carpeta llamada config

Crear archivo routing.yml

Crear un directorio llamado routing

Crear archivo llamado default.yml

-----------------------

Modificar en ROOT app/config/routing.yml

app:

resource: "@AppBundle/Resources/config/routing.yml"

prefix: /

-----------------------

en AppBundle Resource config routing.yml

app\_default:

resource: "@AppBundle/Resources/config/routing/default.yml"

prefix: /

-----------------------

En default.yml

default\_index:

path: /

defaults: {\_controller: "AppBundle:Default:index"}

methods: GET

pruebas\_index:

path: /pruebas

defaults: {\_controller: "AppBundle:Default:pruebas"}

methods: GET

# Funciones

## Crear una función para convertir un array en JSON

El controlador que devolverá los datos quedará de la forma:

public function pruebasAction(Request $request)

{

$em = $this->getDoctrine()->getManager();

$users=$em->getRepository('BackendBundle:User')->findAll();

$pruebas= array("id"=>1, "nombre"=>"andes");

return $this->json($users);

}

Esto se realizará ya que siempre se devolverá un JSON en cada método,. La Funcion que convertirá todo a un JSON para usar en un controller es:

public function json($data){

$normalizers = array( new \Symfony\Component\Serializer\Normalizer\GetSetMethodNormalizer());

$encoders= array ("json"=>new \Symfony\Component\Serializer\Encoder\JsonEncode());

$serializer= new \Symfony\Component\Serializer\Serializer($normalizers, $encoders);

$json = $serializer->serialize($data, 'json');

$response = new \Symfony\Component\HttpFoundation\Response();

$response->setContent($json);

$response->headers->set("Content-Type", "application/json");

return $response;

}

# SERVICIOS

## Crear un servicio Helpers para convertir un objeto o un array a JSON.

Crear un directorio en el bundle que se llame: services

Crear en esta carpeta un archivo clase php que se llame: helpers.php

Declarar el archivo services:

<?php

namespace AppBundle\Services;

class Helpers {

}

Copiar la función del controlador al servicio.

Ir a app/config/services.yml y editar el archivo:

parameters:

# parameter\_name: value

**services:**

# service\_name:

# class: AppBundle\Directory\ClassName

# arguments: ["@another\_service\_name", "plain\_value", "%parameter\_name%"]

**app.helpers:**

**class: AppBundle\Services\Helpers**

**arguments: ["null"]**

Modificamos el controlador para invocar el servicio

public function pruebasAction(Request $request)

{

**$helpers = $this->get("app.helpers");**

$em = $this->getDoctrine()->getManager();

$data=$em->getRepository('BackendBundle:User')->findAll();

**return $helpers->json($data);**

}

# Crear LOGIN para servicio API

## Primeros pasos para Login

### Ruta

Crear una ruta en el directorio default.yml con el acceso al login

default\_login:

path: /login

defaults: {\_controller: "AppBundle:Default:login"}

methods: POST

### Controlador

En el encabezado adjuntar la siguiente linea

use Symfony\Component\Validator\Constraints as Assert;

ge

Crear el contrenido en el controlador en donde se reciva por POST los datos de login en JSON, a continuacion se valida si se recibe un JSON con data y a continuacion se decodifica. la validación de mail se hace con email Constraint, de la siguiente manera,

public function loginAction(Request $request){

$helpers = $this->get("app.helpers");

$json= $request->get("json", null);

if ($json!=null){

$params=json\_decode($json);

$email= (isset($params->email))? $params->email : null;

$password=(isset($params->password))? $params->password : null;

$emailCons= new Assert\Email();

$emailCons->message="Formato de email no valido";

$validate\_email= $this->get("validator")->validate($email,$emailCons);

if(count($validate\_email)==0 && $password!=null){

echo "Data Success!!!";

} else {

echo "Data Incorrect!!!!!";

}

} else {

echo "Not received, Please send JSON data to login";

die();

}

}

Se debe probar el servicio con POSTMAN enviando un JSON con los datos Creados asi:

Key :json value: {"email": "andes@gmail.com", "password": "1234"}

## Crear Token para Login

Para realizar el Login de la API se implementara JWT para crear los tokens de los usuarios registrados y pasar este token entre los sitios a los cuales se tendrá acceso.

Crear Nuevo servicio JWT para validar Usuario y tambien para codificar y decodificar TOKENS

### Services.yml

Se creara un servicio llamado jwt\_auth en el directorio de los servicios de app/config/services.yml , este servicio tendra como argumentos el manejo del ORM doctrine y su entity manager,

app.jwt\_auth:

class: AppBundle\Services\JwtAuth

arguments: ["@doctrine.orm.entity\_manager"]

### JwtAuth.php

En la carpeta de servicios del AppBundle se creara una clase php con nombre jwtAth.php con el siguiente codigo:

<?php

namespace AppBundle\Services;

use Firebase\JWT\JWT;

class JwtAuth {

public $manager;

public function \_\_construct ($manager){

$this->manager=$manager;

}

public function signup($email, $password, $getHash=NULL){

$key="clave-secreta";

$user =$this->manager->getRepository('BackendBundle:User')->findOneBy(

array (

"email"=>$email,

"password"=>$password

)

);

$signup=false;

if(is\_object($user)){

$signup=true;

}

if($signup==true){

// return array("status"=>"success", "data"=>"Correct Login!!!");

//Datos para crear el token

$token= array(

"sub"=>$user->getId(),

"email"=>$user->getEmail(),

"name"=>$user->getName(),

"surname"=>$user->getSurname(),

"password"=>$user->getPassword(),

"image"=>$user->getImage(),

"iat"=>time(),

"exp"=>time()+( 7 \* 24 \* 60 \*60 )

);

$jwt= JWT::encode($token,$key, 'HS256');

$decoded=JWT::decode($jwt,$key, array("HS256"));

if($getHash!=null){

return $jwt;

}else {

return $decoded;

}

}

else{

return array("status"=>"error", "data"=>"Login Failed, User and Password incorrect!!!");

die();

}

}

}

### Controlador

En los encabezados del controlador se utilizara el componente JSON response:

use Symfony\Component\HttpFoundation\JsonResponse as JsonResponse;

Se creara un metodo denominado LoginAction con el siguiente codigo:

**public function loginAction(**Request $request){

$helpers = $this->get("app.helpers");

$jwt\_auth = $this->get("app.jwt\_auth");

$json= $request->get("json", null);

if ($json!=null){

$params=json\_decode($json);

$email= (isset($params->email))? $params->email : null;

$password=(isset($params->password))? $params->password : null;

$getHash=(isset($params->gethash))? $params->gethash : null;

$emailCons= new Assert\Email();

$emailCons->message="Formato de email no valido";

$validate\_email= $this->get("validator")->validate($email,$emailCons);

if(count($validate\_email)==0 && $password!=null){

if($getHash== null){

$signup=$jwt\_auth->signup($email,$password);

}

else{

$signup=$jwt\_auth->signup($email,$password, true);

}

return new JsonResponse($signup);

} else {

return $helpers->json(array(

"status" =>"error",

"data"=>"API login Not Valid"

));

}

}

### Pruebas

Se realizaran las pruebas con la herramienta postman pasando argumentos por el metodo POST de la siguiente manera:

<http://localhost/SymAPI/web/app_dev.php/login>

los datos pasados como llave, valor para que devuelva un Token son:

key: json

value: {"email":"admin@gmail.com","password":"admin","gethash":"true"}

Los datos pasados para que devuelva un JSON con los datos del Usuario son:

key: json

value: {"email":"admin@gmail.com","password":"admin”}