iPSSI

ÉCOLE D'INFORMATIQUE WEB ET DESIGN GRAPHIQUE

Symfony 4

The best PHP framework

whoami

Christophe VILLEGER

Develop'hacker @ Darkmira

Zend Certified PHP Engineer

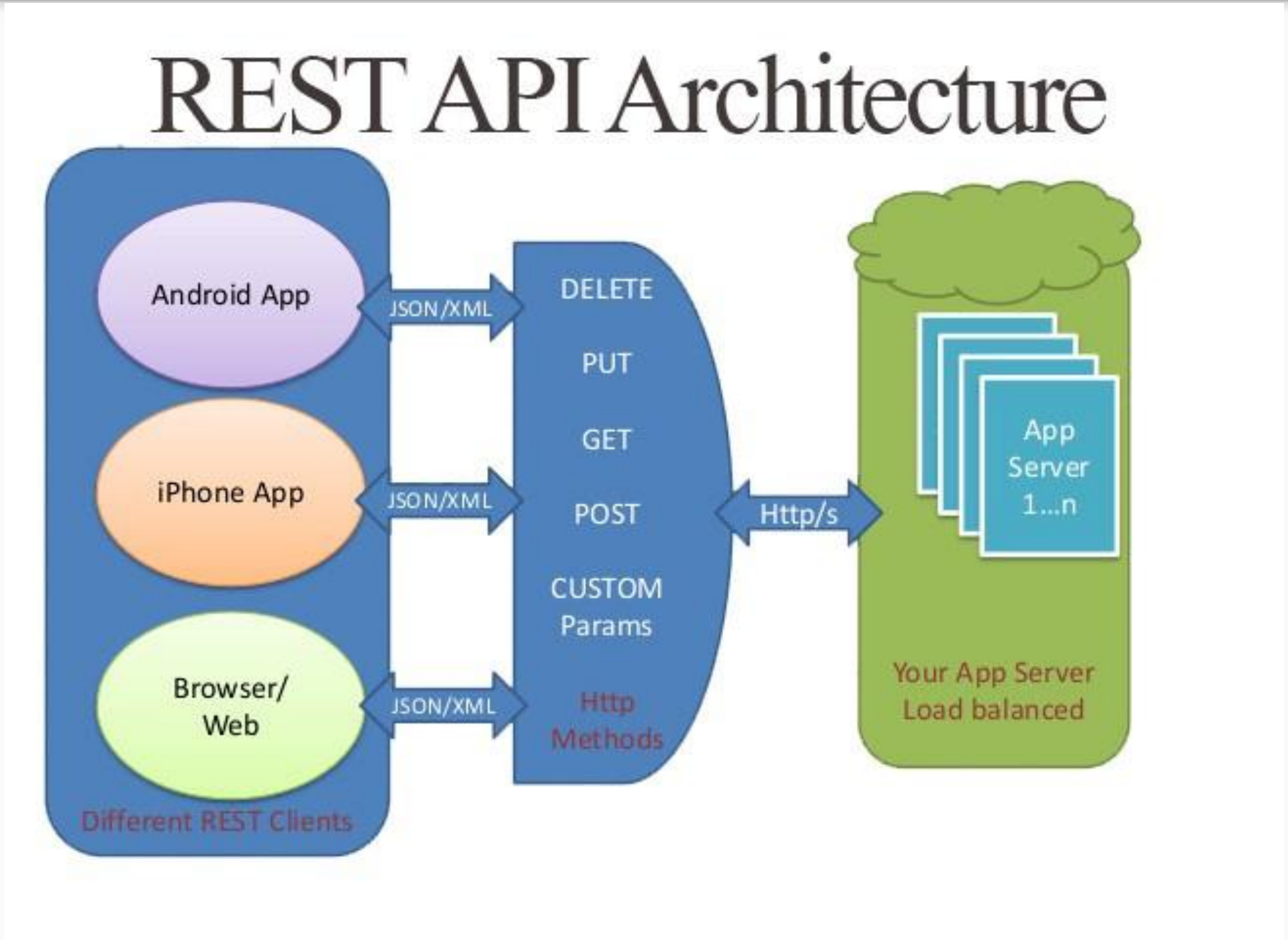
Symfony Code Contributor (v3.4.8; v3.4.9; v4.0.8; v4.0.9)

Loves clean architecture, continuous integration, performance and build robots with Raspberry Pi

API REST (Web Service)

- **Web API** : Interface consisting of one or more publicly **exposed endpoints (URI)** to a defined **request–response** message system, typically expressed in **JSON**
- **Documentation** : To provide a web API of high quality, there needs to be a good level of documentation
- **RE**presentational **S**tate **T**ransfer : Using a **uniform** and predefined set of **stateless** operations. Provide the ability to grow, by **re-using components** that can be managed and updated, even while it is running. The operations available are **GET, POST, PUT, DELETE**, and other predefined **CRUD HTTP** methods.

API REST (Web Service)



API REST Object Oriented Resources

Endpoints based on resources

- List Users : [GET] /users
- Create User : [POST] /users
- Update User : [PUT] /users/{user_id}
- Delete User : [DELETE] /users/{user_id}
- List Comments from User : [GET] /users/{user_id}/comments
- Delete Comments from User : [DELETE] /users/{user_id}/comments/{comment_id}

API REST Client

REST Client doing GET, POST, PATCH, DELETE requests

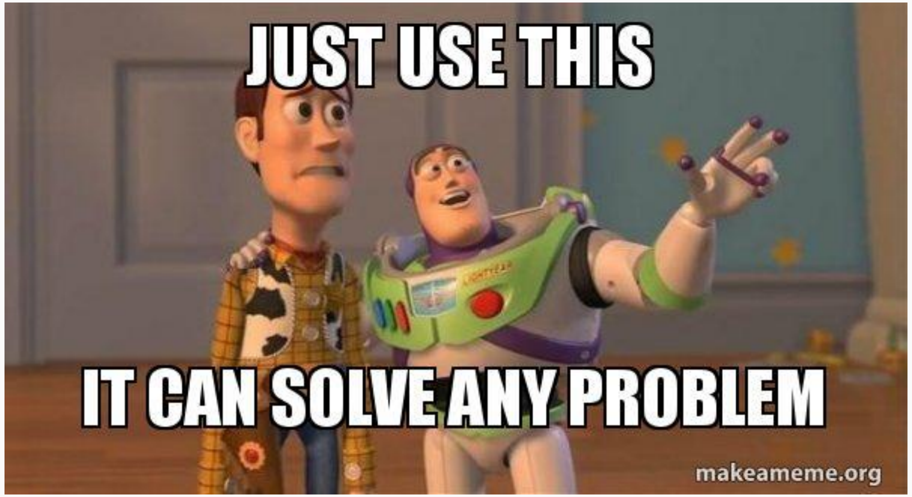
- Postman (Most Used) : https://www.getpostman.com/
- Insomnia : https://insomnia.rest
- PhpStorm self client
- Many more...

Symfony : Build a REST API

FOSRestBundle

Provides various **tools** to rapidly develop **RESTful API's** & applications with **Symfony** :

- A **View layer** to delegate the logic of data output
- A **custom route** loader to generate URIs following REST conventions
- **RESTful** decoding of HTTP **request body** and Accept headers
- **Exception** controller for sending appropriate **HTTP status codes**



TP Install FOSRestBundle with Symfony Flex

composer req

- orm security monolog translator serializer validator friendsofsymfony/rest-bundle sensio/framework-extra-bundle

composer --dev req

- profiler maker

Symfony API REST : Guard Authentication

Guard Auth is one of best ways to do an **API Authentication**

- If you need to build a **traditional login form**, an **API token authentication** system or you need to integrate with some proprietary **single-sign-on system**, the **Guard component** can make it easy and fun!
- You **always** need to create a **User class that implements UserInterface** and **configure a user provider**. Users will be stored in the database via Doctrine, and each user has an **apiKey** property ***they use to access their account via the API***

TP Symfony API **User Auth**

Using Symfony Maker

- php bin/console make:entity User

firstname, lastname, email (unique=true), birthday, roles (type=”simple_array”), apiKey (unique=true);

```
/**
* @UniqueEntity("email")
* @ORM\Entity(repositoryClass="App\Repository\UserRepository")
*/
class User implements UserInterface
```

TP Symfony API **User Auth**

Configure your User Provider

```
# config/packages/security.yaml
security:

   providers:

       your_db_provider:

           entity:

               class: App\Entity\User

               property: apiKey
```

Symfony API How Authenticate User

We've just created an attribute **apiToken** : A property that users will use to **access their account**.

Your clients will send an **API-TOKEN header** on each request with their API token. Your job is to read this and **find the associated user** (if any).

- To create a custom authentication system, just create a **TokenAuthenticator** class and make it extend the **AbstractGuardAuthenticator**. This requires you to implement several methods.

Symfony API TokenAuthenticator

```php
namespace App\Security;

use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Security\Core\User\UserInterface;
use Symfony\Component\Security\Guard\AbstractGuardAuthenticator;
use Symfony\Component\Security\Core\Authentication\Token\TokenInterface;
use Symfony\Component\Security\Core\Exception\AuthenticationException;
use Symfony\Component\Security\Core\User\UserProviderInterface;

class TokenAuthenticator extends AbstractGuardAuthenticator
```

Symfony API TokenAuthenticator

```php
/**
* Called on every request to decide if this authenticator should be
* used for the request.
*/
public function supports(Request $request)
{
   return $request->headers->has('AUTH-TOKEN');
}

/**
* Return whatever credentials you want to be passed to getUser() as $credentials.
*/
public function getCredentials(Request $request)
{
   return array(
       'token' => $request->headers->get('AUTH-TOKEN'),
   );
}

public function getUser($credentials, UserProviderInterface $userProvider)
{
   $apiKey = $credentials['token'];
   if (null === $apiKey) {
       return;
   }
   // if a User object, checkCredentials() is called
   return $userProvider->loadUserByUsername($apiKey);
}
```

Symfony API TokenAuthenticator

```php
public function checkCredentials($credentials, UserInterface $user)
{
    // check credentials - e.g. make sure the password is valid
    // no credential check is needed in this case

    // return true to cause authentication success
    return true;
}

public function onAuthenticationSuccess(Request $request, TokenInterface $token, $providerKey)
{
    // on success, let the request continue
    return null;
}

public function onAuthenticationFailure(Request $request, AuthenticationException $exception)
{
    $data = array(
        'message' => strtr($exception->getMessageKey(), $exception->getMessageData())
    );

    return new JsonResponse($data, Response::HTTP_FORBIDDEN);
}
```

Symfony API TokenAuthenticator

```
/**
 * Called when authentication is needed, but it's not sent
 */
public function start(Request $request, AuthenticationException $authException = null)
{
    $data = array(
        'message' => 'Authentication Required'
    );

    return new JsonResponse($data, Response::HTTP_UNAUTHORIZED);
}

public function supportsRememberMe()
{
    return false;
}
```

Symfony API TokenAuthenticator

```yaml
security:
   providers:
       your_db_provider:
           entity:
               class: App\Entity\User
               property: apiKey
   firewalls:
       dev:
           pattern: ^/(_(profiler|wdt)|css|images|js)/
           security: false
       main:
           anonymous: ~
           logout: ~
           stateless: true
           guard:
               authenticators:
                   - App\Security\TokenAuthenticator
```

TP Symfony API Create our first routes !

Users Resources ! Create a class **UsersController**

```
namespace App\Controller;
use FOS\RestBundle\Controller\FOSRestController;
class UsersController  extends FOSRestController
{
   public function getUsersAction()
   {} // "get_users"            [GET] /users
   public function getUserAction($id)
   {} // "get_user"             [GET] /users/{id}
   public function postUsersAction()
   {} // "post_users"           [POST] /users
   public function putUserAction($id)
   {} // "put_user"             [PUT] /users/{id}
   public function deleteUserAction($id)
   {} // "delete_user"          [DELETE] /users/{id}
}
```

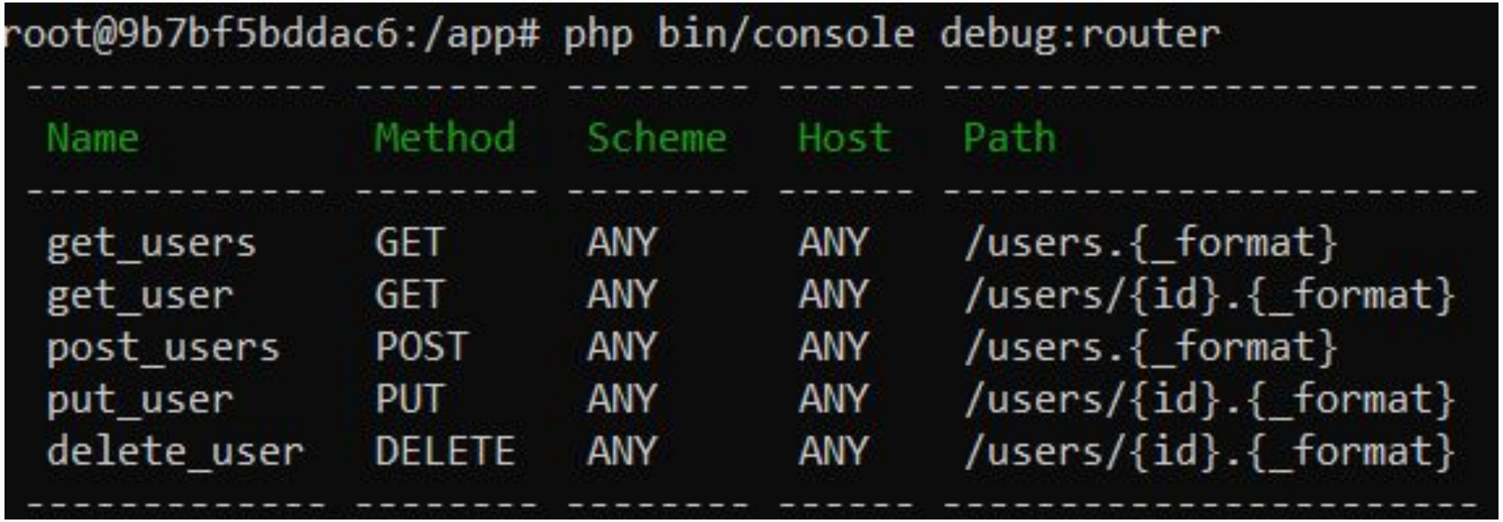
TP Symfony API : Make your Controller “REST”

```yaml
# config\routes.yaml
users:
    type: rest
    resource: App\Controller\UsersController
```

```yaml
# config\packages\fos_rest.yaml
fos_rest:
    view:
        view_response_listener: true
    routing_loader:
        default_format: json
```

TP Symfony API : Debug your Router !

List all your API routes



```
root@9b7bf5bddac6:/app# php bin/console debug:router
 ------------- -------- -------- ------ -------------------------
  Name          Method   Scheme   Host   Path
 ------------- -------- -------- ------ -------------------------
  get_users     GET      ANY      ANY    /users.{_format}
  get_user      GET      ANY      ANY    /users/{id}.{_format}
  post_users    POST     ANY      ANY    /users.{_format}
  put_user      PUT      ANY      ANY    /users/{id}.{_format}
  delete_user   DELETE   ANY      ANY    /users/{id}.{_format}
 ------------- -------- -------- ------ -------------------------
```

TP Symfony API Let’s list all users

```
class UsersController extends FOSRestController
{
   private $userRepository;
   public function __construct(UserRepository $userRepository)
   {
       $this->userRepository = $userRepository;
   }
   public function getUsersAction()
   {
       $users = $this->userRepository->findAll();
       return $this->view($users);
   }
```

Create a first user using phpMyAdmin and check /users with Postman

TP Symfony API List all users

GET /users

```
[
        {
        "id": 1,
        "firstname": "Christophe",
        "lastname": "Villeger",
        "email": "cvilleger@fakeapple.com",
        "birthday": "1990-09-18T00:00:00+02:00",
        "roles": [
        "ROLE_USER"
        ],
        "apiKey": "vvfc1j3h6d4ef64",
        "password": null,
        "salt": null,
        "username": "cvilleger@fakeapple.com"
        }
]
```

Check the response headers

Cache-Control →no-cache, private
Connection →keep-alive
Content-Type →application/json
Date →Sat, 11 Aug 2018 19:25:55 GMT
Server →nginx/1.10.3
Transfer-Encoding →chunked
X-Debug-Token →932e9a
X-Debug-Token-Link →http://localhost/_profiler/932e9a

Symfony API List One User

Fetch Automatically

```php
public function getUserAction(User $user)
{
    return $this->view($user);
}
```

```
GET /users/1

{
        "id": 1,
        "firstname": "Christophe",
        "lastname": "Villeger",
        "email": "cvilleger@fakeapple.com",
        "birthday": "1990-09-18T00:00:00+02:00",
        "roles": [
        "ROLE_USER"
        ],
        "apiKey": "vvfc1j3h6d4ef64",
        "password": null,
        "salt": null,
        "username": "cvilleger@fakeapple.com"
}
```

Symfony API Post User

Request Body Converter Listener

- The Request body converter makes it possible to **deserialize** the **request body** into an **object**

```yaml
fos_rest:
   view:
       view_response_listener: force
   routing_loader:
       default_format: json
   body_converter:
       enabled: true
   param_fetcher_listener:  true
```

```yaml
sensio_framework_extra:
   request: { converters: true }
   router:
       annotations: false
```

Symfony API Post User

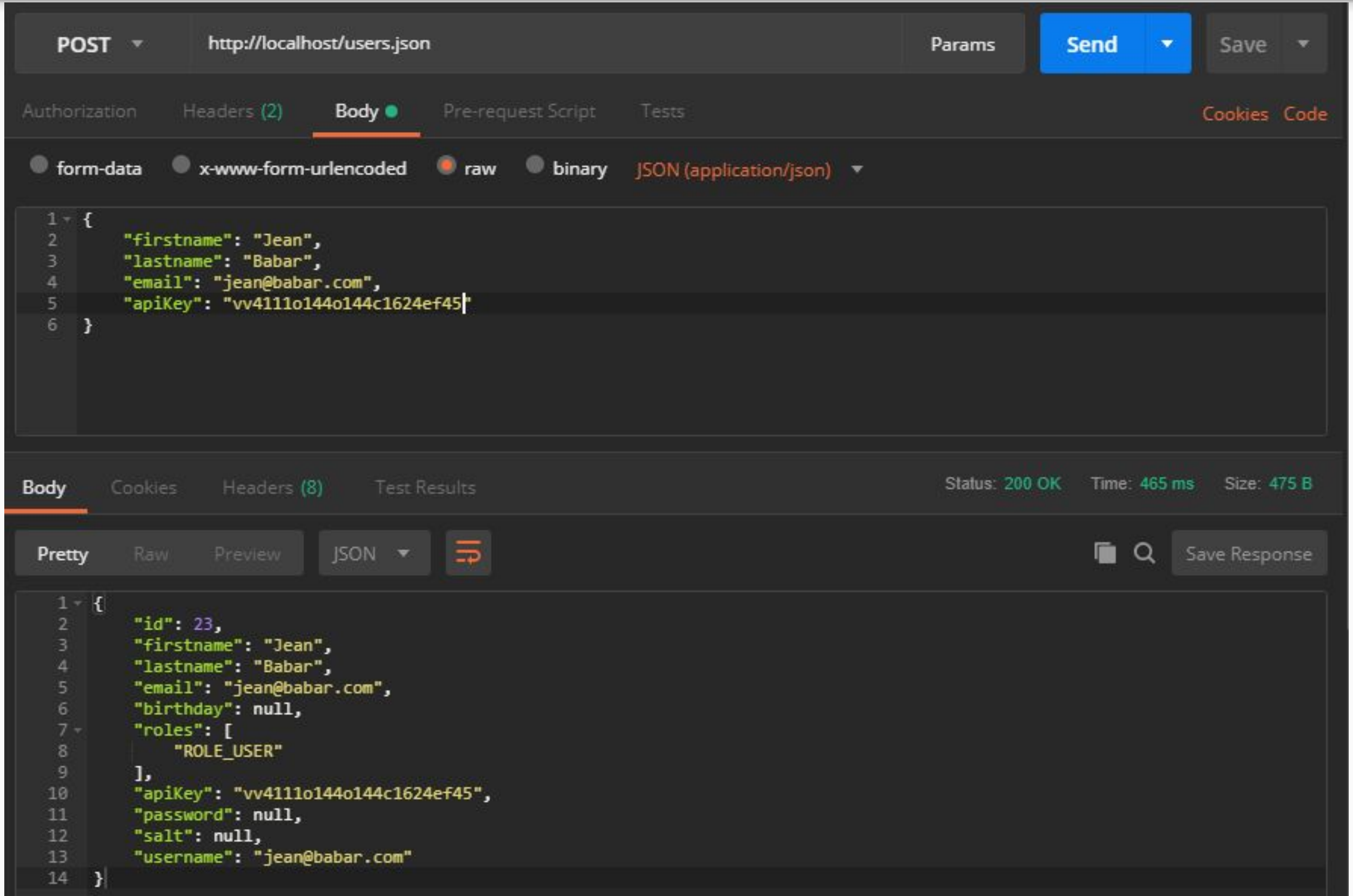
```php
use App\Entity\User;
use App\Repository\UserRepository;
use Doctrine\ORM\EntityManagerInterface;
use FOS\RestBundle\Controller\Annotationsas Rest;
use FOS\RestBundle\Controller\FOSRestController;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\ParamConverter;

class UsersController extends FOSRestController
{
   private $userRepository;
   private $em;

   public function __construct(UserRepository $userRepository, EntityManagerInterface $em)
   {
       $this->userRepository = $userRepository;
       $this->em = $em;
   }

/**
* @Rest\Post("/users")
* @ParamConverter("user", converter="fos_rest.request_body")
*/
public function postUsersAction(User $user)
{
   $this->em->persist($user);
   $this->em->flush();
   return $this->view($user);
}
```

Symfony API Postman



http://localhost/users.json

```
{
    "firstname": "Jean",
    "lastname": "Babar",
    "email": "jean@babar.com",
    "apiKey": "vv4111o144o144c1624ef45"
}
```

Status: 200 OK Time: 465 ms Size: 475 B

```
{
    "id": 23,
    "firstname": "Jean",
    "lastname": "Babar",
    "email": "jean@babar.com",
    "birthday": null,
    "roles": [
        "ROLE_USER"
    ],
    "apiKey": "vv4111o144o144c1624ef45",
    "password": null,
    "salt": null,
    "username": "jean@babar.com"
}
```

Symfony API Edit User

The **Request** contains all **attributes** that user want to **modify** :

```
public function putUserAction(Request $request, int $id)
{
    // $request->get('firstname')
}
```

Symfony API : Group Serializer

Define which attributes you want to serialize

```
/**
* @Rest\View(serializerGroups={"user"})
*/
public function getUsersAction()
{
   $users = $this->userRepository->findAll();
   return $this->view($users);
   // "get_users"
}
```

```
/**
* @Groups("user")
* @ORM\Id()
* @ORM\GeneratedValue()
* @ORM\Column(type="integer")
*/
private $id;
```

TP Symfony API : User CRUD & Article CRUD

- **User** CRUD and **Article** CRUD (**Article** have name, description, createdAt, *optional* User)
- **User** can edit his informations (firstname, lastname, email, apiKey)
- **User** can **CRUD** his **Article**
- **Admin** can **CRUD** all Users and all Articles