Symfony

Pierre Bretéché

11/04/2014

DAWAN, 28, rue de Strasbourg, 44000 NANTES
formation@dawan.fr

Plan de l'intervention

Initiation

- Présentation
- Configuration serveur web
- Projet
- Le contrôleur
- La vue
- Le modèle
- Formulaire

Plan de l'intervention

Approfondissement

- Sécurité
- Internationalisation
- Services
- Qualité & performance

Présentation réalisée principalement à partir de la documentation officielle :
http://symfony.com/doc/current/index.html

Master version : 2.4.2

Présentation

Frameworks

- CakePHP
- Code Igniter
- Ez Component
- Zend Framework
- Symfony

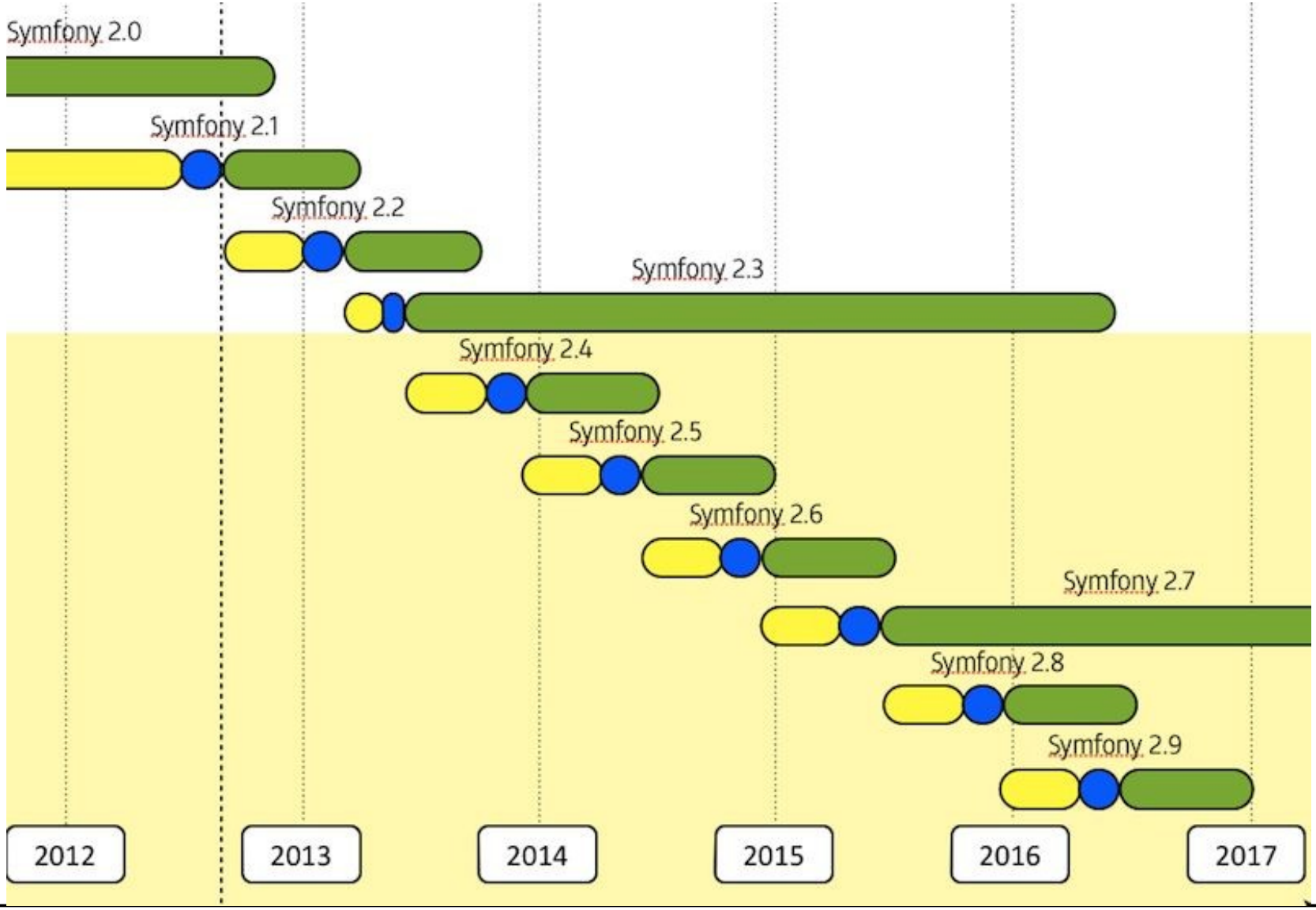
Un cadriciel PHP

- Cadriciel :
  - Impose une architecture
  - Offre des bibliothèques de fonctions
  - Accélère le développement selon un modèle
- Philosophie :
  - Licence Open-Source, extensibilité, standardisation, interopérabilité
- Communauté :
  - Sensio = développeurs, utilisateurs, contributeurs

Historique

- début du projet: 18 oct 05
- V 1.0 : jan 07
- V 1.3 & 1.4 : nov 09 (1.4 LTS 3 ans)
- V 2.0 : juil 11 (refonte du cœur)
- V 2.1 : sep 12 (nouveaux composants)
- V 2.2 : mar 13
- V 2.3 : juin 13 (LTS 3 ans + compatibilité)
- V 2.4 : nov 13

Versions

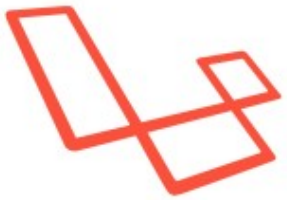

Symfony 2.0
Symfony 2.1
Symfony 2.2
Symfony 2.3
Symfony 2.4
Symfony 2.5
Symfony 2.6
Symfony 2.7
Symfony 2.8
Symfony 2.9
2012
2013
2014
2015
2016
2017

Versions

- Rétro compatibilité → 2.3
- Le travail sur Symfony 3.0 commencera dès qu'il y aura suffisamment de fonctionnalités majeures non rétrocompatibles en attente sur la todo liste.

Projets

- Drupal, phpBB, Laravel, eZ Publish, Composer, Magento, Piwik, Silex, OroCRM, Doctrine ...



Différence Symfony 1 & 2

- Arborescence revisitée :
  - /apps vs /app :
    - Une seule application dans symfony2
    - App ne contient plus que des fichiers de conf
  - /src :
    - Bundles vs plugins
    - Toute l'application est découpée en bundles
  - /lib/vendor vs /vendor
  - /web
    - css, js, images => bundles

Différence Symfony 1 & 2

- Autoload :
  - + nom de classe & namespace / chemin
  - - tableau de référence mis en cache
- Console :
  - symfony => app/console
- Applications :
  - 1 seule app : sub-namespaces, sous-répertoires, différentes configurations...

Différence Symfony 1 & 2

- Bundles vs plugins:
  - Plugins :
    - Configuration, modules, php libs, assets...
  - Bundles :
    -
- Routing et configuration:
  - Inclusion manuelle

Environnement de travail

- IDE : Eclipse/PDT | Netbeans | PHPStorm | ...
- Environnement AMP + navigateur
- composer

Installation

- Via composer

```
$ composer create-project
symfony/framework-standard-edition
/path/to/webroot/Symfony
```

- Ou http://symfony.com/download?v=Symfony_Standard_Vendors_2.4.2.tgz

vHost

```
<VirtualHost *:80>
   ServerName formation.local
   ServerAdmin webmaster@localhost

   DocumentRoot /home/pierre/Documents/www

   <Directory /home/pierre/Documents/www/>
      Options Indexes FollowSymLinks MultiViews
      AllowOverride All
      Require all granted
   </Directory>

   ErrorLog ${APACHE_LOG_DIR}/error.log
   CustomLog ${APACHE_LOG_DIR}/access.log
combined
</VirtualHost>
```

Droits

```
$ rm -rf app/cache/*
$ rm -rf app/logs/*

$ sudo chmod +a "www-data allow delete,write,append,
file_inherit,directory_inherit" app/cache app/logs
$ sudo chmod +a "`whoami` allow delete,write,append,
file_inherit,directory_inherit" app/cache app/logs
```

```
$ sudo setfacl -R -m u:www-data:rwX -m u:`whoami`:rwX
app/cache app/logs
$ sudo setfacl -dR -m u:www-data:rwx -m u:`whoami`:rwx
app/cache app/logs
```

Git

- .gitignore

```
/web/bundles/
/app/bootstrap*
/app/cache/*
/app/logs/*
/vendor/
/app/config/parameters.yml
```

```
$ git init
$ git add .
$ git commit -m "Initial commit"
```

Web Debug Toolbar

- Fichier de conf : web_profiler : toolbar
- Panneaux :
  - Config : environnement, bundles activés...
  - Request : infos HTTP
  - Exception : erreurs
  - Events : écouteurs
  - Logs : principales actions
  - Security : info utilisateur
  - Emails : mails envoyés
  - Doctrine : requêtes SQL, temps d'exécution

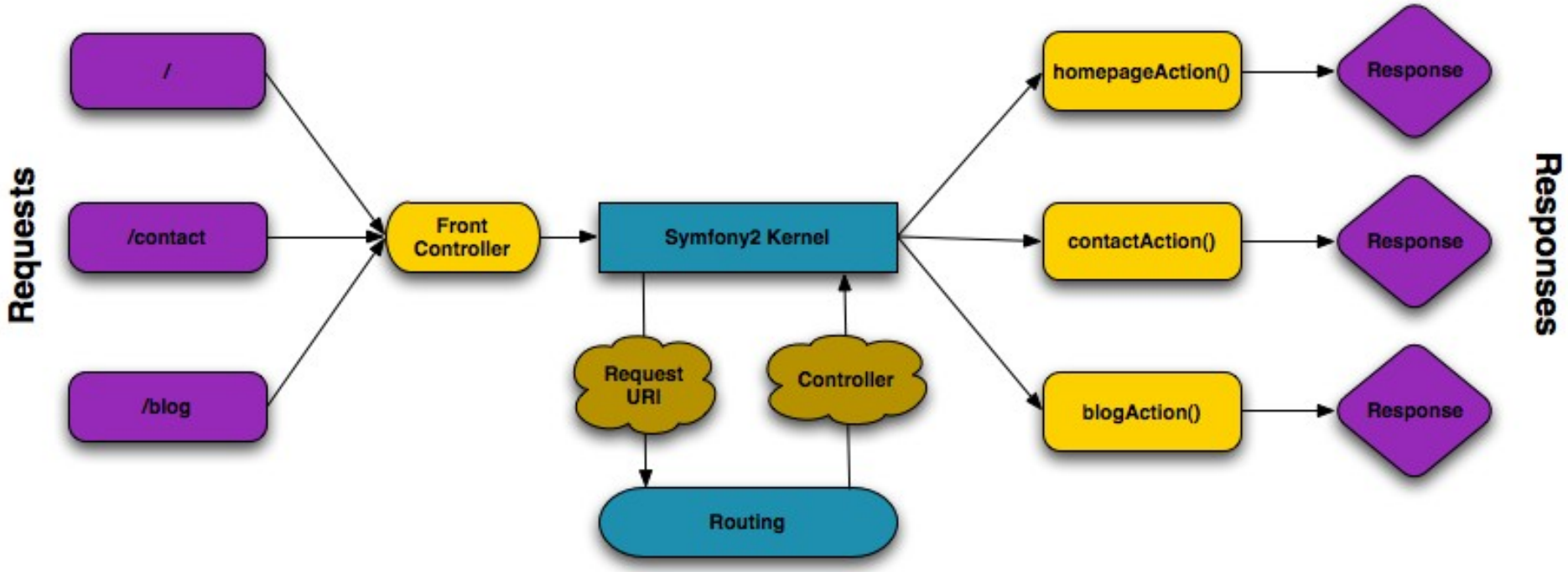
Architecture

Requêtes HTTP

- Composant **HTTPFoundation**
- Classes : Request & Response
- Request :
  - Représentation objet de la requête HTTP
  - Post/Get en tant que ParameterBag
- Response :
  - setContent(), setStatusCode, headers->set()...

URLs

- FrontController & mod_rewrite



MVC

- Contrôleurs :
  - FrontController : app.php | app_dev.php
    - Sécurité, configuration, routing
  - Contrôleurs : supervise la production de la réponse en cohérence avec la requête
- Vue :
  - Met en forme / formate la réponse
- Modèle :
  - Processus et données métiers

Répertoires

- app/: configuration de l'application
- src/: code PHP du projet
- vendor/: bibliothèques tierces (additionnelles)
- web/: racine web, fichiers publiquement accessibles

Répertoire Web

- Contrôleur frontal : app.php
- .htaccess : mod_rewrite
- Ressources publiques

Répertoire App

- AppKernel :
  - regiterBundles()
  - registerContainerConfiguration()
- fichiers conf et routage
- cache + logs
- ressources
- autoload.php
- console

Répertoire src

- Contient tout le code applicatif sous forme de Bundles

Bundles : généralités

- « Similar to a plugin, but even better »
- Tout est bundles dans SF2
- Déf : Ensemble structuré de fichiers liés à une fonctionnalité
- `$ php app/console generate:bundle --namespace=Dawan/HelloBundle --–format=yml`

Bundles : arborescence

- Controller : contrôleurs du bundle (ex HelloController.php);
- Dependency Injection : certaines classes d'extension d'injection de dépendances, voir plus loin(facultatif)
- Resources/config : configuration, notamment la configuration de routage (ex routing.yml)
- Resources/views : contient les templates organisés par nom de contrôleur (ex Hello/index.html.twig);
- Resources/public : contient les ressources web (images, feuilles de style, etc) et sont copiées ou liées par un lien symbolique dans le répertoire de projet web/ grâce à la commande assets:install;
- Tests : contient tous les tests du bundle

Environnement

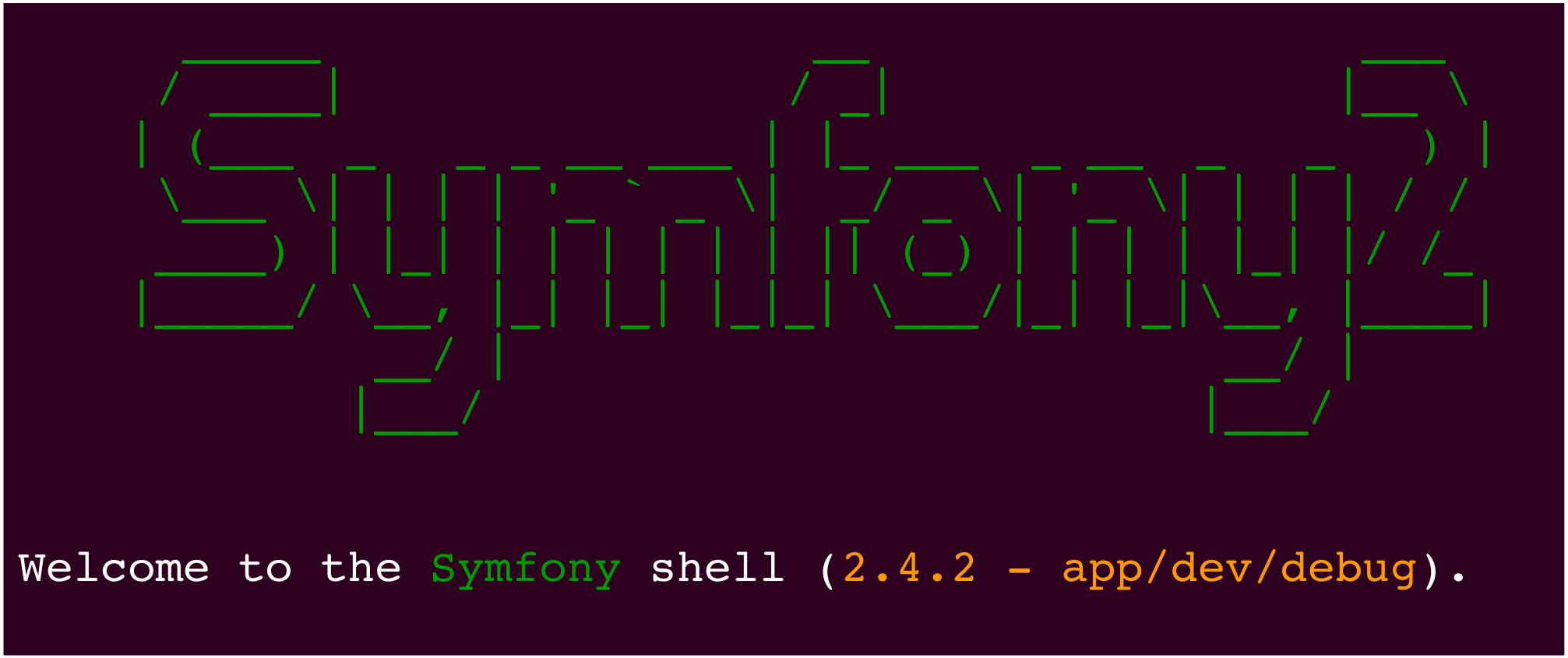
- Dev, prod, test
- Partage tout sauf conf et FrontController

Résumé

- page = route, contrôleur[, template]
- projet : répertoires principaux :
    - web/ (ressources web et contrôleurs frontaux)
    - app/ (configuration)
    - src/ (vos bundles),
    - vendor/ (librairies tierces)
- fonctionnalité de Symfony2 est organisée dans un bundle
- conf de bundle : Ressources/config (YAML, XML, PHP)
- conf globale de l'application app/config
- environnement / contrôleurs frontaux + conf

Console

```
$ php app/console -s
```



Console

- Console / shell Sf2
- Génération automatique de fichiers / code
- Autocomplétion : commandes / args
- Mode interactif

```
> generate:bundle
```

Contrôleur

Format YAML

- YAML Ain't Markup Language
- ~ JSON
  - + lisibilité humaine
  - - facilité de traitement / génération

YAML : collections

- Séquence :

```
- Symfony
- Drupal
- eZ Publish
```

- Mapping

```
name: Symfony initiation
duration: 5
center: Dawan #of course
```

Routage

- pattern + référence au controller
- Paramètre de substitution /hello/{name}
  - Valeurs par défaut :

```
defaults: {_controller: Bdle:Ctrl:act, param: 1}
```

  - Conditions requises :

```
requirements:
 • year: \d+
   _locale: fr|en
```

- Paramètres spéciaux :
  `_controller(obligatoire), _format, _locale`

Routage : paramètres

- _controller :
```
NomDuBundle:Ctrl:act
Vendor\NomDuBundle\Controller\CtrlController::ac
tAction
NomService:actAction
```
- Fusion des paramètres dans tableau associatif, passage en argument au contrôleur
- Ressources externes de routage :
```
dawan_hello:
resource: "@HelloBundle/Resources/config/routing.yml"
prefix: /admin
```
@HelloBundle => répertoire du bundle

Routage : objet Router

- $router->match('/hello/Pierre'):
  array('nom'=>'Pierre', '_controller'=>'Bdle:Ctrl:act')
- $router->generate('route_key',
  array('slug'=>'Pierre')):
  '/hello/Pierre'
- $router->generate('route_key',
  array('slug'=>'Pierre'), true):
  'http://localhost/hello/Pierre'
- $router->generate('route_key',
  array('slug'=>'Pierre', 'year'=>'2012')):
  '/hello/Pierre?year=2012'

Contrôleur

- Cycle de vie : Requête → contrôleur → réponse
- (Front) Router (appelle)→ Controller (génère)→ Response
- Router :
    - pattern : /hello/{param}
    - defaults: {_controller: NomBundle:Ctrl:method}
- class FrameworkBundle\Controller => helper

Contrôleur : helper

- Redirection :
```
redirect($this->generateUrl('homepage'));
```
- Sous-requête:
```
forward('VendorNameBundle:Ctrl:action', $params);
```
- Rendre un template :
```
$content=$this->renderView('template_file', $params);
return new Response($content);
```
- Ou directement :
```
return $this->render('template_file', $params);
```
- Autres services:
```
$router = $this->get('router');
```

Contrôleur : suite

- Page 404 :
```
throw $this->createNotFoundException('Message');
```
- Session :
```
$s=$this->getRequest()->getSession();
$s->set('foo','bar') ; $s->get('foo','default');
```
- Message Flash :
```
$session->getFlashBag()->add('notice', $msg)
```
- Persistance 1 requête, utile lors des redirections
```
$this->get('session')->getFlashBag()
->add('notice', 'Vos changements ont été
sauvegardés!');
```

Response & Request

- En-tête :
```
$response->headers->set('Content-
Type','application/xhtml+xml');
```
- Infos requête:
```
$req->isXmlHttpRequest();
$req->getPreferredLanguage(array('en','fr'));
```
- get/post :
```
$req->query->get('getparam');
$req->request->get('postdata');
```

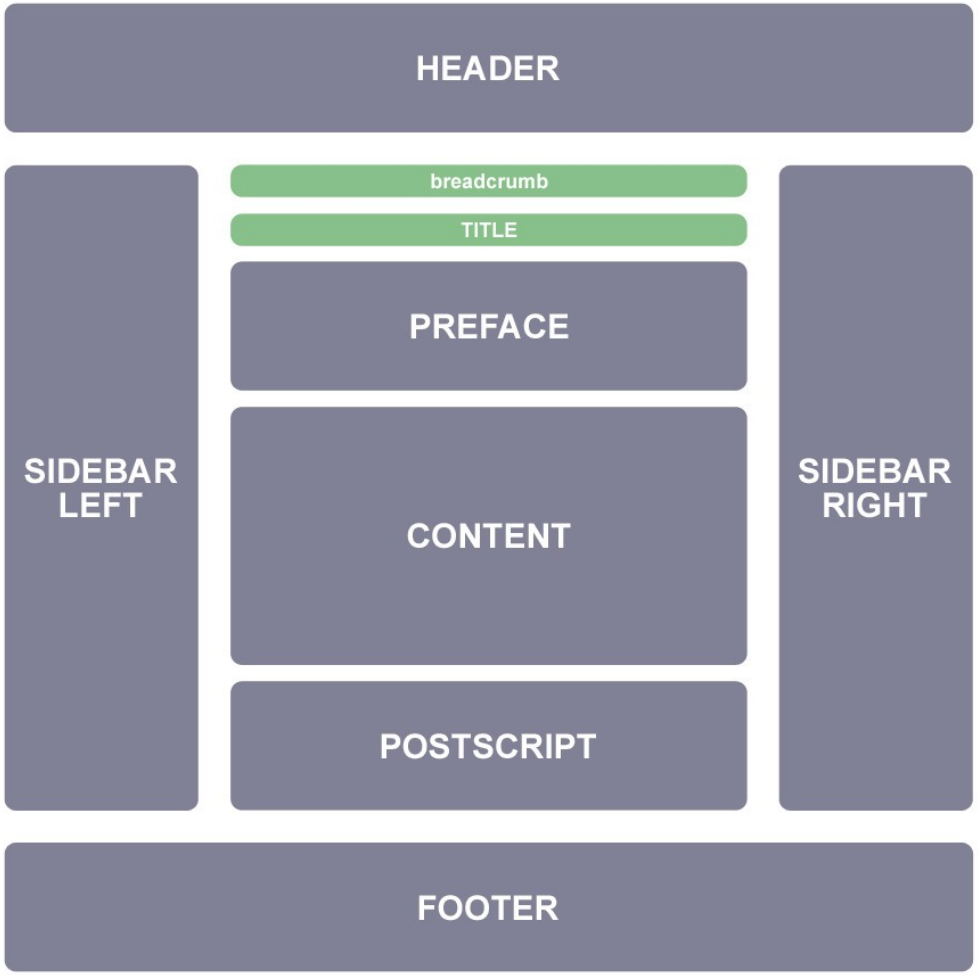
Template

Templates

- Template PHP et Twig
- {{…}} : évalue une expression
- {%...%} : instruction Twig (boucle, condition)
- {#...#} : commentaire
- Héritage :
  - Parent : {% extends ' ::base.html.twig' %}
  - Surcharge : {% block nomBlock %}
  - {{ parent() }}

Héritage : cas pratique

- base / content-type
- Layout / section du site
- Remplissage des régions



Templates : inclusion

- Inclusion de template :

```
{{ include('template', {'key' : data}) }}
```

- Invocation d'un contrôleur :

```
{{ render(
   controller( 'Bdle:Ctrl:action',
      {'key': data})
   )
}}
```

Templates : inclusion

- invocation d'un contrôleur asynchrone

```
{{ render_hinclude|_esi(
   controller('Bdle:Ctrl:action', {'key': data}))
}}
```

- hinclude.js
-
```
templating:
   hinclude_default_template:
   Bdle::hinclude.html.twig
```

Routage et template

- Relatives :

```
<a href="{{ path('route_key', { 'name': 'Pierre'
}) }}">
Lire cette entrée blog.
</a>
```

- Absolue:

```
<a href="{{ url('route_key', {'name': 'Pierre'})
}}">
Lire cette entrée blog.
</a>
```

Templates : liens

- {{ path|url('route_key') }}
- {{ path('route_key', {'name' : user.name}) }}
- {{ asset('images/logo.png')}}
- Combinaison asset('xxx') et parent() pour l'ajout de css/js

Structure de contrôle

```
{ % if data is not null %}
{ % endif %}
```

```
{ % set dl = data|length %}
{ % if dl > 10 %}
{ % elseif dl is divisible by(3)%}
{ % endif %}
```

Structure de contrôle

```
{ % for post in posts %}
{ % else %}
{ % endfor %}
```

```
{ % for digit in 0..9 %}
{ % endfor %}
```

```
{ % for record in ranking %}
   n°{{ loop.index }} : {{ record.line }}
{ % endfor %}
```

Templates : plus loin

- Surcharge de template de bundle app/Ressources/VendorNomBundle/views...
- Filtre : échappement :
  {{ data.from_user | raw }}
- Twig_extension_debug (activer dans conf)
  {{ dump(variables) }}

Liste des filtres

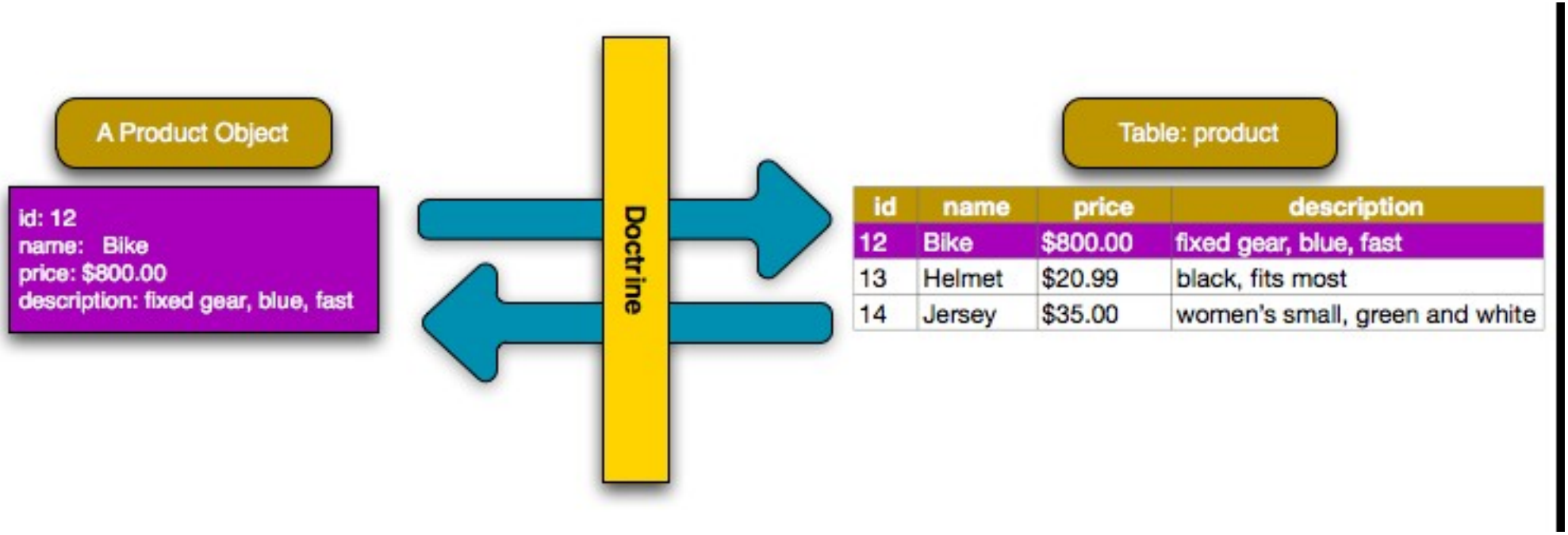
- abs, batch, capitalize, convert_encoding, date, date_modify, default, escape, first, format, join, json_encode, keys, last, length, lower, nl2br, number_format, merge, upper, raw, replace, reverse, round, slice, sort, split, striptags, title, trim, url_encode

ORM et Doctrine

Doctrine

- app/config/parameters.yml
- Création base de données
```
php app/console doctrine:database:create
```
- Création de classe entité
```
php app/console doctrine:generate:entity –
entity="DawanMyBundle:Training"
--fields="name:string(255) price:float
description:text"
```
- Création base de données
```
php app/console doctrine:schema:update --force
```

Doctrine



id	name	price	description
12	Bike	$800.00	fixed gear, blue, fast
13	Helmet	$20.99	black, fits most
14	Jersey	$35.00	women's small, green and white

Doctrine

- Persistance des données

```
$em = $this->getDoctrine()->getManager()
$em->persist($product)
$em->flush()
```

- Accéder aux données

```
$this->getDoctrine()
->getRepository('DawanStoreBundle:Product)
->find($id)
```

- Opérations

```
find, findOneBy{fieldName}, findBy{fieldName},
findAll, findBy(array('col'=>value), ... ),
```

Doctrine Query Language

```
•$this->getEntityManager()->createQuery(
'SELECT p FROM DawanStoreBundle:Product p WHERE
p.price > :price ORDER BY p.price ASC'
)->setParameter('price', '19.99')
->getResult();
```

Doctrine Query builder

- QueryBuilder

```
$repository->createQueryBuilder()
->where/setParameter/orderBy($param)
->getQuery()
```

Relations

- Définit par annotation
- Nombreux paramètres
- OneToOne, ManyToOne, ManyToMany
- Unidirectionnelle, bidirectionnelle

Relations

```
/**
 * @ORM\ManyToOne(
 *      targetEntity="Author",
 *      inversedBy="posts")
 * @ORM\JoinColumn(
 *      name="author_id",
 *      referencedColumnName="id")
 */
```

Relations

```
/**
 * @ORM\OneToMany(
 *      targetEntity="Article",
 *      mappedBy="author")
 */
```

Relations

- Lazy loading
- Objets mandataires

Doctrine

- Cycle de vie
```
* @ORM\HasLifecycleCallbacks()
* @ORM\prePersist
```
- pre/postRemove, ..Persist, ..Update, postLoad, loadClassMetadata
- Voir + :
  - extensions Doctrine
  - app/console help doctrine:object:action

Propel

- Proche de Doctrine2 : entités=>modèles
- Niveau d'abstraction légèrement moins élevé
- Commande console ex:
  - php app/console propel:database:create
- Classes Query / modèles

Formulaire

Formulaire

- Création d'un formulaire depuis le contrôleur:

```
$this->createFormBuilder($object)
   ->add(...)->...->getForm()
$this->render('template.html.twig',
   ['my_form'=>$form->createView()]);
```

- Rendu :

```
{{ form(my_form) }}
```

Patron de gestion

```
$form->handleRequest($request);
   if ($form->isValid()) {
      return $this->redirect(
         $this->generateUrl('task_success')
      );
} }
```

Formulaire : Validation

- Appel automatique (soumission de formulaire) : …/Ressources/config/validation.yml
- Appel manuel (utilise le service validator) : `$validator->validate($entityInstance)`
- Grand choix de contraintes : syntaxe variable
- Applicable sur attributs de classe ou getter
- Création de groupes de validation (permet une validation différente suivant le contexte)
- Possibilité d'utiliser une instance de contrainte :

```
use Symfony\Component\Validator\Constraints\Email;
```

Formulaire : type

- Ajout de groupe de validation au formulaire :

```
$this->createFormBuilder($users, array(
'validation_groups' => array('registration'),
))->add(...)
```

- Closure :

```
array('validation_groups' =>
   array('Dawan\\DawanBundle\\Entity\\User',
   'determineValidationGroups'),
```

- Types de champs ~ HTML5 élargi:

```
text(10), choix(6), date(4), autres(3)
+ groupes(2), cachés(2), base(2)
```

- Options varient suivant le type

Formulaire : avancé

- Prédiction de type suivant contraintes
- Rendu détaillé :

```
{{ form_errors(form) }}, form_row, form_rest
form_label, form_errors, form_widget //per row
```

- Classes de formulaire

```
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
```

- Déclaration de la classe de données :

```
public function
setDefaultOptions(OptionsResolverInterface $resolver)
{  $resolver->setDefaults(array(
   'data_class' => 'Acme\TaskBundle\Entity\Task',
   )); }
```

Formulaire : avancé

- Formulaire imbriqué

```
$builder->add('form2', new Form2Type());
$resolver->setDefaults(array([...],
'cascade_validation' => true));
```

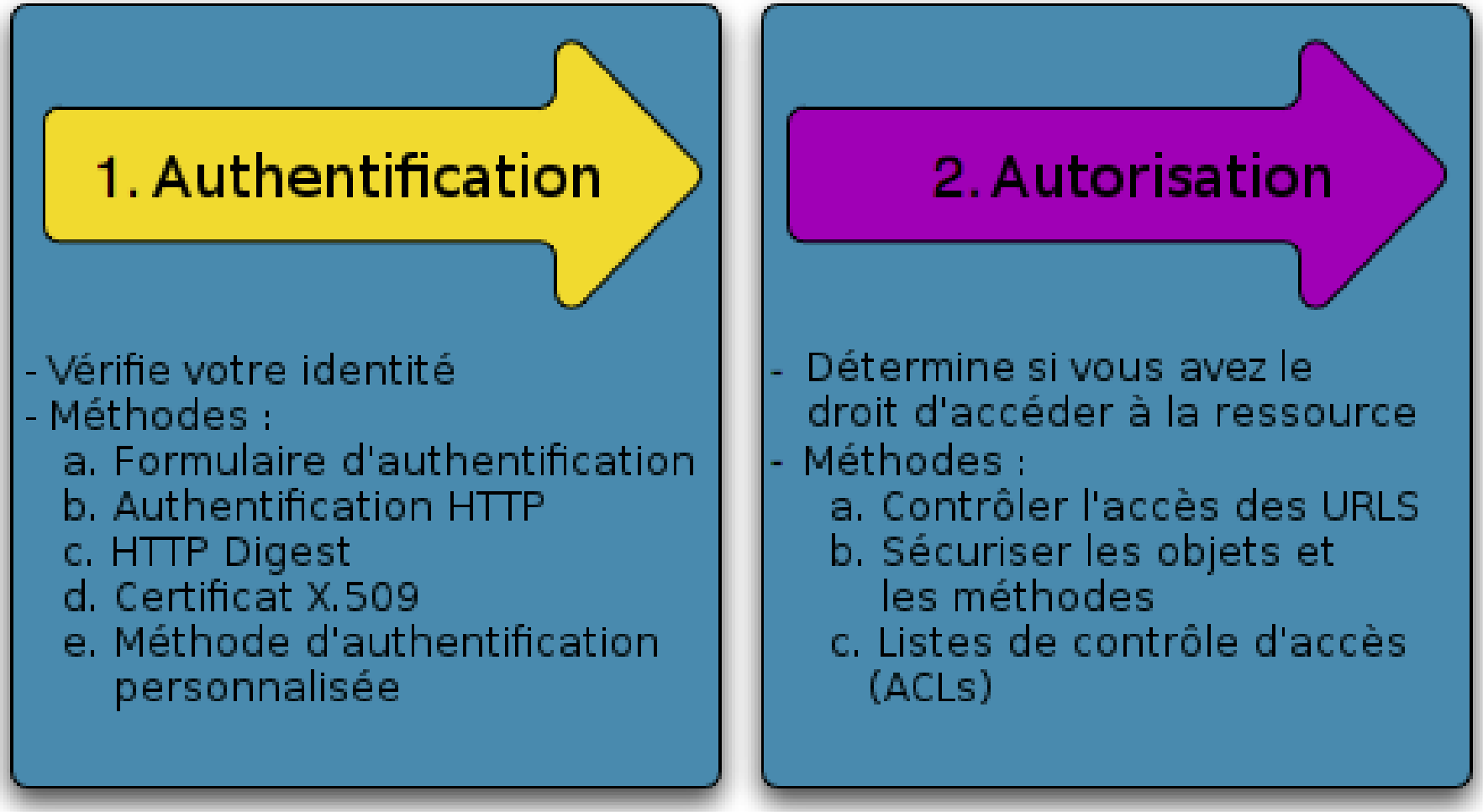
- Theming :

```
{% form_theme form
'DawanMyBundle:Form:fields.html.twig' %}
```

- surcharge : `{% block field_row %}`
- Nommage
- CSRF (automatique)

Sécurité

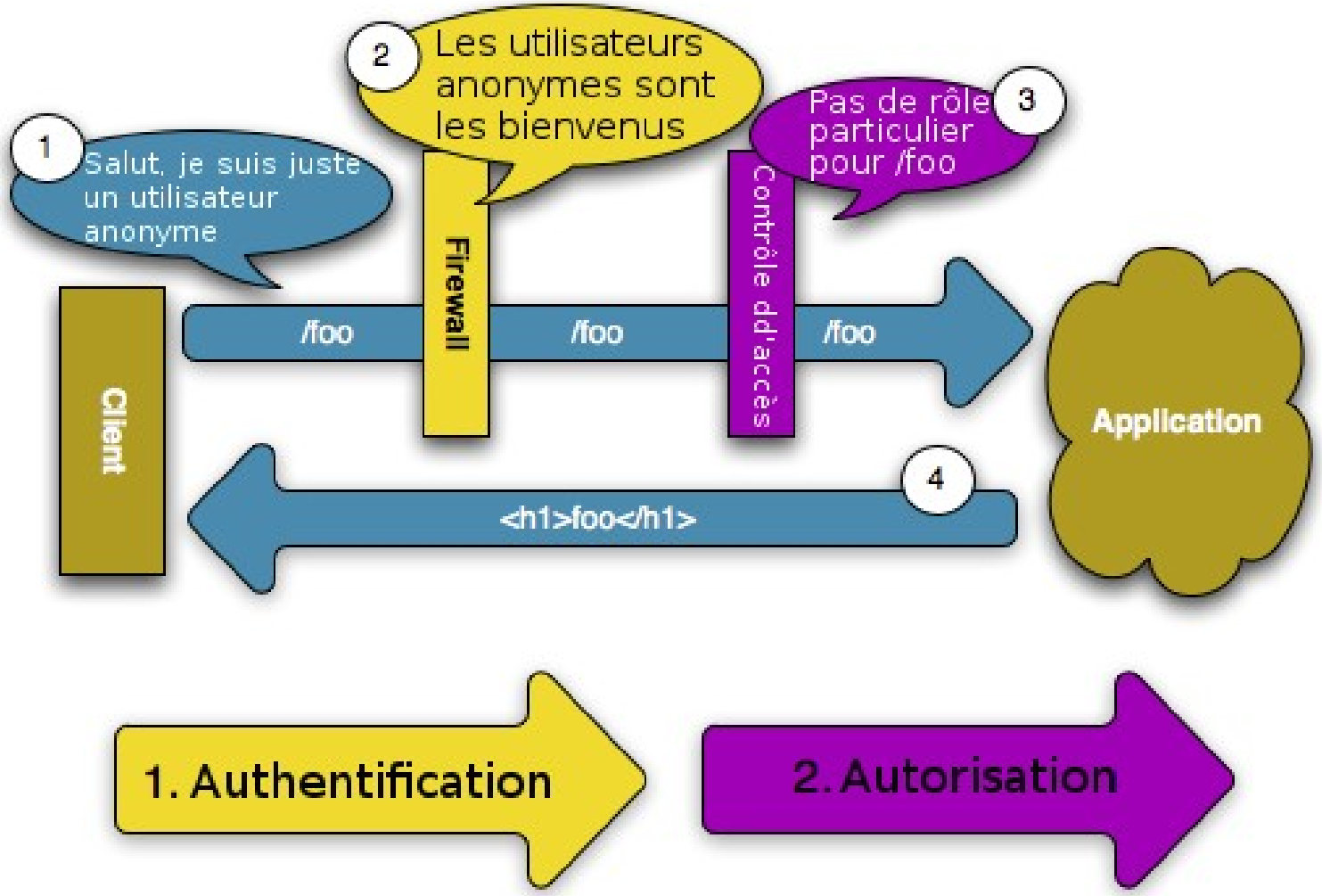
Principe


1. Authentification
- Vérifie votre identité
- Méthodes :
a. Formulaire d'authentification
b. Authentification HTTP
c. HTTP Digest
d. Certificat X.509
e. Méthode d'authentification personnalisée
2. Autorisation
- Détermine si vous avez le droit d'accéder à la ressource
- Méthodes :
a. Contrôler l'accès des URLS
b. Sécuriser les objets et les méthodes
c. Listes de contrôle d'accès (ACLs)

security.yml

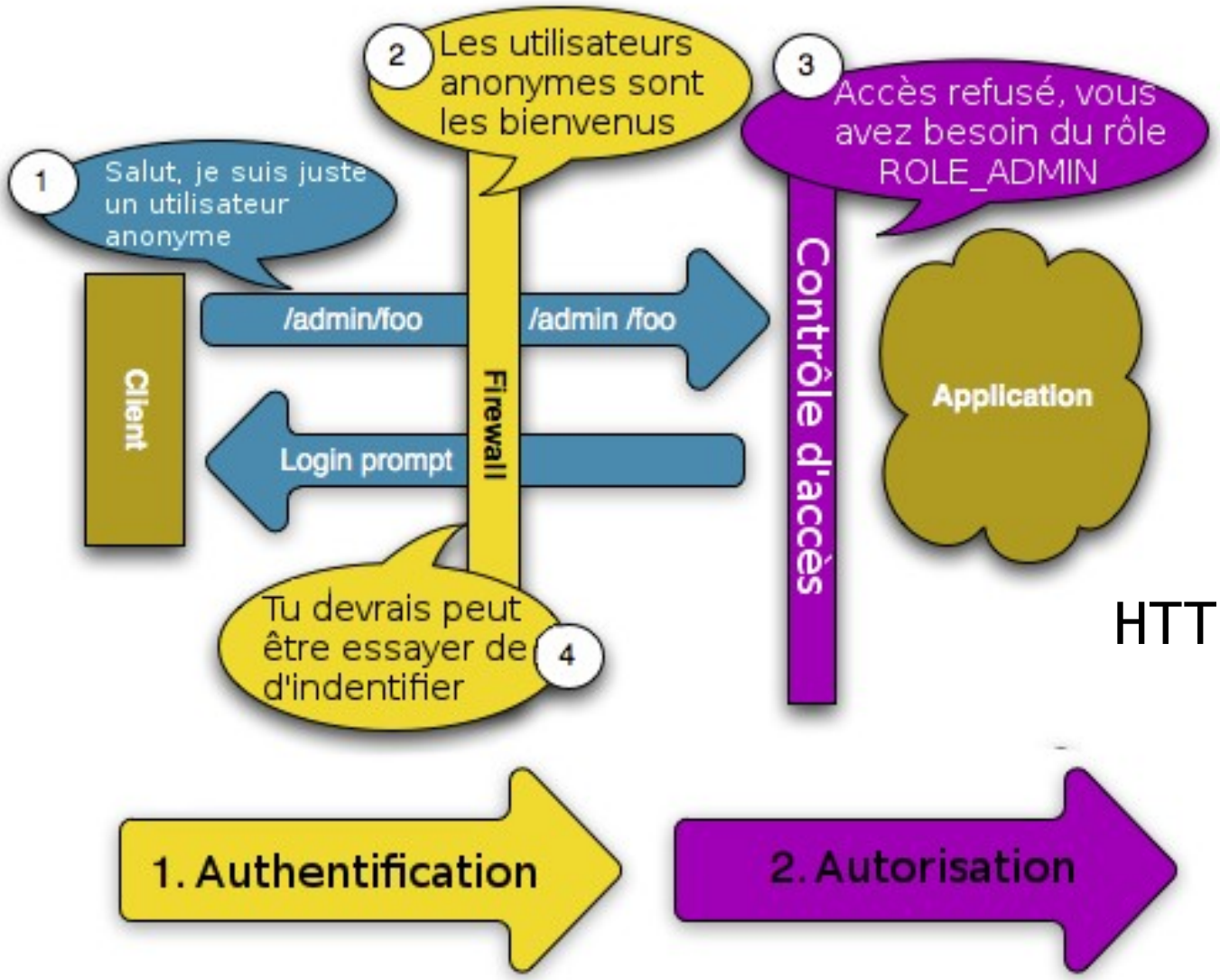
```
security:
  firewalls:
    secured_area:
      pattern:    ^/
      anonymous: ~
      http_basic:
        realm: "Secured Demo Area"
  access_control:
    - { path: ^/admin, roles: ROLE_ADMIN }
  providers:
    in_memory:
       memory:
         users:
           ryan: { password: ryanpass, roles: 'ROLE_USER' }
           admin: { password: kitten, roles: 'ROLE_ADMIN' }
  encoders:
    Symfony\Component\Security\Core\User\User: plaintext
```

Principe



HTTP 200 Ok

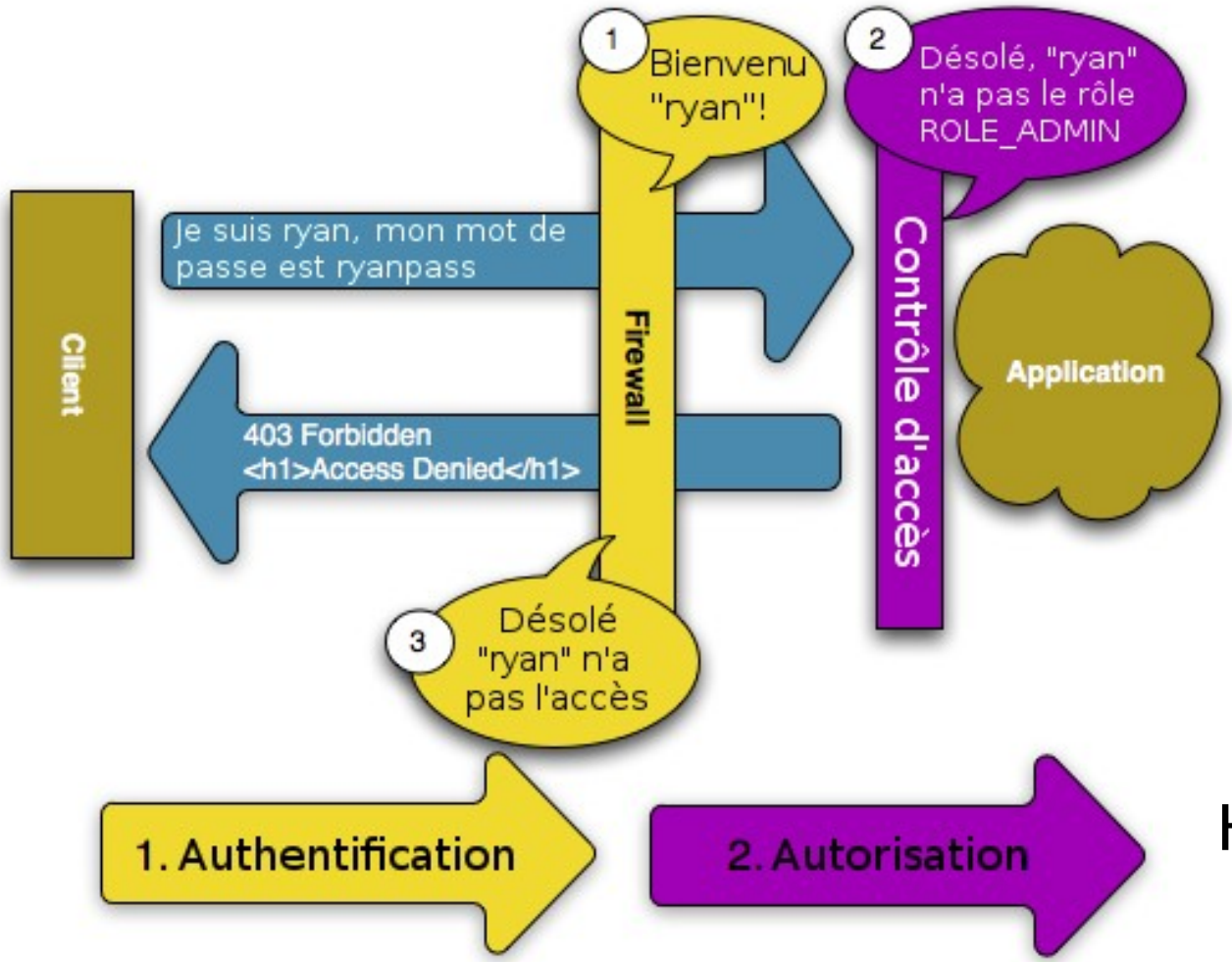
Principe



HTTP 401 Unauthorized

HTTP 302 Found

Principe



HTTP 403 Forbidden

Principe

- Pare-feu
  - Gère l'authentification
- Login
  - DOIT se trouver derrière un pare-feu
  - DOIT être accessible aux anonymes
- → access_control IS_AUTHENTICATED_ANONYMOUSLY ou
- → second pare-feu

security.yml

```
if (false === $this->get('security.context')
->isGranted('ROLE_ADMIN')) {
        throw new AccessDeniedException();
}
```

```
{% if is_granted('ROLE_ADMIN') %}
```

Tests

Tests et PHPUnit

- 1 test = 1 classe PHP
- Répertoire Tests du Bundle
- Commande :

```
phpunit -c app
src/Dawan/DemoBundle/Tests/Directory/DemoTest.php
```

Tests fonctionnels

- Extends WebTestCase
- static::createClient() => crawler
- Récupération de lien et clic

```
$link = $crawler->filter('a:contains("Greet")')
->eq(1)->link();
$crawler = $client->click($link);
```

- Récupération de formulaire et envoi

```
$form = $crawler->selectButton('submit')
->form();
$form['name'] = 'Lucas';
$crawler = $client->submit($form);
```

Tests fonctionnels

- Parcours du DOM :
  `filter($cssSelector), filterXPath, parent, children, first, last, siblings...`
- Fonctions de validation assertXXX :
  `assertGreaterThan(), assertCount(), assertTrue(), assertRegExp(), etc.`
- Création de formulaire :
  `$client->request('POST', 'url', array(...))`
- Naviguer :
  `$client->back(), forward(), reload()`

Tests fonctionnels

- Accès aux objets internes :
  `$client->getHistory(), CookieJar, Request, Response, Crawler, Container, Kernel, Profile`
- Suivi des redirections :
  `$client->followRedirect[s]()`
- Configuration de test : config_test.yml
- Configuration PHPUnit : phpunit.xml.dist

Cache

Accelération

- Byte code Cache : APC
- Class map / autoload.php
  - Composer dump-autoload --optimize
- ApcClassLoader
- Bootstrap file (màj auto avec composer install)

HTTP gateway cache

```
// app/AppCache.php
[ … ]
    return array(
        'debug'                  => false,
        'default_ttl'            => 0,
        'private_headers'        => array('Authorization',
'Cookie'),
        'allow_reload'           => false,
        'allow_revalidate'       => false,
        'stale_while_revalidate' => 2,
        'stale_if_error'         => 60,
          );
```

HTTP gateway cache

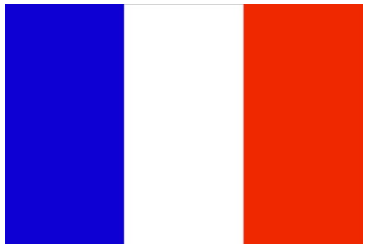
- Intercepte la requête et retourne une réponse devant l'application
- Sf2 en possède un écrit en PHP
- Varnish, nginx ou Squid (en mode reverse proxy)

HTTP Caching

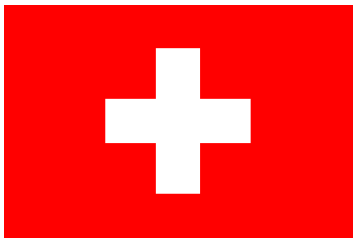
- En-tête HTTP:
  - Cache-Control
  - Expires
  - ETag
  - Last-Modified

Cache-Control

- $response->setPublic() / setPrivate()
- set[Shared]MaxAge(600)
-



i18n



Traduction

- Fichier de traduction
  - nom_catalogue.langue.format

```
# app/config.yml
framework:
    translator:      { fallback: "%locale%" }
```

```
# Resources/translations/messages.fr.yml
Symfony2_is_great: J'aime Symfony2
```

```
$this->get('translator')->trans('Symfony2_is_great');
```

Chaine de substitution

```
'Hello %name%': Bonjour %name%
```

```
$this->get('translator')->trans('Hello');
```

Conteneur de services

Définition

- Service : objet pouvant être mis à disposition de toute l'application
- Conteneur de service (Dependency Injection Container) : objet gérant l'instanciation des services

Exploitation

- Depuis le contrôleur :

```
$log = $this->get('logger');
$log->notice('J\'utilise un service !') ;
```

- Une classe ContainerAware :

```
class CustomClass extends ContainerAware {
   public function customMethod() {
      $this->container->get('router') ;
   }
}
```

Plus d'informations sur http://www.dawan.fr
Contactez notre service commercial au **0800.10.10.97** (prix d'un appel local)

DAWAN, 28, rue de Strasbourg, 44000 Nantes
formation@dawan.fr