

2017-2018

Haute-École Arc

# Développement Web

Technologies d'interaction

David Grunenwald <david.grunenwald@he-arc.ch>

24 septembre 2018

# Table des matières

<b>1</b>	<b>Présentation</b>	<b>3</b>
<b>2</b>	<b>Introduction aux frameworks PHP</b>	<b>7</b>
<b>3</b>	<b>Laravel</b>	<b>27</b>
<b>4</b>	<b>HTML 5</b>	<b>34</b>
<b>5</b>	<b>JavaScript et DOM</b>	<b>36</b>
<b>6</b>	<b>HTTP et AJAX</b>	<b>45</b>
<b>7</b>	<b>jQuery</b>	<b>55</b>
<b>8</b>	<b>Syndication (RSS)</b>	<b>60</b>
<b>9</b>	<b>Services Web</b>	<b>66</b>
<b>10</b>	<b>Responsive Web Design</b>	<b>71</b>
<b>11</b>	<b>HTTPS</b>	<b>77</b>
<b>12</b>	<b>Risques</b>	<b>81</b>
<b>13</b>	<b>Ruby on Rails</b>	<b>87</b>

# 1 Présentation

## Programme

- Frameworks MVC : Laravel, (Rails), Django, ...
- HTML5 : vue d'ensemble
- Javascript : AJAX, DOM, JSON, Node.js, jQuery
- (Syndication : RSS, Atom)
- Déploiement et configuration Serveur
- (Responsive) Web Design
- Webservices : REST vs SOAP
- Sécurité : Technologies, prévention des risques courants
- Vos souhaits ?
- Slides cours : [ghpages](#), Source : [github/HE-Arc](#)

## Organisation

- Cours
- Workshops intervenants externes
  - Automatisation du déploiement (R. Emourgeon) ?
  - Flask (M. Amiguet) en janvier 2019
  - Webdesign (M. Schmalstieg) ?
  - Vue.js ? React ? AngularJS ?
- 2 Projets
  - 2 frameworks : Laravel & Django (ouvert à d'autres propositions)
  - Groupes de 3, [30h](#) par personne et par projet
  - Présentation de 20min
- Vos présentations ? Vos propositions ?

## Projets

- Faire pour apprendre
- Les rôles dans une équipe de développement web

- Ne pas réinventer la roue ou tout faire soi-même
- Critères d'évaluation d'un projet
- En profiter pour apprendre des choses qui vous intéressent
- Avant le 1er octobre :
  - Avoir un compte github avec une [clé SSH](#) (indispensable au déploiement)
  - Constitution des équipes de 3 personnes
  - Choix du projet
  - Forge : Créer projet sur github dans l'entité [HE-Arc](#)
  - [S'inscrire](#)
- Offre d'essai Pluralsight 90 jours sur [MS Imagine](#)

## Choix des projets

- Contrainte : appli basée sur des données
- Choix
  - Besoin réel (ex : Concours robots P1 TIN)
  - Données existantes : [Inventaire](#), [dbpedia](#), [opendata](#), DB Bikini Test à dispo
  - S'inspirer de l'existant :
    - \* [Product Hunt](#), [blinklist](#), [makeuseof](#), ...
    - \* Volées précédentes : [2016-18](#), [2015/16](#), [2014/15](#)

## Calendrier

Semaine	Automne	Semaine	Printemps
38		8	
39	Projet Laravel	9	
40		10	
41		11	
43		12	
44		13	
45		14	
46		15	Présentations
47	T. Autonome	17	Présentations
48		18	
49		19	T. Autonome
50	Présentations	20	Examens
51	Présentations	21	Début TB
2			
3	Projet Python		
4			

Semaine	Automne	Semaine	Printemps
5	T. Autonome		
6	Examen		

## Jalons (Objectifs à atteindre pour le début de la semaine)

- 1
- 2 Objectifs et maquettes
- 3 Authentification et 1er déploiement
- 4
- 5 Modèles avec relations (au moins 3)
- 6
- 7 Minimal Viable Product
- 8
- 9
- 10
- 11 Rendu projet, Présentation

## Conseils

- Le plus simple possible
- Pas trop de données
- Application crédible (vraies données, cas réalistes)
- Projet à blanc pour la prise en main du framework
- [Maquettes](#)
- [Organisez](#) l'utilisation du dépôt
- Le temps disponible à l'horaire ne suffira pas !
- Essayez de commit avec la même identité
- Signalez dans le commit msg si vous n'êtes pas l'auteur
- Le déploiement est long : commencez tôt !
- Il est moins risqué travailler plus au début du projet qu'à la fin !

## Évaluation

- User Experience : 50%
  - Utilisabilité : Efficacité, efficience, satisfaction
  - Design UI
- Code : 30%
  - Absence bugs, qualité code, lisibilité
  - Respect conventions et bonnes pratiques

- Déploiement, configuration
- Gestion de projet : 20%
  - Fichiers versionnés, messages de commit
  - Issues, planification, travail en équipe
  - Documentation (wiki)
  - Investissement, volume de travail
- Bonus (ceux qui vont plus loin) : 0-20%
  - WebSockets ou autre API HTML5,
  - Webservice, ...
- Tous les membres d'un groupe n'ont pas forcément la même note

## Présentation facultative

- Facultatif, ne peut qu'augmenter la moyenne
- DOIT être annoncé au semestre d'automne
- Un thème absent du cours
- 2 à 4 personnes
- Une présentation claire avec démo (printemps)
- Un exercice d'application
- Critiques et discussion
- Au plus tôt :
  - Constitution des équipes
  - Proposer 1 à 3 thèmes
  - [Proposer](#) le(s) thème(s) de présentation et l'équipe

## Mon expérience en développement web

- [Questionnaire](#) obligatoire (votre username github vous y sera demandé)

M E R C I !

## Sources

## 2 Introduction aux frameworks PHP

### Framework

- Fonctionnalités similaires pour de nombreuses applis
- Composants de haut-niveau réutilisables (faible couplage)
- Règles de codage et d'architecture
- Code sûr et efficace
- Facilite les tests et la gestion de projets complexes
- Utilisation de Design Patterns dès que possible
- Comportement par défaut
- Extensible
- Principe d'inversion de contrôle

Différences entre framework et library sur [Stack Overflow](#) ou [artima developer](#).

### Design Patterns et webdev

- Inversion de contrôle (IoC)
- Model View Controller
  - M : Accès aux données, logique métier
  - V : Templates des pages à générer
  - C : Orchestration, transfert des infos
- Front Controller
  - Traitement et dispatch des requêtes
  - (bootstrap, ré-écriture des URL, ...)
- [Object Relational Mapping](#)
  - Active Record, Table Data Gateway, Data Mapper, ...
- [UI Patterns](#)

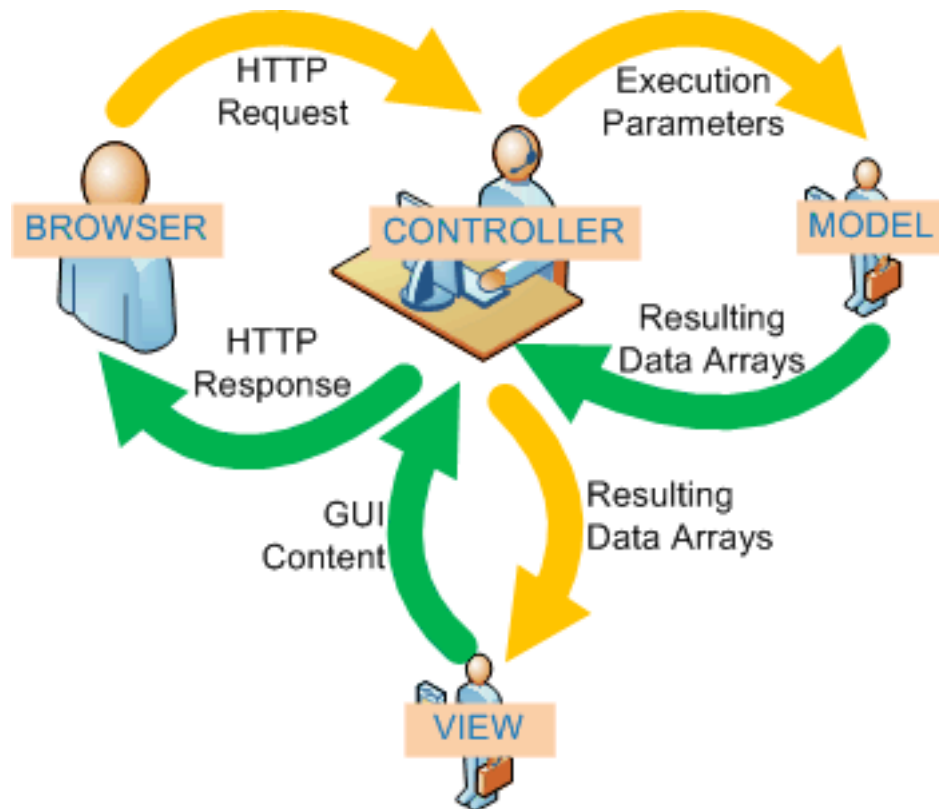


FIG. 2.1 : ``MVC''



## MVC for webdev

### Conventions

- Nommage
  - Classes
  - Base de données
  - Fichiers et dossiers
- ROUTES : `http://app.host.tld/controller/action[/key/val]`
- Arborescence :
  - Imposée ou libre selon frameworks
  - Pas de code (minimum) sous la racine web
- Conventions obligatoires ou non, mais RECOMMANDEES dans tous les cas

### Bonnes pratiques

- Heavy Model, Light Controller
- Don't Repeat Yourself
- You Ain't Gonna Need It
- Convention Over Configuration
- Keep It Simple and Stupid
- [12 factor app](#) - [fr](#)


### Pretty ( | smart | clean | formatted) URL

- Les URL doivent être explicites :
  - Manipulées par l'utilisateur
  - Utilisées pour le référencement
- Cohérence avec l'implémentation MVC :

`http://app.host.tld/controller/action[/key/val]`

- Le routage (routing)
  - Le Front Controller reçoit toutes les requêtes (URL rewriting)
  - Il les dispatche vers les contrôleurs


## Smart URL & SEO



**seomoz.org**  
Read SEOMoz. Rank Better

### SEO Cheat Sheet: Anatomy of A URL

1  
SEO-FRIENDLY URL



- 1
- 2
- 3
- 4
- 5
- 6
- 7

`http://store.example.com/topics/subtopic/descriptive-product-name#top`

- 1 Protocol
- 2 Subdomain
- 3 Domain
- 4 Top-Level Domain
- 5 Folders / Paths
- 6 Page
- 7 Named Anchor

**Keyword Priority<sup>1</sup>**  
Observed Google priority of keyword placement:


(1) Domain  
(2) Subdomain  
(3) Folder  
(4) Path/Page

**SEO Tips for URLs**

- Use **subdomains** carefully. They may be treated as separate entities, splitting domain authority.
- Separate **path** & **page** keywords with hyphens ("-").
- Anchors** may help engines understand page structure.
- Keyword effectiveness in URLs decreases as URL length and keyword position increases.<sup>1</sup>

<sup>1</sup> SEOMoz correlational data (2009)

2  
OLD DYNAMIC URL



- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 7
- 7

`http://www.example.com/index.php?product=1234&sort=price&print=1`

- 1 Protocol
- 2 Subdomain
- 3 Domain
- 4 Top-Level Domain
- 5 Page / File Name
- 6 File Extension
- 7 CGI Parameters

**Popular TLDs<sup>2</sup>**

.com - commercial  
.net - infrastructure  
.org - non-profit  
.edu - schools  
.info - informational  
.biz - small business  
.name - personal sites

**Popular ccTLDs\***

.cn - China  
.de - Germany  
.uk - United Kingdom  
.nl - Netherlands  
.eu - European Union  
.ru - Russian Federation  
.ar - Argentina

**Popular Extensions**

.htm - Static HTML  
.html - Static HTML  
.php - PHP code  
.asp - ASP code  
.aspx - ASP.NET  
.cfm - ColdFusion  
.jsp - Java Code

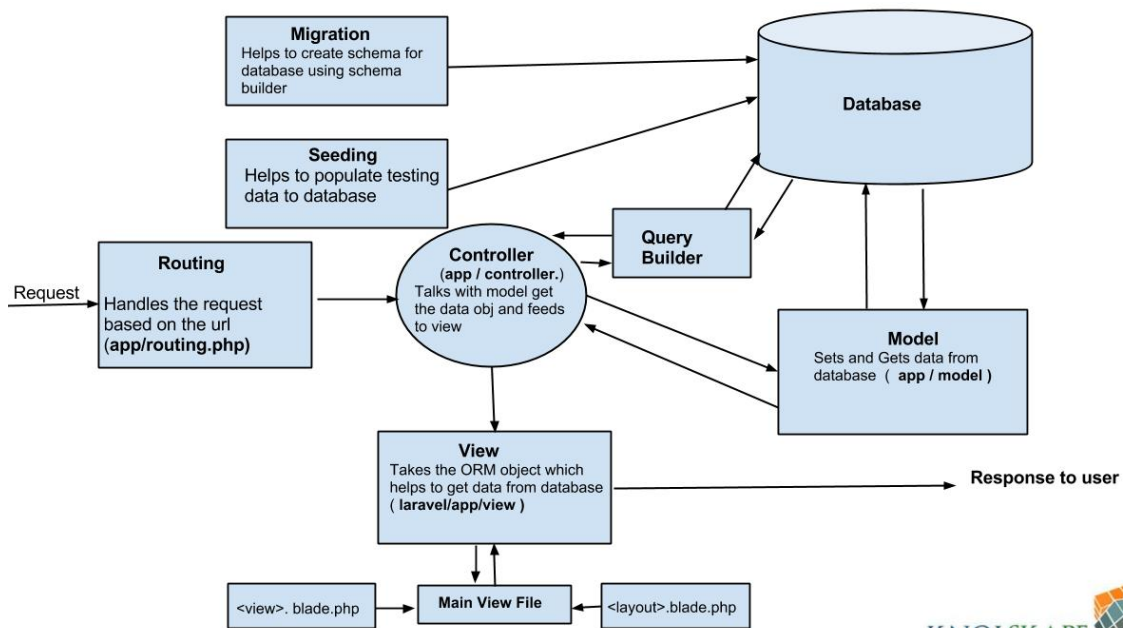
<sup>2</sup> Verisign domain report (2009)      \* ccTLD = Country Code TLD

© 2009 SEOMoz · www.seomoz.org · Read SEOMoz. Rank Better.

## Autres Services

- Migrations : Evolutions de la strucutre de la BDD
- Tests
- Génération, validation et traitement de formulaires
- Authenfication, Sessions, Permissions, Roles, ACL
- Pagination
- I18n
- Génération de code
- Mail
- Connecteurs aux webservices
- Captchas
- Loggers
- ...

## Exemple d'architecture : Laravel




## Performance

- Un framework web est lent :
  - Rendu d'une page nécessite de traverser tout le code
  - Pour chaque requête toute l'appli est chargée
  - Plus de code qu'une appli standalone
  - Plus de requêtes
- Solutions
  - Cache de pages, d'opcode
  - Jointures ORM, vues, procédures stockées
  - Outils d'optimisation : YSlow, page speed, mytop

## Frameworks PHP

- Lesquels connaissez-vous ?
- Lesquels avez-vous utilisé ?
- Pourquoi y en a-t-il tant ?

L'explication donnée par Joe Gregorio pour [le langage Python](#) est : « parce que c'est facile. » Dans les faits, cela montre également une maturité de la plateforme.

*There are people who actually like programming. I don't understand why they like programming.* Rasmus Lerdorf 

- PHP-FI *Forms Interpreter*
- PHP 3, réécrit en C++
- PHP 4 *Zend Engine*, fausse POO
- PHP 5, vraie POO
- PHP 5.1, PDO
- PHP 5.2, JSON
- PHP 5.3, goto et namespace
- PHP 5.4, [] et trait
- PHP 5.5, yield
- PHP 6, Unicode 🐛, 🍁, 🐧
- PHP 7, que du rêve !

Il y a plus de vingt ans, Rasmus Lerdorf bricola un outil pour savoir qui consultait son CV. Zend, c'est à dire *ZEev* et *aNDi*, ont réécrit PHP et qui allait devenir PHP 3 le précurseur du langage de prédilection pour créer sur le web.

PHP a évolué depuis pour devenir ce qu'il est aujourd'hui. Sa popularité est liée au fait qu'il est simple à mettre en œuvre, gratuit et libre. Tout un tas de modules est fourni avec pour faire de l'imagerie, des bases de données, du XML, etc.

Et plus encore sur la page [History of PHP](#) et [Wikipedia : PHP](#).

Les différentes moutures de PHP 7 offrent ceci, entre autres.

- PHP 7, performances
- PHP 7.1, void
- PHP 7.2, sodium

L'évolution de PHP a fait que les usagers du langage, créateur de *frameworks*, d'outils (comme [Composer](#)), ont senti le besoin d'émettre des recommandations afin d'aller vers un plus interopérable.

Durant ce cours, nous allons vous embêter avec PSR-1, PSR-2 et PSR-4.

## Quiz

Qui est qui ?

oOops, ceci n'a rien à voir avec le cours.

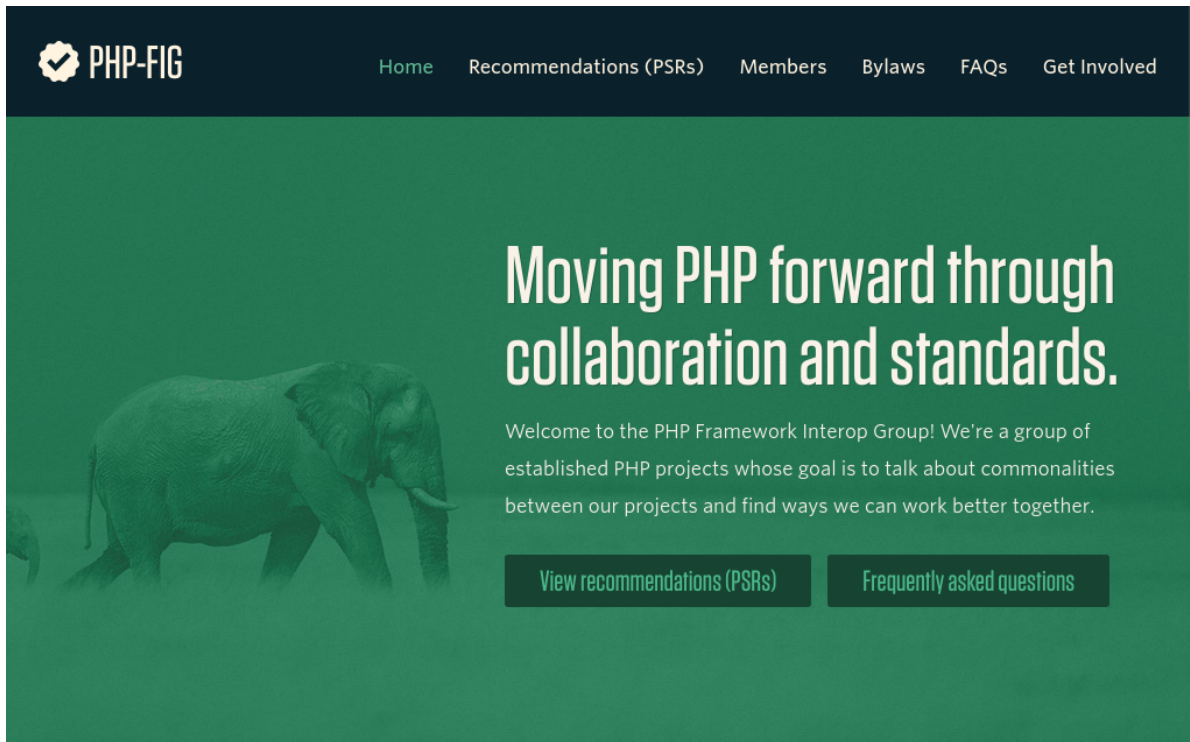


FIG. 2.2 : PHP Framework Interop Group



FIG. 2.3 : [source](#)





Donc, ce ne sont pas Gandalf (sans sa barbe) et Saruman mais bien Sir Tim Berners-Lee et Vinton Cerf, responsables du (World Wide) Web et de l'Internet.

### Qu'est-ce qu'**Internet** ?

- un réseau IP

### Qu'est-ce que le **World Wide Web** ?

- URI/URL, des identifiants uniques
- HTML, un langage de publication
- HTTP, un protocole d'échange de texte (ou *HyperText*)

## Préparatifs

<https://github.com/HE-Arc/php-intro-framework/>

```
$ sudo systemctl start httpd
$ cd /var/www/html
$ git clone \
> https://github.com/\
> HE-Arc/php-intro-framework
```

```
$ cd php-intro-framework
$ open http://localhost/php-intro-framework
```

Les exemples suivant travaillent sur le code disponible dans le dépôt [HE-Arc/php-intro-framework](#).

```
$ curl -v "http://he-arc.ch/?id=25"
> GET /?id=25 HTTP/1.1
> Host: he-arc.ch
>
< HTTP/1.1 200 OK
< Content-Type: text/html; charset=utf-8
<
```

```
<!DOCTYPE html>
<title>HE-Arc</title>
<p>Hello
```

HTTP est un protocole texte plutôt simple, jugez plutôt :

Ce que nous voyons est une connexion TCP/IP au serveur `he-arc.ch`. Une fois la connexion établie, il envoie en texte ASCII les entêtes HTTP puis deux retours à la ligne (ce qui correspond à une ligne vide). La requête HTTP commencent toujours par la demande, ici `GET /index.php?page=equipe&id=25 HTTP/1.1` puis les entêtes, ici : `Host: www.he-arc.ch`. La réponse du serveur est du même type, le code de réponse (`HTTP/1.1 200 OK`), les entêtes, une ligne vide puis le contenu.

La demande et les entêtes sont en US-ASCII mais le corps peut être encodé autrement, ici c'est dit dans l'entête `Content-Type: text/html; charset=utf-8`.

### Fait #1

PHP parle HTTP.

Le fichier `index.php` est le code PHP le plus simple qui soit. Simple au sens du niveau de compréhension de PHP et d'une forme de complexité.

```
<?php // 00-base

// Lecture de la query string `page=<XX>&id=<YY>`.
$page = $_GET["page"] ?? null;
$id = (int) ($_GET["id"] ?? 0);

// Connexion à la base de données.
$db = new PDO("sqlite:../users.db");
```

```
// Page HTML
?>
<!DOCTYPE html>
<meta charset=utf-8>
<title>HE-Arc</title>
<?php
// Contenu
if ("equipe" === $page):
    $query = $db->query("SELECT * FROM `personnes` WHERE `id` = :id;");
    $query->execute(compact('id'));

    $personne = $query->fetch(PDO::FETCH_OBJ);
?>
<p><a href="<?php echo $_SERVER["PHP_SELF"] ?>">retour</a>
<h1>Équipe</h1>
<h2>
    <?php echo $personne->prenom ?>
    <?php echo $personne->nom ?>
</h2>
<p>
    
<?php
else:
?>
    <h1>Accueil</h1>
    <ul>
        <li><a href="?page=equipe&id=1">Yoan Blanc</a>
        <li><a href="?page=equipe&id=2">Yoan Blanc</a>
    </ul>
<?php
endif
```

### Fait #2

PHP est un langage de template.

Pour preuve, il faut ouvrir une balise <?php pour commencer la partie code.

Avec la pratique, on a réalisé que mélanger la logique métier et celle d'affichage n'est pas optimal car difficile à lire et maintenir.



## Séparation métier/affichage

```
<?php // 01-includes/index.php

// ...

include "templates/entete.html";

if ("equipe" === $_GET["page"]) {
    // SELECT FROM u WHERE id=$_GET["id"]
    // ...
    include "templates/equipe.html";
} else {
    // ...
    include "templates/accueil.html";
}
```



Quel est le problème avec cette solution ?  
([Source de l'image](#))

## Sécurité des templates

- *Principle of Least Privilege* ( [polp](#) )
- Intégration faite par un graphiste, société externe

Dans ce le cas présent rien ne nous empêche de mettre de la logique métier dans nos fichiers de *template*, car ils sont faits de PHP eux aussi.

```
{# 02-twig/templates/collaborateur.html #}  
{%- extends "base.html" -%}
```

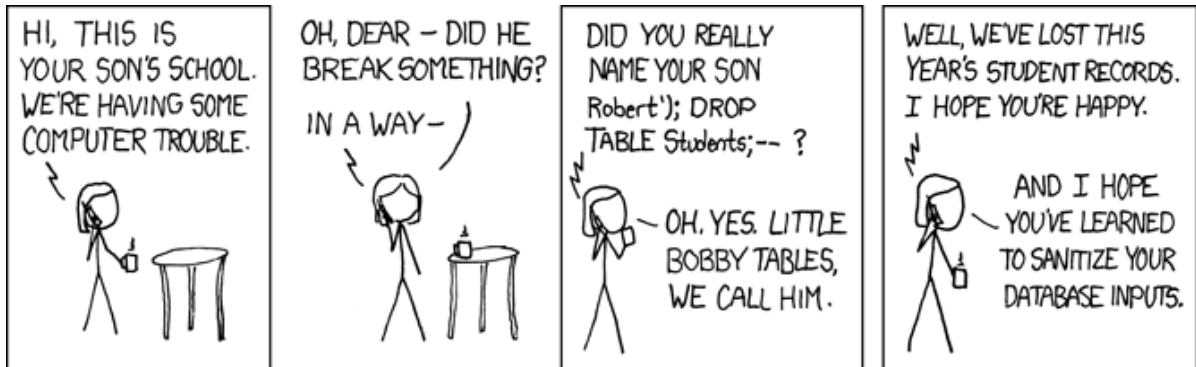
```
{% block corps -%}  
<p><a href="#">retour</a>  
<h1>Équipe</h1>  
<h2>  
    {{- personne.prenom -}}  
    {{ personne.nom -}}  
</h2>  
<p>  
{% endblock -%}
```

La page est réalisée avec [Twig](#) <2.0. À partir de la version 2.0, il faut utiliser un *autoloader* externe, comme celui de *composer* (voir ci-dessous).

Le code est un poil plus propre du côté de nos *templates* qui ne peuvent plus exécuter de PHP sauf ce qu'on leur autorise, ici `md5` et `strtolower`. Voir [02-twig/index.php](#).

```
<?php // 02-twig  
  
require_once 'Twig/lib/Twig/Autoloader.php';  
Twig_Autoloader::register();  
  
// ...  
  
// Configuration de Twig  
$loader = new Twig_Loader_FileSystem("templates");  
$twig = new Twig_Environment($loader);  
  
// Ajout des filtres md5 et strtolower qui sont les fonctions PHP du même nom.  
$twig->addFilter(new Twig_SimpleFilter('strtolower', 'strtolower'));  
$twig->addFilter(new Twig_SimpleFilter('md5', 'md5'));  
  
// variable globale  
$titre = "HE-Arc";  
  
// Contenu  
if ("equipe" === $page) {  
    // ...  
    $personne = // ...  
  
    echo $twig->render("equipe.html", compact("titre", "personne"));  
} else {
```

```
$personnes = // ...  
  
echo $twig->render("accueil.html", compact("titre", "personnes"));  
}
```



Problème d'injection SQL.

Effectuer des requêtes MySQL à la main ou devoir connaître tous les champs crée beaucoup de redondance et de failles de sécurité potentielles.

Une solution est d'ajouter une couche d'abstraction qui va cacher la structure réelle de notre base de données et offrir une interface orientée objet. Un *Object-Relational Mapping* ou ORM(3) dans le jargon.

```
<?php  
// Ne dites plus  
$query = $db->query(  
    "SELECT * FROM `personnes` ".  
    "WHERE `id` = :id;"  
);  
$query->execute(compact('id'));  
$personne = $query->fetch(PDO::FETCH_OBJ);  
  
// Mais dites plutôt  
  
// RedBean  
$personne = R::load('personnes', $id);  
// ou Doctrine  
$personne = $om->find('Personne', $id);
```

## Object-Relational Mapping

- [RedBean](#)
- [Doctrine](#) (ORM, ODM)

- Eloquent ORM
- etc.

Une bibliothèque qui va créer ce lien entre les mondes objet et relationnel ou document (généralement MongoDB). Il en existe toute une foule.

```
<?php // 03-redbean/index.php
require 'RedBean/rb.php';
R::setup("sqlite:../users.db");
// ...
if ("equipe" === $page) {
    $personne = R::load("personnes", $id);
    echo $twig->render(
        "equipe.html",
        compact("titre", "personne")
    );
} else {
    $personnes = R::find("personnes");
    echo $twig->render(
        "accueil.html",
        compact("titre", "personnes")
    );
}
```

## URI as UI

Pensez à Wikipedia.

Les adresses des pages font partie de l'expérience utilisateur. Un utilisateur doit être capable d'imaginer le contenu de la page en lisant l'URI. Certainement, ce que vous faites avant de cliquer sur un lien.

## Comment humaniser ?

/index.php?page=equipe&id=42

La personne avec l'identifiant 42 aura également un *slug* unique créé à partir de son nom, ici jean-bon.

La solution à notre problème est de demander au serveur web de réécrire les URL pour nous.

## Réécriture d'URL

```
# 04-routes/.htaccess

# mod_rewrite
RewriteEngine on
RewriteBase /php-intro-framework/04-routes/

RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php/$1 [L,QSA]
```

Apache le fait via `mod_rewrite` et Nginx `try_files`.

```
// 04-routes/index.php

$uri = $_SERVER['REQUEST_URI'],
$matches = [];

preg_match(
    "#^/(?P<page>[^/]+)/(?P<slug>[^/]+)/?#",
    $uri,
    $matches
) or die('Arrrrrrgh');

echo call_user_func_array(
    $matches['page'],
    [$matches['slug']]
);
```

Le code complet va nettoyer l'URI et définir les fonction correspondant aux pages possibles.

## Routing

Lien entre les adresses (URI) et des actions dans le code.

a.k.a. the *Front Controller*.

En pratique, les actions ne sont pas des fonctions mises à plat mais sont encapsulées dans une classe qu'on nomme un contrôleur. Faire ainsi permet de regrouper logiquement les fonctions et éviter d'utiliser d'affreux éléments tel que `global`.

## Modèle - Vue - Contrôleur

- Modèle : l'ORM qui s'occupe de notre base de données

- Vue : les templates qui affiche les données
- Contrôleur : une classe qui définit quoi faire en fonction des entrées utilisateur (URI, formulaire, etc.)

MVC(4) vient des applications bureau et ne représente pas toujours le fonctionnement dans le monde du web. Par exemple, Django, un framework Python, se décrit comme étant *Modèle - Template - Vue*(5).

Les frameworks web en PHP (ou d'autres langages) reposent majoritairement sur ce paradigme.

## Composer

Gestionnaire de paquets pour PHP : [getcomposer.org](https://getcomposer.org)

Maintenir notre répertoire de *vendor* ainsi que les `require` est peu pratique. Voici qu'entre en scène [Composer](#), le gestionnaire de paquet pour PHP. [Packagist](#) est le dépôt en ligne de paquets public et utilisé par défaut.

### composer.json

```
{
    "require": {
        "twig/twig": "^2.0",
        "gabordemooij/redbean": "^4.3",
    }
}
```

Nos dépendances sont ainsi matérialisées dans le projet et peuvent être installée, ou mises à jour simplement.

En principe les numéros de version respectent le [SemVer](#) (*Semantic Versioning*) et les différents signes permettent de sélectionner une ou plusieurs versions (voir [Version and constraints][<https://getcomposer.org/doc/articles/versions.md>]).

```
$ composer install
```

puis

```
<?php // 05-composer/index.php
```

```
require 'vendor/autoload.php';
```

```
use RedBeanPHP\Facade as R;
```

Enfin, nous pouvons réduire le nombre de `require` et `include` à un seul, en laissant soin à l'*auto-loader* de charger le bon fichier à la demande. Tout ceci est spécifié dans [PSR-4](#). Ainsi, les définitions de Twig sont présentes et il nous suffit d'obtenir la classe `R` depuis [RedBean](#).

## Front-Controller

Utilisation de [FastRoute](#) (voir [06-fastroute/index.php](#)).

```
$ composer require nikic/fast-route
```

FastRoute repose sur un système proche de celui que nous avons utilisé jusqu'ici. D'autres systèmes, tels que Aura.Router pour ne citer que lui, reposent sur la spécification [PSR-7](#). Cette dernière décrit l'interface objet d'un message HTTP, tant au niveau de la requête que de la réponse.

Si ça ajoute, une bonne couche de complexité, l'énorme avantage offert par cette idée là est de déléguer le rendu d'une page, ni echo, ni header, Donc il est envisageable de pouvoir tester (au sens de test unitaire), notre *FrontController*.

D'autre part, le `call_user_func_array` d'avant n'était pas très solide,

```
<?php // 06-fastroute/index.php
// ...
use function FastRoute\simpleDispatcher;
use FastRouter\Dispatcher;

$dispatcher = simpleDispatcher(function($r)
{
    $r->addRoute('GET', '/', 'accueil');

    $r->addRoute(
        'GET',
        '/equipe/{slug}',
        'equipe'
    );
});
```

```
<?php // 06-fastroute/index.php (suite)

$httpMethod = $_SERVER["REQUEST_METHOD"];
$uri = $_SERVER["REQUEST_URI"];

// nettoyage de $uri
// - prefix
// - query string
// - caractères spéciaux (e.g. %20)

$routeInfo = $dispatcher->dispatch(
    $httpMethod,
```



```
    $uri
);

<?php // 06-fastroute (suite)

switch($routeInfo[0]) {
    case Dispatcher::NOT_FOUND:
    case Dispatcher::METHOD_NOT_ALLOWED:
        /* ... */break;
    case Dispatcher::FOUND:
        try {
            echo call_user_func_array(
                $routeInfo[1],
                $routeInfo[2]
            );
        } catch (Exception $e){
            echo server_error($e);
        }
        break;
}
```

### Framework PHP

Une collection de bibliothèques avec un peu de glue.

Un framework web vous propose une structure de base pour construire selon une méthode jugée bonne par ses concepteurs. Il est possible de remplacer un composant par un autre, par le sien. Et même de créer sa *glue* ou même ses outils propres.

### Liens avec Laravel

- Modèle MVC
- Templates utilisant *blade*.
- ORM nommé *Eloquent*.
- *Front-Controller* (Illuminate\Routing)
- Bibliothèques ... (Illuminate\\*)
- [Composer](#)

Je vous invite à aller lire le code généré pour vous par Laravel. Vous allez retrouver ces éléments. Symfony, CakePHP, etc. auront les mêmes idées.

### Exercice

- Refaites les différentes étapes à partir de 00-base.
- Tel quel ou en utilisant d'autres bibliothèques : [Smarty](#), [Doctrine](#), [Aura.Router](#)

## Fin

Questions ?

## Sources

1. W3C. W3C 20 Anniversary Symposium. [en ligne]. 2014. [Consulté le 7 février 2017]. Disponible à l'adresse : <https://www.w3.org/20/Overview.html>
2. MUNROE, Randall. Exploits of a mom. [en ligne]. 2007. [Consulté le 7 février 2017]. Disponible à l'adresse : <https://xkcd.com/327/>
3. WIKIPEDIA. *Mapping objet-relationnel* [en ligne]. [Consulté le 7 février 2017]. Disponible à l'adresse : [https://fr.wikipedia.org/wiki/Mapping\\_objet-relationnel](https://fr.wikipedia.org/wiki/Mapping_objet-relationnel)
4. WIKIPEDIA. Modèle-Vue-Contrôleur. [en ligne]. [Consulté le 7 février 2017]. Disponible à l'adresse : <https://fr.wikipedia.org/wiki/Modèle-vue-contrôleur>
5. DJANGO PROJECT. Django appears to be a MVC framework, but you call the Controller the « view », and the View the « template ». How come you don't use the standard names? *FAQ : General* [en ligne]. [Consulté le 7 février 2017]. Disponible à l'adresse : <https://docs.djangoproject.com/en/1.11/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names>

## 3 Laravel

### Pourquoi **Laravel** ?

- Framework full stack / glue
- Prise en main rapide
- Bonne documentation, grande [communauté](#)
- Incite au respect des principes [S.O.L.I.D](#)
- Gratuit et opensource (Licence MIT)

### Historique

- Projet initié en 2011 par [Taylor Otwell](#)
- Basé sur des composants d'autres frameworks
- Mai 2013 : version 4, utilise [composer](#)
- Août 2014 : projet PHP le plus [populaire](#) sur github
- [Qui](#) utilise Laravel ?
- version 5.7 sortie en août 2018

### Principales fonctionnalités

- Routes RESTful
- ORM (Eloquent, implémentation du pattern Active Record)
- Migrations
- Moteur de templates (Blade)
- Pagination
- Authentification, sessions
- Mail
- Tests unitaires
- Extensible par [packages](#) (bundles) via composer

### Le Front Controller

### Architecture



FIG. 3.1 : Logo Laravel

## MVC

- Structure d'une appli web = [cycle Requête/Reponse](#)
- Modèle : Eloquent ORM
- Vue : Blade Engine
- Contrôleur : hérite de BaseController

## Pratique

- Conventions de codage : Laravel respecte [PSR-2](#)
  - Vous aussi avec [StyleCI](#)
- Editeurs et IDE : PhpStorm, [thimble](#), brackets, Sublime Text, Atom, VS Code...
- Tests : unitaires, Jmeter, Selenium, ...
- Outils : devtools Chrome ou FF, [Emmet](#), git
- Doc
  - [Documentation officielle](#) de Laravel
  - [Cheat Sheet](#)
- Tutoriels
  - [Best Momo](#), [Open Classroom](#), [CodeSchool](#) -> [Pluralsight](#)

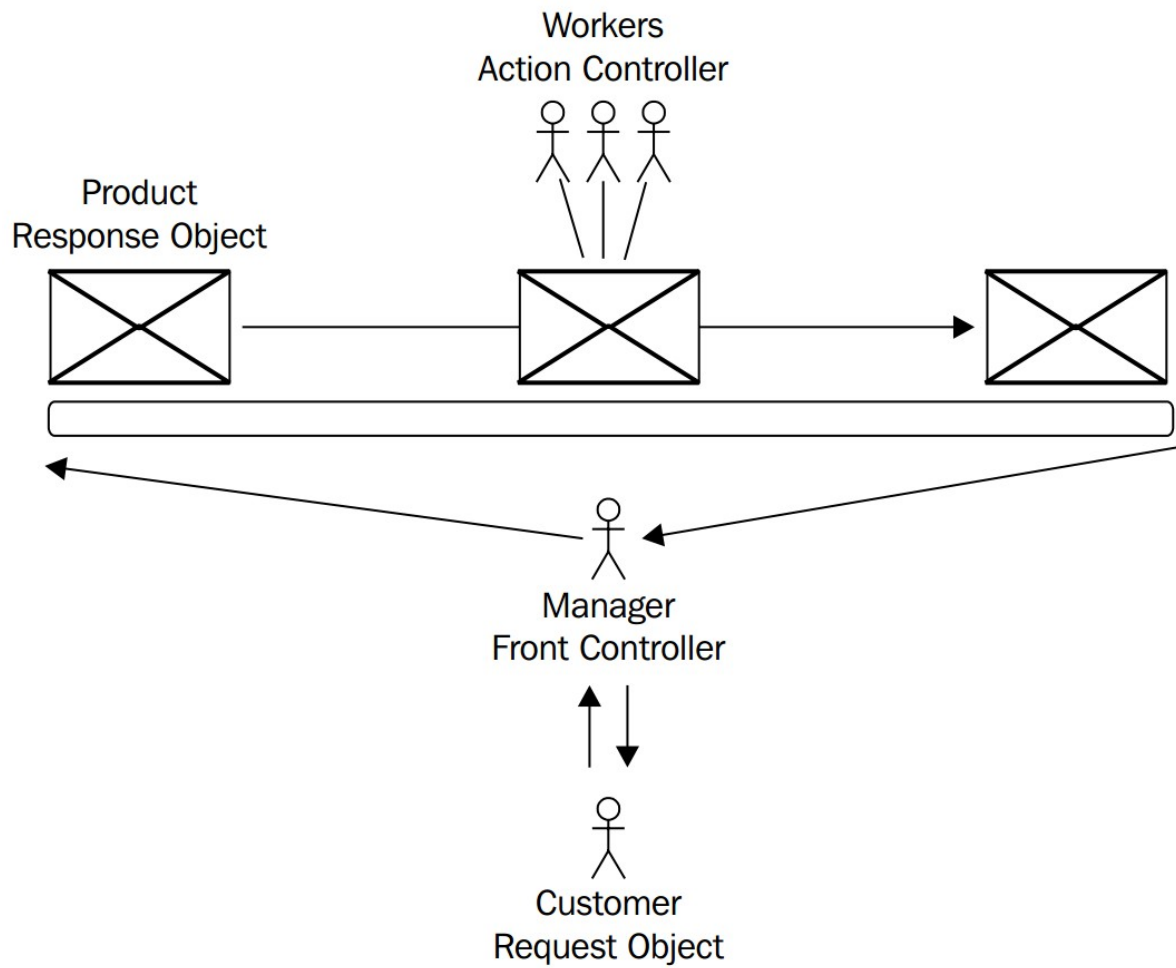


FIG. 3.2 : Rôle du front controller

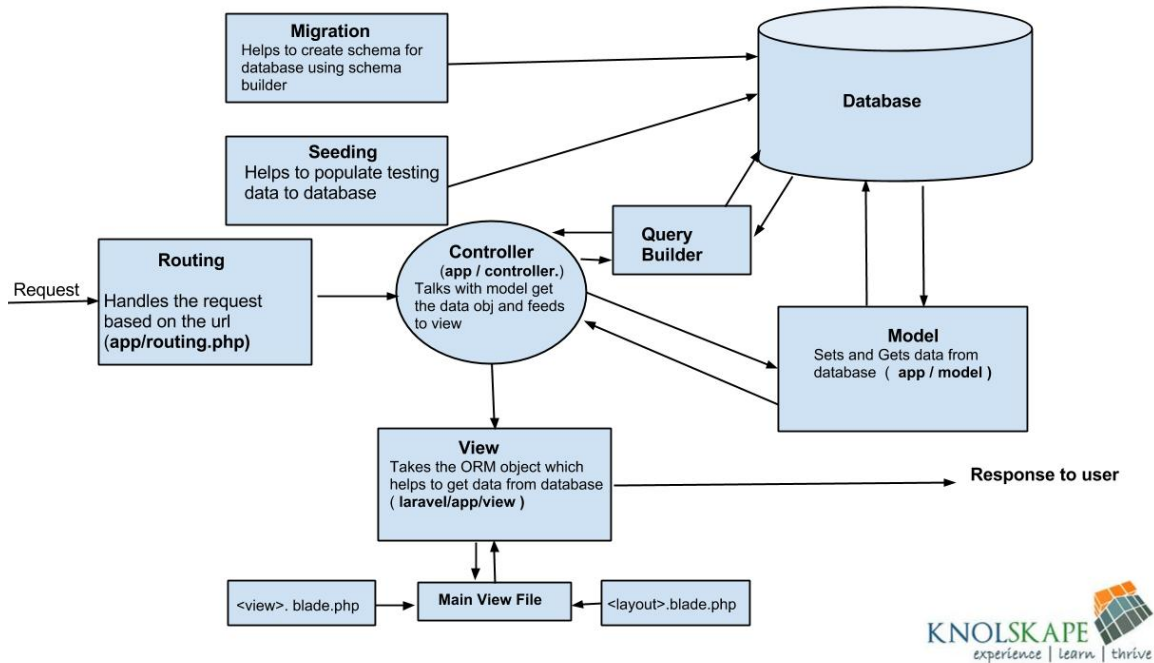


FIG. 3.3 : Architecture de Laravel

## Environnement de développement

- Local
  - Installation AMP, git + configuration : Long
  - Dépendant du poste de travail
  - Travail offline
  - Windows : [WSL](#) est votre ami !!!
- VM (Vagrant - [Homestead](#)) ou conteneur
  - Mise en route plus rapide : pré-configuré
  - Environnement dédié au dev, identique pour chaque développeur
- Cloud (Cloud9, ...)
  - Mise en route plus rapide : pré-configuré
  - Indépendant du poste de travail (navigateur)
  - Outils de synchro disponibles

## Environnement de développement

- Cloud : [Cloud9](#)
- Local ou VM

- Installer : serveur http, SGBD, git, php7, composer
- Installer Laravel :

```
$composer global require "laravel/installer"
```

## Démarrer un projet

- Créer un nouveau projet

```
$ composer create-project laravel/laravel raidit
# ou si ~/.composer/vendor/bin est dans le PATH :
$ laravel new raidit
$ cd raidit
```

- Racine du site dans /public (lien symbolique ou virtual host)

## Le dépôt

- Initialiser le dépôt

```
$cd raidit
$git init
$git add .
$git commit -m "Install laravel"
$git remote add origin git@github.com:bastian/raidit.git
$git push --set-upstream origin master
```

- Penser à ajouter sa clé publique à Github

## Apache

- Virtual hosts
  - http-vhosts.conf (activer dans httpd.conf)
  - Un par site
  - Pointer dans /public
- AllowOverride : active .htaccess
- .htaccess : redirection des requêtes
- Alternative : Remplacer le dossier racine http par un lien symbolique vers le dossier /public

## Artisan

- Laravel's CLI
- Construit avec Symfony Console
- Aide aux tâches courantes, ex :

```
$php artisan route:list
```

```
$php artisan migrate
```

```
$php artisan make:controller
```

```
$php artisan list
```

- [Extensible](#)

## Premiers pas

- [Routes](#)
  - Ajouter une route `/test`
  - Ajouter un paramètre qui sera affiché : `/test/param`
  - Utiliser une vue pour cette route
  - Lister les routes avec la commande `artisan`

...

- [Contrôleurs](#)
  - Ajouter un contrôleur : `Test`
  - Lui ajouter une action : `index`
  - Ajouter la route correspondante : `/test/index`

...

- [Vues](#)
  - Ajouter une vue Blade (`.blade.php`)
  - Afficher cette vue dans l'action `index`

## Ressources

- [Laracast](#)
- [Laravel Tips](#)
- [Learning Laravel](#)
- [RESTful API with Laravel 5](#)
- [Les vôtres](#)



## Sources

# 4 HTML 5

## Exemples

- Vue d'ensemble : [slides](#) Google 2011 ( [sources](#) )
- API d'accès à la [caméra](#)
  - ... saupoudré de [webGL](#)...
- Bachelor NIFFF 2014 : une webapp mobile pour LACIS
- Plein d'exemples
  - [html5 rocks !](#) => [Web Fundamentals](#)
  - [Chrome Experiments](#)
  - [MDN](#)
  - [html5 demos](#)
  - [plus de demos ?](#)
- Veille : [Frontend Focus](#) (newsletter)

## Progressive Web Apps

- Priorité à l'UX
- Utilise moins d'espace qu'une app native
- Avantages des 2 mondes (natif et web)
- [Article](#) d'Alex Russel 15.06.15
- Vue d'ensemble par [Wikipedia](#)
- Partiellement supporté par [iOS](#) (support SW en dev)

## PWA : howto

- Portabilité : [Progressive Enhancement](#)
- Rapidité : [App Shell](#), cache
- Offline : [Service Workers](#)
- [Install Banner](#) : HTTPS, WebApp Manifest, SW, 2 visits
- [Tests](#) : Lighthouse en automatise une partie

## Exemples et tutos

- Exemples
  - [PWA rocks](#)
  - [HN PWA](#)
  - [Gokulakrishnan Kalaikovan](#)
- Tutos
  - [Getting started with PWA](#)
  - [Your 1st PWA](#)
- [Awesome PWA](#)

## Sources

# 5 JavaScript et DOM

## JavaScript hier

- Page web = HTML (+ CSS + JavaScript)
- Exécuté par le browser (client)
- Interprété, faiblement typé, OO
- Historiquement
  - Depuis Netscape 2 (1995, Brendan Eich)
  - Petites applications exécutées par le navigateur
  - DHTML : rollovers, validation de formulaires, ...

## JavaScript aujourd'hui

- Page web = HTML + CSS + JavaScript
- Compilation JIT
- HTML5, AJAX, bookmarklets
- One Page Apps
- Implémentations hors-browser
  - Node.js, Spidermonkey, Rhino
  - script d'app (Qt, Notepad++, ...)
- Langage cible de compilateurs : [emscripten](#), [WebAssembly](#)
- Embarqué : [Espruino](#)

## \*Script

- ECMAScript : Norme depuis 1997
  - Juin 2017 : [ECMA-262 8th edition / 2017](#)
  - [Support](#) des différentes implémentations
  - Conversions avec [BabelJS](#)
- JavaScript : implémentation Firefox (réf. MDN)
- Variantes (à transpiler) :

- [Typescript](#) : variante fortement typée, avec des classes (MS)
- [Coffescript](#)
  - \* sucre syntaxique
  - \* compilé -> js

## JavaScript

- Différentes [implémentations](#) : navigateur, srv, apps, ...
- Permissif : du mauvais code est peu maintenable
  - [Design Patterns](#)
  - [Bonnes pratiques](#)
- Interface pour scripter le navigateur
  - Accès et modification du contenu via DOM
  - [Bookmarklets](#), [exemples](#)
  - Requêtes HTTP (Xml Http Request)
- Développement d'applications complètes, parfois offline
- Langage de script généraliste (paquets npm)

## Caractéristiques du langage

- Orienté Objet par prototype
- Syntaxe proche de C, Java
- Faiblement typé :
  - Pas de déclaration, type déterminé par la dernière affectation
  - Risque : typo => nouvelle variable. Utiliser var
- Types :
  - Primitifs : Boolean Null Undefined Number String Symbol
  - Objets : Object Function
- Particularités
  - [Prototypes](#)
  - [Fermetures](#)
  - [Promesses](#) ([MDN](#), [Google](#))

## Fonctions

- Pas de type de retour
- Possibilité de retourner ou non une valeur

- Sans retour, valeur spéciale : undefined
- Pas de surcharge (la dernière définie prime)
- fonction est un type
- Fonctions imbriquées, anonymes
- Fonctions globales :

`escape()`, `unescape()`, `isFinite()`, `isNaN()`,  
`parseFloat()`, `parseInt()`, `Number()`, `String()`,  
`eval()`, ...

## JavaScript dans la page web

- Éléments `<script>` exécutés dans l'ordre de la page
- Conseillé de les placer en [fin de page](#)
- Événements (onclick, onerror, onsubmit, ...)
  - Embarqués dans les balises (onXXX)

```
<div id="intro" onclick="change();" />
```

Utiliser DOM

```
<script type="text/javascript">
```

```
document.getElementById("intro").onclick = change;
```

```
</script>
```

- Conseillé d'inclure le code (attribut src)

```
<script type="text/javascript" src="script02.js"></script>
```

language="JavaScript" est déprécié et type vaut par défaut text/javascript.

The type attribute gives the language of the script or format of the data. [...] The default, which is used if the attribute is absent, is ``text/javascript`.

[HTML5 : script](#)

## Unobstrusive JS

- Séparation JS...

```
document.addEventListener("DOMContentLoaded", function() {  
    document.getElementById('date').addEventListener("change", validateDate);  
});
```

- ...et HTML

```
<input type="text" name="date" id="date" />
```

- Dégradation élégante
  - Alternatives pour un browser ne supportant pas JS
- Accessibilité
  - Les fonctionnalités restent accessibles en cas d'erreur
- Utilisabilité
  - Le script doit faire gagner du temps, pas distraire

It is an incredibly popular mistake to use `load` where `DOMContentLoaded` would be much more appropriate, so be cautious.

[MDN : DOMContentLoaded](#)

## Node.js

- Node.js : une implémentation hors navigateur
  - environnement d'exécution + bibliothèques
  - event driven, non-blocking IO -> scalable
  - V8 engine
  - scripts exécutables sans navigateur
  - [npm](#) : gestionnaire de paquets
  - gulp : make js
- [Exemples](#) d'applications
  - gulp, grunt, bower, yarn
  - browserify
  - serveur http
  - express, cordova, forever, dev, pm2, karma, sails
- [Tuto](#), [Playground](#)

## DOM

- Document Object Model
- Représentation arborescente de la page
- Accessible depuis objet JS document
- Possibilité d'accéder au contenu de la page :
  - Lecture
  - Modification
  - Ajout
- JS peut donc modifier le contenu d'une page

## DOM

```
<html>
<head>
  <title>My title</title>
</head>
<body>
  <h1>A heading</h1>
  <a href="#">Link text</a>
</body>
</html>
```

## L'objet Document

- Trouver ou modifier des éléments
- Méthodes de Document

```
getElementById(), getElementsByTagName(), getElementByClass(),
createElement(), createTextNode()
```

- Méthodes de Node (appel depuis nœud parent)

```
insertBefore(child), appendChild(child),
removeChild(child), replaceChild(new,old)
```



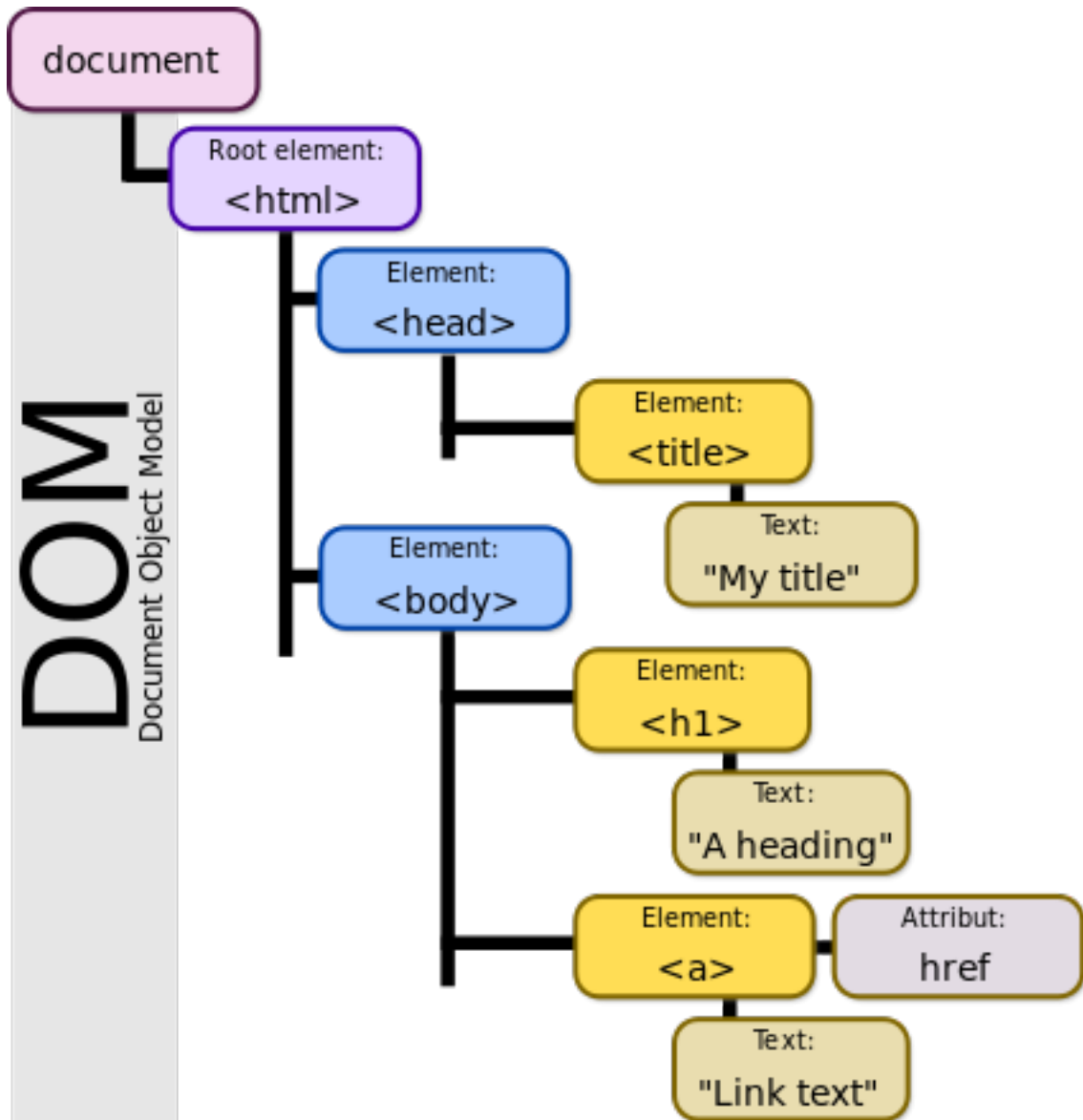


FIG. 5.1 : DOM tree

## Ajouter un noeud

```
function addNode() {
    var inText = document.getElementById("textArea").value;
    var newText = document.createTextNode(inText);

    var newGraf = document.createElement("p");
    newGraf.appendChild(newText);

    var docBody = document.getElementsByTagName("body")[0];
    docBody.appendChild(newGraf);
}
```

- Création du nouveau nœud :
  - newText contient le texte à ajouter
  - newGraf est un élément p qui contient le texte
- Ajout du nœud comme une feuille de body :
  - Sélection du parent (le premier noeud body)
  - Ajout du nouveau nœud depuis son parent

## Supprimer un nœud

```
function delNode() {
    var allGraf = document.getElementsByTagName("p");

    if (allGraf.length > 1) {
        var lastGraf = allGraf.item (allGraf.length-1);
        lastGraf.parentNode.removeChild(lastGraf);
    }
    else {
        console.error("Nothing to remove!");
    }
}
```

- Sélection du nœud à supprimer :
  - allGraf contient tous les éléments p
  - lastGraf contient le dernier du tableau allGraf
- Suppression :
  - Suppression du nœud sélectionné depuis son **parent**

## Insérer un nœud

```
function insertNode() {
    var newText = document.createTextNode("New Text");
    var newGraf = document.createElement("p");
    newGraf.appendChild(newText);

    var divMod = document.getElementsByTagName("div")[0];
    var allGraf = divMod.getElementsByTagName("p");
    var oldGraf = allGraf.item(0);           // position

    divMod.insertBefore(newGraf,oldGraf);
}
```

- Création du nouveau nœud :
  - allGraf contient tous les éléments p
  - lastGraf contient le dernier du tableau allGraf
- Insertion :
  - Recherche du parent
  - Recherche du frère gauche
  - Insertion depuis le parent

## Avec jQuery

- Création et ajout :

```
var noeud = $('<p>Nouveau texte</p>'); // create node
$("body").append(noeud);                 // après le dernier fils
```

- Sélection et Suppression :

```
var noeud = $("p"); // select node(s)
noeud.remove();
```

## Références

- Une [réintroduction à JavaScript](#)
- [How does it feel to learn JS in 2016](#)
- Référence [MDN](#)
- Tutoriels [w3schools](#)
- Outils de développement Chrome et Firefox (Ctrl+Shift I)
- Firefox :
  - [Tilt3D](#) (Ctrl+Shift+L)
  - Barre développement (Shift+F2)
- Outils web
  - [JSFiddle](#)
  - [JSLint](#)

## Sources

# 6 HTTP et AJAX

## HyperText Transfer Protocol

- Protocole application : invention www en 1990 (v0.9)
  - Connexion, GET, réponse, fermeture
- HTTP 1.0 (1996)
  - Entêtes de requête (Host, Referer, User-Agent, ...)
  - Entêtes de réponse (Content-Type, Set-Cookie, Location, ...)
- HTTP 1.1 (1997)
  - Keep-alive, pipelining, cache, ...
  - Plus d'entêtes, Host obligatoire
- [HTTP 2.0](#) (2015)
  - Binaire, multiplexage connexions, compressions entêtes, push, ...
  - Supporté par [presque tous](#) les navigateurs, une majorité de serveurs

## Codes de réponse

- 1xx : Information
- 2xx : Succès
- 3xx : Redirection
- 4xx : Erreur Client
- 5xx : Erreur Serveur

## Méthodes HTTP (verbes)

- GET : Demander une ressource
- POST : Création d'une ressource
- PUT : Remplacement total d'une ressource
- PATCH : Remplacement partiel d'une ressource
- DELETE : Suppression d'une ressource
- HEAD : Demande l'entête de la réponse, sans la ressource

- TRACE, OPTIONS, CONNECT

idempotentes sûres

## Echanges HTTP

- Requête

```
GET / HTTP/1.1[CRLF]
Host: www.cff.ch[CRLF]
Connection: close[CRLF]
User-Agent: Opera/9.20 (Windows NT 6.0; U; en)[CRLF]
Accept-Encoding: gzip[CRLF]
Accept-Charset: ISO-8859-1,UTF-8;q=0.7,*;q=0.7[CRLF]
Cache-Control: no[CRLF]
Accept-Language: de,en;q=0.7,en-us;q=0.3[CRLF]
Referer: http://web-sniffer.net/[CRLF]
[CRLF]
```

- Réponse

```
HTTP Status Code: HTTP/1.1 302 Found
Date: Mon, 16 Nov 2009 08:01:35 GMT
Server: Apache
Location: http://www.sbb.ch/fr/
Content-Length: 205
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head><title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www.sbb.ch/fr/">here</a>.</p>
</body></html>
```

## HTTP

- Requête POST : paramètres dans le corps

```
POST /login.jsp HTTP/1.1
Host: www.mysite.com
User-Agent: Mozilla/4.0
Content-Length: 27
Content-Type: application/x-www-form-urlencoded
```

```
userid=joe&password=guessme
```

- Outils HTTP
  - CLI : curl
  - Browser dev tools
  - WebApp : [HURL](#)
- Exemples PATCH : [mnot](#) , [SOA bits](#)

## AJAX : Historique

- Asynchronous Javascript And Xml
- Buzzword, [Jesse James Garret](#), 2005
- Mise à jour sans rechargement intégral
- Utilisation de [Remote Scripting](#) et de DOM
- Historique de techniques de remote scripting
  - [\(i\)frames](#)
  - Bibliothèques JS (ex : [JSRS](#))
  - Utilisation des images/cookies (ex : [GIF](#))
  - Applets, Flash, ActiveX, ...
  - XHR : XML HTTP Request (IE5, 1999 pour OWA)
  - Fetch API
- Pas obligatoire d'avoir du JS, XML ni d'être asynchrone !

## AJAX

- XHR est devenue la méthode standard
  - Popularisée par Google (GMaps, GMail, ...)
  - Le w3c fait évoluer un [draft](#) depuis 2006

- Principe
  1. Envoi de requête HTTP
  2. La réponse provoque l'exécution de la fonction de rappel
  3. Le DOM de la page est mis à jour
- Applications
  - GUI ressemblant à des app natives
  - MAJ dynamiques de formulaires, autocompletion
  - Validation avec interrogation du serveur
  - ...

## L'objet *XMLHttpRequest*

- Initiative de Microsoft
  - Composant ActiveX de IE5
  - Adopté par Mozilla 1.0 et Safari 1.2
  - Standardisation W3C en cours
- Requête HTTP en JS
- Fonction de rappel (callback)
- Asynchrone : Non bloquant
- Non standard => différentes implémentations
- Supporté par Chrome, FF, Safari, IE, Konqueror, ...
- Alternative souhaitable si JS désactivé

## XHR en JS

```
var xhr;
function createXMLHttpRequest()
{
    if (window.ActiveXObject)
    {
        xhr = new ActiveXObject("Microsoft.XMLHTTP");
    }
    else if (window.XMLHttpRequest)
    {
        xhr = new XMLHttpRequest();
    }
}
```

- Dans son [contexte](#)



## XHR en jQuery avec *load()*

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#div1").load("demo_test.txt");
    });
});
</script>
</head>

<body>
    <div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>
    <button>Get External Content</button>
</body>
</html>
```

- [Tester](#)
- D'[autres](#) façons de faire

## XHR : propriétés et méthodes

- readyState, status, onreadystatechange
- .responseText, responseXML
- open (Verbe, URI, async) :
  - Verbe HTTP : ``GET``, ``POST`` ou ``PUT``
  - URI : destinataire de la requête
  - async (bool) : true = asynchrone, false = bloquant
- send (null | string) : peut être bloquante
- setRequestHeader(header, value)
- getResponseHeader(string)
- abort()

## Envoi de données

- GET
  - Obtenir des données

- Longueur URL limitée par le navigateur (2'048 pour IE)
- Utilise le cache (navigateur, proxy)
- manipulables par l'utilisateur (bookmarks, partage, ...)
- POST
  - Faire quelque chose
  - Données sensibles
  - Longueur limitée par le serveur (assez large)
  - Utilisation de la méthode `send()` de XHR
  - Requête Ajax en 2 temps (entête, puis données)
- Cache
  - Client : Construire des [URL uniques](#)
  - Serveur : Envoi d'[entêtes](#) interdisant le cache

```
MyXhr.open("GET", "fichier.xml", true);
MyXhr.setRequestHeader("Cache-Control", "no-store, no-cache, must-revalidate,
    post-check=0, pre-check=0");
MyXhr.setRequestHeader("Pragma", "no-cache");
MyXhr.setRequestHeader("Expires", "Wed, 09 Aug 2000 08:21:57 GMT");
```

## Préférer GET, sauf

[Détails](#)

## Réponse en texte

- Si la requête aboutit :
  - `readystate == 4`
  - `status == 200`
- La réponse est dans l'attribut `responseText`
- ou dans `responseXML`
  - Utilisation du DOM (`getElementsByTagName()`, ...)

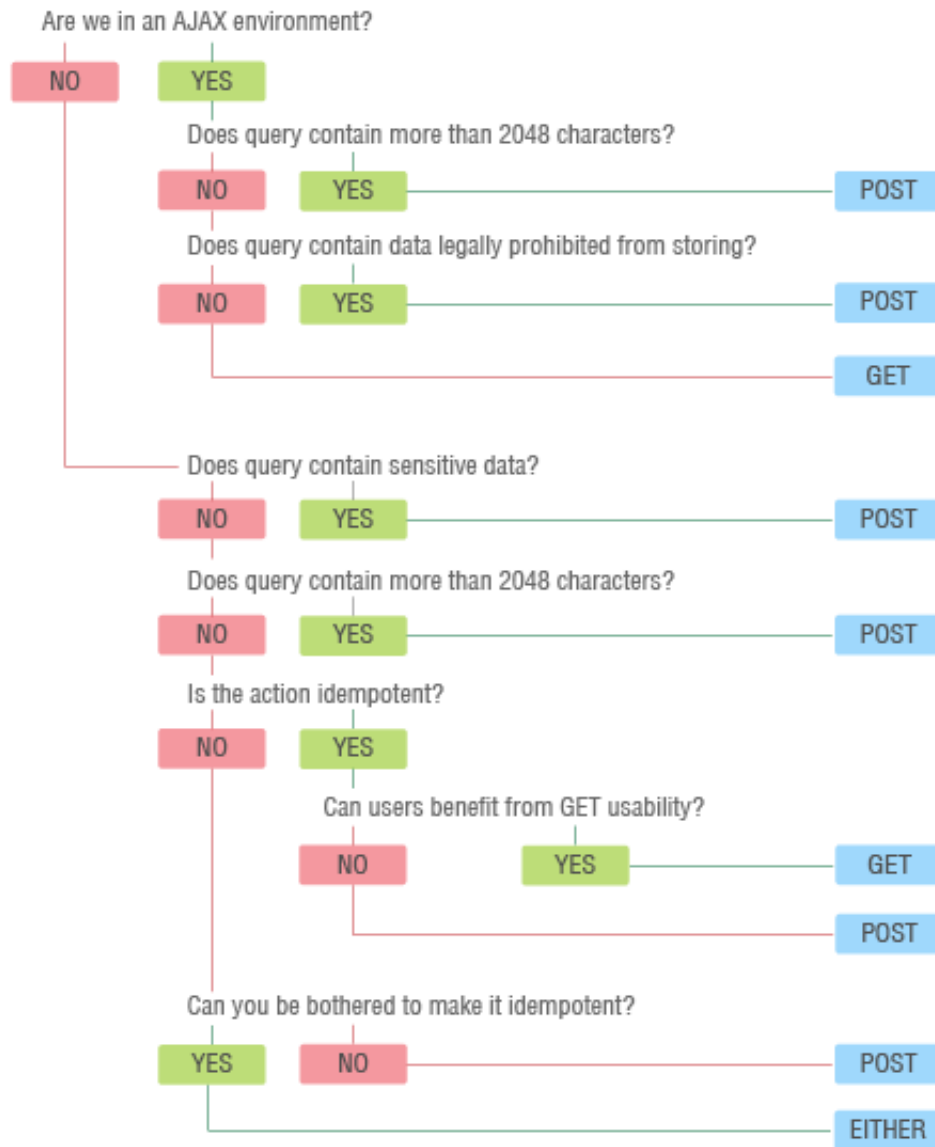


FIG. 6.1 : ``GETorPOST''

## Réponse en XML

```
<?xml version="1.0" ?>
<liste>
  <personne>
    <nom>Berger</nom>
    <prenom>Laurent</prenom>
  </personne>
  <personne>
    <nom>Borgo</nom>
    <prenom>Sébastien</prenom>
  </personne>
  <personne>
    <nom>Bux</nom>
    <prenom>Rémy</prenom>
  </personne>
</liste>
```

- Dans responseXML

## Réponse en JSON

- [Standard](#) depuis octobre 2013 ([Douglas Crockford](#))
- Tableau d'objets js :
  - pour chacun, ses attributs sont des paires clé : valeur

```
{objet1 : nom : 'Berger', prenom: 'Laurent'}
```

```
[objet1, objet2, objet3]
```

```
[
  {nom:"Berger", prenom:"Laurent"},
  {nom:"Borgo", prenom:"Sébastien"},
  {nom:"Bux", prenom:"Rémy"}
]
```

- Utilisation de : `~~var users = eval('(' + myXHR.responseText + ')');` `~~` pour créer le tableau d'objets correspondant

## « eval is Evil »

- `eval()` : évalue et exécute la chaîne en paramètre
- Risque : instructions au lieu d'un tableau d'objets
- Solution : le [parser](#) JSON

```
var users = JSON.parse(myXHR.responseText);
var myString = JSON.stringify(users);
```

- Avec jQuery :

```
var obj = jQuery.parseJSON('{ "nom": "Berger" }');
alert(obj.nom);
```

## Fetch API

- Le successeur d'XHR est [fetch](#) : [Exemple](#)
- Fetch a un *polyfill* pour les navigateurs ne le supportant pas
- L'API Fetch est native et plus simple d'utilisation que jQuery

```
fetch("fichier.json")
  .then(function(response) {
    return response.json()
  })
  .then(function(json) {
    console.log(json);
  })
  .catch(function(error) {
    console.error("erreur", error)
  })
```

- L'API fetch est native et utilise les [promesses](#) plutôt que les callbacks

## Traitement d'erreurs

- Utiliser les [entêtes HTTP](#)
  - Champ Status
  - Code d'erreur
- En PHP

```
header("Status: Message d'erreur explicite", true, 400);
```

- Afficher le message au client :

```
myXHR.getResponseHeader("Status");
```

## Penser à l'utilisateur !

- Requêtes XHR non enregistrées dans l'historique :
  - Bouton précédent non opérationnel (sauf GET et URL uniques)
  - Pas de bookmark
  - solution via [History API](#)
- Utilisabilité : signaler à l'utilisateur ce qui est en cours :
  - GIF [AJAX loading](#)
  - Rectangle Loading en haut à droite (Google)
  - [Yellow Fade Technique](#) (37signals) : partie modifiée
- Code client :
  - Pas de maîtrise performance
  - Mauvais code == Appli lente
- En cas de doute, faire tester des utilisateurs

## Bonnes pratiques d'utilisabilité

- Trafic minimal
- Pas de surprise
- Respect des conventions
- Pas de distraction
- Accessibilité ([ARIA](#))
- Ne pas switcher AJAX/non-AJAX
- Se mettre à la place de l'utilisateur

## Sources

# 7 jQuery

## jQuery

- John Resig, 2006
- Bibliothèque JS, gratuit, OS (licence MIT)
- Facilite le développement JS pour les tâches fréquentes :
  - Manipulations DOM
  - Manipulations CSS
  - Réponse aux événements du navigateur
  - Effets visuels et animations
  - Requêtes et réponses Ajax
- Abstraction implémentations différents navigateurs
- Facile à apprendre
- Utilisation du chaînage des méthodes et des callbacks

## Utilisation

- Inclusion [CDN](#)

```
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
```

- Nos scripts

```
<script src="application.js"></script>
```

- Syntaxe basique

```
$(selecteur).action(); // $() est un raccourci pour jQuery()
```

- Utilisation de sélecteurs CSS, id ou classes

```
$(document); // retourne le DOM
$("h3").hide(); // cache tous les éléments h3
$(".post"); // sélectionne les éléments de classe "post"
var node = $('<p>New</p>'); // un nouveau noeud
```

- Pour être sûr que le document est chargé :

```
$(document).ready(function(){
    console.log("prêt!");
});
```

ou

```
$(function() {
    console.log("prêt!");
});
```

## Sélection dans le DOM

- Sélection

```
$("h1"); // noeud élément
$("h1").text(); // noeud texte en lecture
```

- Modification

```
$("h1").text("Nouveau Texte"); // noeud texte modifié
```

- Tous les fils (sélecteur descendant)

```
$("#intro li");
```

- Que les fils directs (sélecteur d'enfants)

```
$("#intro > li");
```

- Sélecteur multiple

```
$(".post, #main ");
```

- D'autres [exemples](#) de sélecteurs



## Parcours (**traversing**)

- Parcours du DOM dans les trois directions :

- Depuis le noeud courant (sélectionné)
- Haut : `parent()`, `parents()`
- Bas : `children()`, `find()`
- Frères : `sibling()`, `next()`, `prev()`

- Filtrage

- `first()`, `last()`, `eq()`
- `filter()`, `not()`
- [Référence](#)

## Modifications de contenu

- Accès au contenu :

- `text()` : get/set le texte entre les balises
- `html()` : get/set l'élément complet (yc balises)
- `val()` : get/set les valeurs d'un formulaire
- `attr()` : set la valeur d'un attribut

- Ajout de contenu :

- `append()`, `prepend()` : au début/fin de la sélection (dans l'élément)
- `before()`, `after()` : avant/après la sélection

- Suppression

- `empty()` : suppression des enfants
- `remove()` : suppression de la sélection (possibilité de filtrer)

## Accès aux CSS

- Accès aux classes

- `addClass()` : ajout de classe(s) à l'élément sélectionné
- `removeClass()` : suppression de classe(s)
- `toggleClass()` : suppression si présente, ajout sinon

- Attribut style d'un élément : `css()`

```
$("#p").css("background-color");           // get
$("#p").css({"background-color":"yellow","font-size":"200%"}); // set
```

## Evénements

- Souris
  - click, dblclick, mouseenter, mouseleave
- Clavier
  - keypress, keyup, keydown
- Formulaires
  - submit, change, focus, blur
- Document
  - ready, load, resize, scroll, unload
- Exemple

```
$("#p").click(function(){  
    // code à exécuter ici  
});
```

## AJAX

- \$(selector).load(URL, data, callback)
  - URL : Ressource ciblée par la requête
  - data : éventuel contenu
  - callback : fonction de rappel avec 3 paramètres :
    - \* responseTxt
    - \* statusTxt
    - \* xhr
- \$.get(URL, callback)
- \$.post(URL, data, callback)

## Effets et animations

- hide(), show(), toggle()
- fadeIn(), fadeOut(), fadeToggle()
- slideDown(), slideUp(), slideToggle()
- [animate\(\)](#)

## Alternatives

- *jQuery aussi, ça fait vieux*, YBL 17.10.29
- [bling.js](#)
- API [querySelectorall\(\)](#) au lieu des `getElementsBy...`

## Références

- Site officiel de [jQuery](#)
- Tutos [w3schools](#)
- Tutos [codeschools](#)
- [SizzleJS](#) : uniquement les sélecteurs
- Comparaison avec [Vanilla JS](#)

## Sources

# 8 Syndication (RSS)

## Syndication

- Principe de vendre un contenu à plusieurs médias
- Dans les journaux : dépêches, bandes dessinées, ...
- Télévision : jeux, séries
- Web : Flux RSS / Atom
  - 1 source de donnée, plusieurs abonnés
  - Contenu : news, blogs, podcast, ...
  - Accès unique à plusieurs sources d'informations
  - Mises à jour fréquentes

## Historique

- Feed (fil ou flux) RSS
- Format d'échange de données en XML
  - fournir ou recueillir des données structurées
- Utilisation d'un lecteur (agrégateur) RSS
- RSS V.90 Créé en 1999 par Netscape
- RSS v1.0 par O'Reilly en 2000
- RSS v2.0 par Dave Winer (Harvard) en 2002
- Atom v1.0 en 2005 (développement communautaire)

Il y a *neuf* versions de RSS généralement incompatibles entre elles. Lire [The myth of RSS compatibility](#)

## Applications

- Récupérer l'info pour :
  - la lire
  - la réutiliser sur un site
- News

- Notification : activité, mise à jour
- Podcasts
- Accès unique à des infos de plusieurs sites
- Source de contenu
- Augmenter le trafic d'un site
- [Exemples](#) et [Passerelles](#)

## Agrégateurs

- Natifs
  - Navigateurs (IE, FF, ...)
  - Clients mail (OL, TB, Evolution, ...)
  - Applis dédiées (Newsgator, FeedDemon, ...)
- WebApps
  - Feedly, NetVibes, Sniptracker...
- Extensions
  - Sage
- [Liste](#)

## Générer un flux RSS

- Fichier XML :
  - Canal / Items (RSS)
  - Entrées (Atom)
- Indiquer le flux au navigateur
- Permettre l'abonnement : logo visible dans la page
- Génération dynamique du fichier XML

## Formats

- RSS 2.0 (Really Simple Syndication)
  - Simple, le plus répandu
  - Extensible par modules (éléments supplémentaires)
- Atom 1.0 : 2 standards web
  - Atom Syndication Format
  - Atom Publishing Protocol

- RSS 0.90, 1.0 (RDF Site Summary) : obsolète
  - Basé sur RDF
  - Extensible par modules
- Antérieurs : RSS 0.91, 0.92 (Rich Site Summary) : obsolètes
  - Migration facile vers RSS 2.0

## RSS 2.0

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0">
  <channel>

    <title>Arc Info News RSS 2.0</title>
    <link>http://www.he-arc.ch/</link>
    <description>News HE-Arc (RSS 2.0)</description>

    <language>fr</language>
    <pubDate>Sun, 26 Oct 2008 04:00:00 GMT</pubDate>
    <lastBuildDate>Sun, 26 Oct 2008 09:41:01 GMT</lastBuildDate>
    <docs>http://blogs.law.harvard.edu/tech/rss</docs>
    <managingEditor>david.grunenwald@he-arc.ch</managingEditor>
    <webMaster>david.grunenwald@he-arc.ch</webMaster>
    <ttl>5</ttl>

    <item>
      <title>Nouveau cours d'Applications Internet 2</title>
      <link>https://intranet.he-arc.ch/sites/ingenierie/
        Bachelor_Modules_Annees_Fich/12-13/Niveau-3/
        ING-DM3254-12-D%C3%A9veloppement%20web%20et%20mobile-V1.docx</link>
      <description>Un nouveau cours</description>
      <pubDate>Mon, 27 Oct 2008 09:39:21 GMT</pubDate>
    </item>

  </channel>
</rss>
```

## Atom 1.0

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
```

```
<title>Arc Info News Atom 1.0</title>
<subtitle>version Atom</subtitle>
<link rel="self" type="application/atom+xml"
      href="http://www.he-arc.ch/rss-generator/atom.php" />

<updated>2008-10-27T18:30:02Z</updated>
<author>
  <name>David Grunenwald</name>
  <email>david.grunenwald@he-arc.ch</email>
</author>
<id>http://dgr.he-arc.ch/</id>

<entry>
  <title>Nouveau cours d'Applications Internet 2</title>
  <link>https://intranet.he-arc.ch/sites/ingenierie/
    Bachelor_Modules_Annees_Fich/12-13/Niveau-3/
    ING-DM3254-12-D%C3%A9veloppement%20web%20et%20mobile-V1.docx</link>
  <id>http://dgr.he-arc.ch/atom/1234</id>
  <updated>2008-10-27T18:30:02Z</updated>
  <summary>Un tout nouveau cours.</summary>
</entry>

</feed>
```

## Générer le flux

- Données dynamiques
- Source de données identique à celle de l'application
- Nécessité de générer le fichier à la volée
- Nouveaux items en premier
- Possibilité d'afficher le flux avec XSLT

## Signaler la présence d'un fil RSS

- Au navigateur

```
<link rel="alternate" type="application/rss+xml" title="RSS"
      href="http://www.site.tld/feedfilename.xml">
```

- À l'utilisateur

- Icône + lien vers le script générant le flux

```
<a href="http://www.site.tld/feed">  
</a>
```

- Valider un flux
  - [w3c](#)
  - [feedvalidator](#)
- MIME Types
  - application/atom+xml
  - application/rss+xml

## Podcasts

- Élément en RSS 2.0 :

```
<item>  
<title>Podcast</title>  
<link>http://www.website_url.com</link>  
<description>Podcast : audio.mp3</description>  
  
<enclosure url="http://www.site.tld/sounds/audio.mp3" length="666666" type="audio/mpeg"/>  
  
<guid isPermaLink="false">2004-11-30-02</guid>  
</item>
```

- Élément en Atom 1.0 :

```
<entry>  
<id>http://www.example.org/entries/1</id> <title>Atom 1.0</title>  
<updated>2005-07-15T12:00:00Z</updated>  
<link href="http://www.example.org/entries/1" />  
<summary>An overview of Atom 1.0</summary>  
  
<link rel="enclosure"  
type="audio/mpeg" title="Sttella - ça va comme un lundi"  
href=" http://www.site.tld/sounds/audio.mp3 "  
length="666666" />  
  
</entry>
```



## Alternatives

- [Facebook Open Graph](#)
- [Twitter Cards](#)
- [Google Schema.org](#)
- [Microformats](#)
- [JSON-LD](#)

De multiples spécifications permettent d'enrichir le contenu d'une page afin de la rendre aisément « consommable » par un moteur de recherche, ou une plateforme sociale (e.g. Facebook, Twitter, Reddit, etc.)

RDF/XML (utilisé par RSS 0.90, 0.91) est progressivement remplacé par les *microdata* (Schema.org), RDFa ou JSON-LD. Les microformats sont notamment utilisés par LinkedIn.

## Pour en savoir plus...

- [Étapes de création d'un flux](#)
- [Spécification RSS 2.0](#)
- [Spécification Atom 1.0](#)
- [Comparatif RSS 2.0 / Atom 1.0](#)
- [Stats d'utilisation](#)
- [Is RSS dead ?](#) (03.2015)

## Sources

# 9 Services Web

## Applications distribuées

- Motivation : répartir l'exécution sur plusieurs machines
  - Principe : Les composants/services communiquent par le réseau
  - Problèmes : Hétérogénéité systèmes, langages, ...
  - Solution : Protocole générique, abstraction différences
  - Exemples : RPC, RMI (java), CORBA, DCOM (MS)
- Utiliser les technologies du web, comme HTTP et XML :
  - indépendantes de la plateforme, éprouvées, largement utilisées
- Système distribué importance de l'architecture :
  - [orientée ressource](#) : atome : ressource (donnée) : REST
  - [orientée service](#) : atome : service (traitement) : RPC (SOAP)

## Service web

- 2 visions :
  - Utiliser les technos web pour développer des applis distribuées
  - Accès pour une application aux services offerts aux humains
- Service web = webapp pour une autre application :
  - Webapps : pour humains, via un navigateur (HTTP + HTML)
  - Services web : aux autres applications (HTTP + XML/JSON)
- Exemples :
  - [Applications distribuées](#) pour l'entreprise
  - [Mashups](#) d'applications web ([exemples](#))
  - Applications Facebook, [API Google](#)
  - [IFTTT](#), [potions Netvibes](#)
- Consommer un service web ≠ Créer un service web

## SOAP

- AVANT : Simple Object Access Protocol (obsolète)
- Evolution de XML-RPC, format XML d'envoi de messages
- Architecture Orientée Service (SOA)
- Indépendant du langage et de la plateforme
- Recommandation du w3c depuis 2003
- SOAP = abus de langage, service web WS-\* est plus exact
- Spécifications [WS-\\*](#) :
  - spécifications liées aux différents aspects des services web
  - pour déployer un WS : au minimum SOAP + WSDL + UDDI

## SOAP

- Structure d'un message SOAP
  - Enveloppe, Entête, Corps, Erreurs
- Squelette :

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Header> ... </soap:Header>
  <soap:Body> ...
    <soap:Fault> ... </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

## SOAP

- [Exemple](#) requête/réponse
- [Introduction à SOAP](#) (fr)
- Créer un service web WS (SOAP) nécessite WSDL et UDDI :
  - SOAP : Echange de messages XML sur le réseau
  - WSDL : Web Service Description Language
  - UDDI : Universal Description, Discovery and Integration
- WSDL : Description des interfaces des web services
- UDDI : Découverte et inscription aux services web

- annuaire d'informations sur les services web
- annuaire d'interfaces de services web décrites en WSDL
- [Tutorial WSDL/UDDI w3schools](#)

## REST : REpresentational State Transfer

- Style d'architecture sur lequel a été bâti le web
- Architecture Orientée Ressource (ROA)
- Chapitre 5 de la [thèse](#) de [Roy T. Fielding \(fr\)](#), 2000
- Parmi les [contraintes](#), une interface uniforme :
  - Identification des ressources (URI)
  - Manipulation des ressources par des représentations
  - Messages autodescriptifs
  - Hypermédia comme moteur de l'état de l'application
- Ressource : information ou moyen d'accès
  - ex. : météo du jour, adresse ajout d'un article à un blog, ...
- Représentation : forme donnée à la ressource
  - ex. : page html, fichier PDF, image, flux RSS, fichier sonore, ...

## REST

- Principes
  - Identifier les ressources avec des URI (noms)
  - Actions déterminées par des méthodes HTTP (verbes)
    - \* GET : READ (sûre)
    - \* POST : CREATE
    - \* PUT, PATCH : UPDATE (idempotente)
    - \* DELETE : DELETE (idempotente)
  - Les liens hypertextes permettent de représenter le contenu : navigation
  - Les types MIME déterminent la représentation de la ressource
- Rappel
  - Sûreté : Etat de la ressource (contenu) inchangé
  - Idempotence : plusieurs appels donnent le même résultat

## REST

- URI logique plutôt qu'URL physique
- L'appel d'une ressource avec des méthodes différentes produira un résultat différent :

```
* GET      http://www.monblog.com/posts    // Liste des billets
* GET      http://www.monblog.com/posts/1  // Billet 1
* POST     http://www.monblog.com/posts    // Création d'un billet
* PUT/PATCH http://www.monblog.com/posts/1 // Mise à jour billet 1
* DELETE   http://www.monblog.com/posts/1 // Suppr billet 1
```

- Avec [Laravel](#) ou Rails, ces actions sont nommées : index, show, store/create, update, destroy
- Laravel et Rails sont RESTful !

## Niveaux de maturité de [Richardson](#)

- 0 : Plain Old Xml (POX)
  - Utilisation de HTTP pour faire du RPC
- 1 : Ressources
  - Ressources identifiées par URI
- 2 : Verbes HTTP
  - Respect des propriétés des verbes HTTP
- 3 : Hypertext As The Engine Of Application State (HATEOAS)
  - Les états suivants sont documentés dans la réponse (<link>)

## SOAP vs REST

- webservice : exposer son API en REST ou SOAP ?
- SOAP (WS-\*)
  - hérité du monde de l'entreprise
  - plus de code pour manipuler la requête et générer la réponse
  - plus flexible, extensible (namespace)
  - valider requêtes depuis WDSL
  - nécessité d'un framework (ex : nuSOAP en PHP)
- REST

- hérité du web
- plus facile et rapide à utiliser
- plus lisible et plus compact
- maintenance plus facile
- meilleure tolérance aux pannes

## Pour aller plus loin...

- Références
  - [SOAP](#), [WSDL](#), [UDDI](#), [XML-RPC](#), [REST](#), [The WSIO](#)
  - [Des services web RESTful](#), [Une apologie de REST](#) (recommandés)
  - [REST et architectures orientées service](#), [Présentation ROA](#)
  - [The RESTful cookbook](#), [Implementing REST](#)
  - How important is [HATEOAS](#) ([stack overflow](#))
- Exemples de services web :
  - [Google](#), [Yahoo](#), [Flickr](#), [Twitter](#), [Netvibe](#), ...
  - [APIary](#) : Aide au design d'une API REST
- [GraphQL](#)
  - est destiné à devenir la prochaine évolution des apis REST utilisant JSON. Initié par Facebook, Github permet également d'en [faire usage](#).

## Sources

# 10 Responsive Web Design

## Site adaptatif ?

- Surfer depuis : PC, mobiles, tablettes, tv, ...
- UX : navigation avec un minimum de zoom, pan, scroll
- S'adapter aux spécificités des appareils
  - orientation
  - taille caractères
  - modes d'interaction (ex : tactile, hover, ...)
  - ...
- 1 seul site à gérer : m.cool.com ni de cool.com/mobile
- Le même contenu pour tous
- Souvent basé sur la largeur de l'écran
- CSS3
- *Responsive Web Design* (1), [Exemple](#)

## Techniques

- Media queries : Taille de l'écran (ou sortie)
- UNITES RELATIVES
- Fonts : Dimensions en em
- Fluid Grids : Disposition et taille des éléments en %
- Flexible images (and media) : Taille des médias en %
- Autres considérations
  - Adaptatif avec [grilles fixes](#)
  - [Performances](#) : tps chargement, requêtes inutiles, ...
  - Transitions CSS
  - ...
- [Exemple](#)

## Media Queries

- media type : all, screen, print, tv, braille, handheld, ...

```
<link rel="stylesheet" type="text/css" href="style.css" media="screen" />
<link rel="stylesheet" type="text/css" href="printfriendly.css" media="print" />
```

- media query

1. dans élément link

```
<link rel="stylesheet" type="text/css" media="screen and (max-device-width: 800px)"
href="style800.css" />
```

2. dans une feuille de style

```
#nav {float: right;}
    #nav ul {list-style: none;}
    @media screen and (min-width: 400px) and (orientation: portrait)
    {
        #nav li {float: right; margin: 0 0 0 .5em; border: 1px solid #000000;}
    }
    @media screen and (min-width: 800px)
    {
        #nav { width: 200px; }
        #nav li {float: left; margin: 0 0 0 .5em; border: none;}
    }
```

3. Règle CSS import

```
@import url(style600min.css) screen and (min-width: 600px);
```

## Media Queries

width, height, device-width, device-height, orientation, aspect-ratio,  
device-aspect-ratio, color, color-index, monochrome, resolution, scan, grid

- Règles CSS selon medium (souvent min-, max-width)
- Opérateurs : only, not, and
- Au moins 3 layouts : mobile, tablet, desktop
- Resolution breakpoints : 320, 480, 600, 768, 1024, 1200px
- Souvent ces règles sont utilisées pour :
  - agrandir la taille du texte
  - agrandir la taille des zones cliquables (utilisation au doigt)
  - faire passer le contenu sur une seule colonne
  - masquer ou afficher des éléments spécifiques
  - ajuster les dimensions et marges
- Attention à l'ordre de chargement



## Meta Tag **viewport**

- Introduit pour **iPhone**, puis standard de fait
  - Par défaut, l'affichage est réduit (980px affichés sur écran 320px)
  - Meta tag viewport permet de changer ce ratio
- viewport : display area  $\neq$  rendering surface
- Mobiles : viewport > écran (iphone 3 : v :980 / é :320)
- Media queries comparent au viewport
- Mise à jour du viewport à la taille de l'écran nécessaire :

```
<meta name="viewport" content="width=device-width; initial-scale=1.0">
```

## Résultat = Cible / Contexte

### Texte

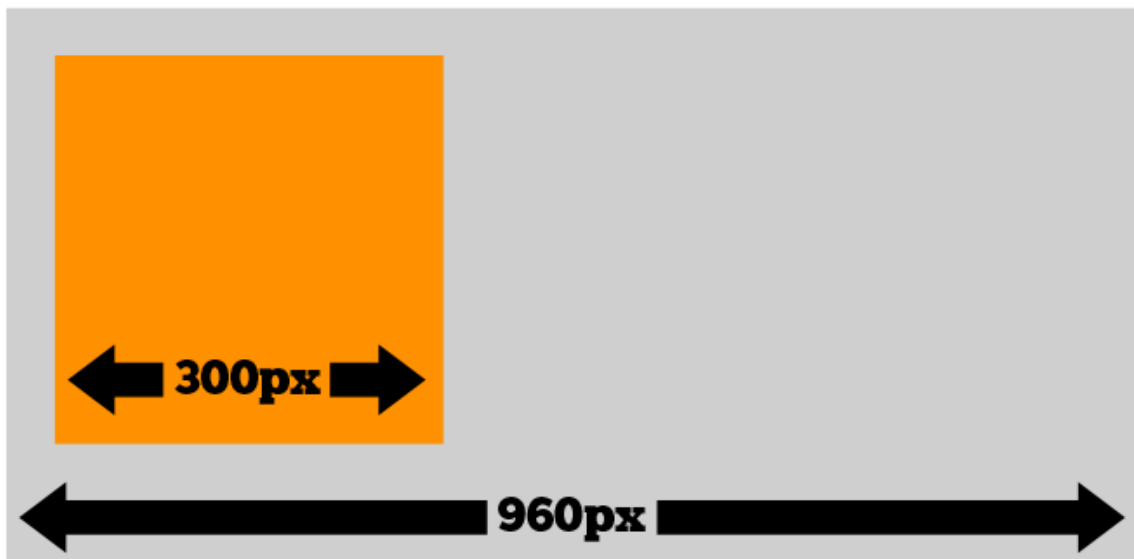
- Texte redimensionnable : em pour les polices
- 1rem : font-size:100% dans élément html (16px/défaut)
- 1em : font-size:100% dans élément courant
- Conversion px -> em :  $result = target / context$ 
  - ne pas arrondir
  - laisser le rapport en commentaire

### Fluid Grids

- Layout basé sur une grille en pixel
- Conversion px -> % : **result = target/context**
  - ne pas arrondir
  - laisser le rapport en commentaire
- Appliqué au style des divs :

width, margin, padding, background-position, ...

**target / context = result**



**300px / 960px = 31.25%**

FIG. 10.1 : Target / Context

## Responsive Images

- Nouveautés de HTML 5
  - Éléments <picture>, <source>
  - Attributs srcset et sizes
- [Besoins](#)
  - Écrans haute densité : srcset
  - Taille variable : srcset et sizes
  - [Substitution](#) et modification layout : <picture>, <source>
  - Choix formats de fichiers <picture>
- [Différences](#) entre <picture> et srcset
- [Exemple](#) en français

## Flexible images

- Eviter qu'une image ne déborde de son conteneur
  - La réduire

```
img, embed, object, video{ max-width: 100%; }
```

- La découper

```
.feature { overflow: hidden; }  
.feature img { display: block; max-width: auto; }
```

- Pas de standard pour servir différentes tailles de fichier
- Quelques idées recensées par *Smashing Magazine* (2)

## Outils

- Tester
  - Simulateur mobile des devtools, largeur browser
  - [bookmarklet](#) pour afficher les media queries
  - mais surtout tester sur mobile
- Et Après ? [MOBILE FIRST](#), [OFFLINE FIRST](#), [PWA](#)
- framework ou from scratch ?

## Références

- Exemples
  - Site support du [livre](#) d'Ethan Marcotte
  - [mediaqueri.es](#)
  - [thenextweb](#)
  - [designshack](#)
- Plus loin...
  - [Généralités](#)
  - [viewport et media queries](#)
  - D'autres techniques, liste de Smashing magazine (2)
  - Améliorer la [performance](#)
  - [Making sites more responsive, responsibly](#)

## Pratique

- Workshop [Pierre Spring](#) 26.02.13, [sources](#)
- Tester les exemples sur un mobile
- Comprendre les sources
- Présentation adaptative de votre équipe de projet

## Sources

1. MARCOTTE, Ethan. Responsive Web Design. [en ligne]. 25 mai 2010. [Consulté le 6 novembre 2017]. Disponible à l'adresse : <https://alistapart.com/article/responsive-web-design>
2. THE SMASHING EDITORIAL. Responsive Web Design Techniques, Tools and Design Strategies. [en ligne]. 22 juillet 2011. [Consulté le 6 novembre 2017]. Disponible à l'adresse : <https://www.smashingmagazine.com/2011/07/responsive-web-design-techniques-tools-and-design-strategies/>

# 11 HTTPS

## Sécuriser un site web

- Authentification du serveur
  - Assurer que le serveur est celui qu'il prétend être
- Intégrité des données
  - Assurer que les données reçues sont celles qui ont été envoyées
- Confidentialité des données
  - Eviter que des tiers ne puissent voir les données
- Authentification du client (optionnelle)
  - Assurer que le client est celui qu'il prétend être
- Pour un site web, ces services sont fournis par https
  - HTTPS : HTTP sécurisé par SSL/TLS, par défaut sur le port 443

## Secure Socket Layer --> Transport Layer Security

- Conçu par Netscape (v2.0 en 1994, v3.0 en 1996)
- Brevet racheté par l'IETF : TLS v1.0 en 1999 (SSL 3.1)
- Couche Application :
  - Entre les couches transport et application
  - Pas besoin de modifier la pile TCP/IP
- Possibilité de sécuriser d'autres protocoles :
  - HTTP, SMTP, SIP, ...
- Services offerts :
  - Authentification serveur + intégrité données
  - Confidentialité des données
  - Authentification optionnelle du client
- Certificats (clé publique associée au certificat)

## Rôle d'un certificat

- Garantir le lien entre une entité physique et une entité numérique :
  - Intégrité des données
  - Authentification
  - Confidentialité
- Document contenant une identité et une signature numérique
- Utilisations courantes : https, mails
- Délivré par une autorité de certification
- Certificats clients

## Autorité de Certification

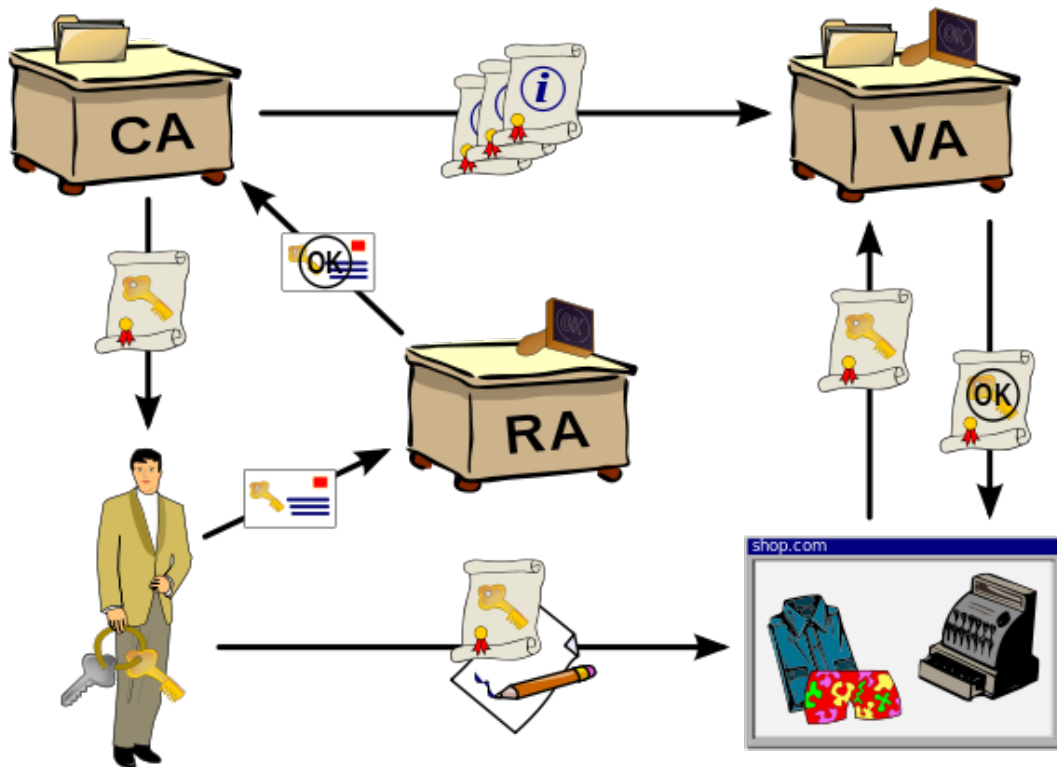
- Tiers de confiance
  - enregistrée et certifiée par des autorités publiques ou de gouvernance de l'Internet
- Rôle :
  - Vérifier et garantir les informations sur l'entité
  - Emettre, délivrer et révoquer les certificats
  - Leur assigner une période de validité
  - Maintenir la liste des certificats valides/révoqués
- Certificats auto-signés :
  - usage interne
  - pas de tiers de confiance

## Contenu d'un certificat X509

- version de X.509 (v3, depuis 1996)
- numéro de série du certificat
- algorithme de chiffrement utilisé pour signer le certificat
- nom de l'AC émettrice
- informations sur la clé publique
- dates de début et fin de validité du certificat
- clé publique du propriétaire du certificat
- signature de l'émetteur du certificat (thumbprint)
- ...

## Composants d'une PKI

CA : Autorité de certification - VA : Autorité de validation - RA : Autorité d'enregistrement



## Scénario simplifié de connexion HTTPS

1. Le client demande une page sécurisée
2. Le serveur émet sa clé publique et son certificat
3. Le client vérifie la validité du certificat (et qu'il correspond au site)
4. Le client utilise la clé publique pour chiffrer la clé symétrique (CS) utilisée ensuite
5. Le serveur déchiffre cette CS (avec sa clé privée) et l'utilise pour décoder la requête HTTPS
6. Le serveur répond à la requête en chiffrant avec la CS
7. Le navigateur décode la réponse avec la CS

- En [images](#) ou en [slides](#)
- 2-5 en TCP

## Déploiement

- Installer OpenSSL

- (Créer son autorité de certification si autosigné)
- Obtenir le certificat et la clé privée du serveur
- Configurer httpd. Pour Apache :
  - virtual host (port 443), ssl.conf, (ports.conf)
- Création de l'arborescence sécurisée
- Démarrage serveur
- OU BIEN utiliser [Let's encrypt](#)
- OU BIEN utiliser un serveur pré-configuré comme [Caddy](#)

## Ressources

- [Security Party 23.10.2009](#)
- [SebSauvage](#)
- Diagramme de séquence [EventHelix](#)
- [HowTo certificats SSL](#)
- [Faux Certificat](#)
- Autorités de certification :
  - [Let's Encrypt](#)
  - [CA Cert](#)
  - [Startcom](#)
  - [Verisign](#) (Symantec)
  - [Thawte](#)
- Différences TLS / SSH : [Snailbook](#), [StackExchange](#)

## Sources



# 12 Risques

## Risque

- Faille ou bug permettant d'altérer le fonctionnement
- Un attaquant pourra :
  - Modifier le fonctionnement
  - Accéder ou modifier les données
- Présence possible à tous les niveaux d'un système
  - Application
  - Serveur et Client
  - OS
  - SGBD, ...
- Responsabilité des développeurs :
  - OS, serveurs, langages : patches rapidement disponibles
  - nos applications : **c'est nous qui en sommes responsables**

## Injection de code

- Données mal validées : possibilité d'exécuter du code
- Passées par requêtes :
  - formulaires
  - URL
  - ...
- Type de code injectable : TOUS !
  - HTML
  - SQL
  - Javascript
  - ...

## Injections SQL

- Modifier les requêtes envoyées au SGBD
- Obtention d'un résultat non prévu par le développeur
- Deviner la structure du code pour l'exploiter
- SQL est puissant : UNION, INTO DUMPFILE, ...

### Exemples

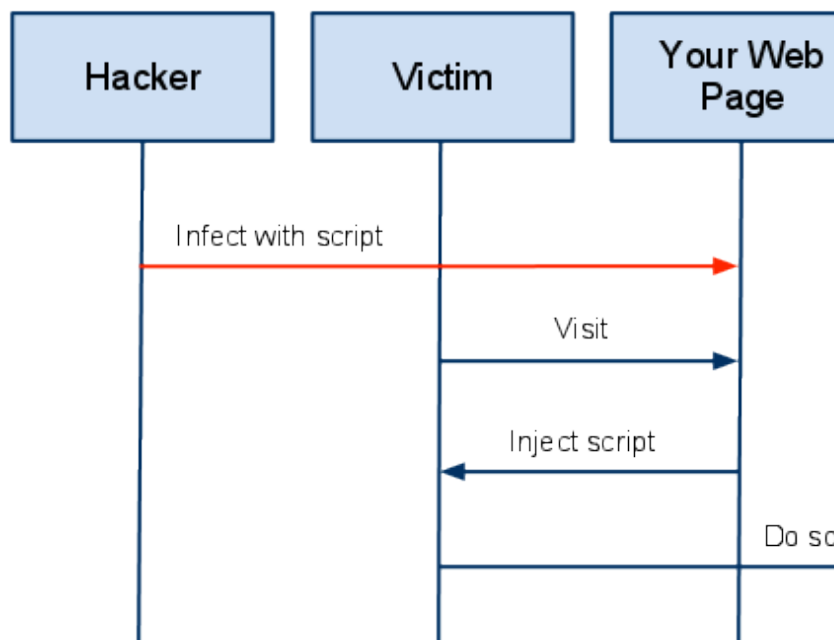
```
SELECT titre, num FROM livres WHERE num=2 UNION
SELECT login, password FROM user INTO DUMPFILE 'www/exploit.txt'
```

## Eviter les injections SQL

- N'accepter que des caractères valides
- A défaut, neutraliser les caractères dangereux
- Utiliser les entités HTML
- Vérifications strictes dans le code
- Eviter les noms prévisibles pour une appli critique

## Cross Site Scripting (XSS)

- Injection de code (html et script)



A High Level View of a typical XSS

- Exécution par le navigateur du client

## Cross Site Scripting (XSS)

- Enjeux : tout ce qui est possible en JS
  - Redirection
  - Lecture de cookies (session, ...)
  - Envoi d'info à un autre serveur
  - Modification du contenu de la page
  - ...
- Souvent utilisé pour transmettre le cookie de session

```

```

## 3 types de XSS

- Reflected XSS
  - Affichage d'une partie de la requête (recherche, erreur, ...)
- Stored XSS
  - Stockage dans la BDD et affichage (= exécution) par plusieurs clients
- DOM based XSS
  - Exécutée lors de la modification du DOM ([Exemple](#))

## Cross Site Request Forgery (CSRF - Sea Surf)

- Principe :
  - Faire réaliser à quelqu'un une action à son insu, avec ses propres infos d'authentification (credentials)
- Envoi par mail ou post forum de liens ou images
- Les URL correspondent à actions (vote, suppression, ...)

[Exemple](#) (SOP, CORS)



FIG. 12.1 : top 500 passwords cloud

# Phishing

- Site sosie d'un site officiel :
  1. L'utilisateur saisit ses données...
  2. ... l'attaquant les récupère...
  3. ... et les utilise sur le site officiel
- Difficile à contrer pour le développeur
- L'utilisateur doit être prudent
- Bien lire les URLS et le GUI du navigateur (Exemples)

## Risques non liés à l'application

- IoT : souvent mal sécurisé ([shodan.io](https://www.shodan.io))
- DoS
- Spoofing (IP, DNS, ARP)
- Buffer Overflows (surtout en C)
- Trojans, backdoors
- Usurpation de mots de passe : dictionnaire, force brute
- **SOCIAL ENGINEERING !!!**

## Top 500 passwords cloud

## Mots de passe

- 91% of users have a password from the top 1000 ([source](#))
- Our passwords habits [revealed](#)
- xkcd's [password strength](#)
- [passwordless](#) authentication

- 2017 : [NIST 800-63-3](#) suivi par la [NCSC](#)
  - Mots de passe longs plutôt qu'avec des caractères spéciaux
  - Ne forcer le changement qu'en cas de nécessité
  - Autoriser et accompagner l'utilisation de password managers
  - Utiliser la 2FA

## Collecte d'information

- Toute information est bonne pour l'attaquant
  - Messages d'erreur
  - Configuration OS serveur
  - Configuration serveurs (http, sql, php, ...)
  - Identifiants et commentaires dans sources -au cas où-
  - SOCIAL ENGINEERING !
- Le développeur doit laisser filter un minimum d'info !
- Utilisée aussi par les ``white hats" (etical hackers) : [Honeynet Project](#)

## Bonnes pratiques

- Configuration stricte du serveur
- Valider toutes les entrées (formulaires, requêtes HTTP)
- Filtrage/encodage de toutes les entrées en entités HTML
- Ne jamais afficher directement une saisie de formulaire
  - Ni aucune donnée transmise par HTTP avant de l'avoir filtrée !
- Tester ses formulaires avec des expressions à risques
- Contrôler le maximum de paramètres (même si redondant) :
  - Session, IP, user agent, proxy, ...
- Utiliser un framework
  - ces bonnes pratiques sont déjà implémentées
- Suites et logiciels de test

## Top 10 OWASP 2017

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities ([XXE](#))

5. Broken Access Control
6. Security Misconfiguration
7. Cross Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

- Top 10 [mobile](#)

## Références

- Référence
  - [OWASP](#)
- Exemples, explications
  - [Présentation XSS et CSRF](#) en français
  - [Protection CSRF](#) en français
- Utilitaires, tutos, exercices
  - [Web Goat](#)
  - [Insecure Labs](#)
  - [Google-Gruyere](#)
  - [Tutoriaux et challenges](#) en français

## Sources

## 13 Ruby on Rails



## Connexion

Nom de domaine et port SSH sur : <http://srvz-webapp2.he-arc.ch/>.

# Exemple

```
$ ssh -p 2030 yoan@srvz-webapp2.he-arc.ch
yoan@yoan$ more ~/README.md
```

## Mise à jour de Rails

```
yoan@yoan$ rails -v
Rails 5.0.0.1
yoan@yoan$ gem update
Updating installed gems
...
yoan@yoan$ rails -v
Rails 5.0.1
```

## Une application Ruby

Comme pour Laravel, c'est une bonne pratique d'avoir un répertoire pour le contenu publiable sur Internet.

```
$ cd /var/www/app
$ ls
config.ru
Gemfile
Gemfile.lock
public/nginx-puma.png

$ more config.ru
```

## Rack 101

```
run ->(env) do
  [
    200,
    {"Content-Type" => "text/html; charset=utf-8"},
    [
      "<!DOCTYPE html>",
```



```

    "<meta charset=utf-8>",
    "<title>Hello!</title>",
    "<h1>Hello</h1>",
    "<p>:-)"
  ]
]
end

```

Une fonction, Proc ou lambda qui :

- reçoit un tableau associatif de son environnement ;
- retourne un triplet de réponse HTTP.

Réponse HTTP :

- le code HTTP ;
- un tableau associatif des entêtes HTTP ;
- un itérateur sur le corps du document.

## Gemfile

Un paquet Ruby se nomme une *gemme*.

```

# Gemfile
source "https://rubygems.org"

gem "puma", "~> 3.6.2"
gem "rack"

```

Comme le composer.json pour PHP.

## NGINX

```

root /var/www/app/public;

location / {
    try_files $uri/index.html $uri @rack;
}

location @rack {
    proxy_pass http://puma;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_redirect off;
}

```

```
upstream puma {
    server unix:/tmp/puma.sock fail_timeout=0;
}
```

Le serveur HTTP qui sert les fichiers statiques (public) et redirige le reste vers le serveur d'application Ruby (puma).

## Puma

Le serveur d'application pour Ruby.

```
#!/usr/bin/env puma

environment "production"

directory "/var/www/app"
bind "unix:///tmp/puma.sock"

# À ajouter.
plugin :tmp_restart
```

En PHP, nous utilisons PHP-FPM. Qu'utilisez-vous avec JEE ?

## Serveur

```
$ ls /etc/services
cron nginx puma sshd syslog

$ pstree
tini  runsvdir  runsv  cron
      runsv   nginx    4*[nginx]
      runsv   syslog-ng
      runsv   bundle   {reactor.rb:151}
                                   {ruby-timer-thr}
                                   {server.rb:301}
                                   6*[{thread_pool.rb*}]
```

## Exercice 1

Modifiez l'environnement *puma* en développement.

`http://[ PRENOM.NOM | GITHUB ].srvz-webapp2.he-arc.ch/` doit afficher :

```
RACK_ENV
  development
```

## Première application

Archivez app.

```
$ cd /var/www  
$ mv app demoapp
```

Créez une nouvelle application Rails.

```
$ rails new app --database=postgresql  
$ cd app
```

Si vous changez le nom, vous devez modifier les configurations des serveurs.

## Plein de fichiers

app/	# votre code
bin/	
config/	# fichiers de config
config.ru	# point d'entrée, « index.php »
db/	# migrations et seeds
Gemfile	# comme le composer.json
Gemfile.lock	
lib/	
log/	
public/	# fichiers publics
Rakefile	
README.md	
test/	# tests unitaires, fonctionnels, etc.
tmp/	
vendor/	

## Exercice 2

Que peut-on faire à l'aide de la commande rails ?

Et de la commande bundle ?

Avant Rails 5, rails et rake avaient des rôles séparés, condensés dans rails.

## Premier démarrage

```
$ sudo sv restart puma
```

Kaboom !

## Connexion

Utilisez [pgAdmin](#) pour vous connecter à votre base de données.

```
$ echo $GROUPNAME $PASSWORD
```

Ou pour les durs à cuire :

```
$ psql -h $POSTGRES_HOST -U $GROUPNAME
> \l
> \dn
> \dt
> \q
```

## Configuration

```
default: &default
  adapter: postgresql
  encoding: unicode
  pool: <%= ENV.fetch("RAILS_MAX_THREADS") { 5 } %>
  host: <%= ENV.fetch("POSTGRES_HOST") { "localhost" } %>
  port: <%= ENV.fetch("POSTGRES_PORT") { 5432 } %>
  database: <%= ENV["GROUPNAME"] %>
  username: <%= ENV["GROUPNAME"] %>
  password: <%= ENV["PASSWORD"] %>
```

```
development:
  <<: *default
```

```
test:
  <<: *default
  schema_search_path: test
```

```
production:
  <<: *default
  schema_search_path: production
```

## Application de démon

Téléchargez l'application pré-configurée pour vous.

```
$ cd /var/www
$ rm -rf app

$ git clone \
    https://github.com/HE-Arc/rails-intro \
    app

$ cd app
$ bundle install
```

## Migration

Installation de la base de données.

```
$ rails db:migrate
```

Que s'est-il passé ?  
(hint : `git status`)

## Exercice 3

Créez un produit possédant un titre, une description et un prix.

## Réponse

Nous obtenons une migration, un modèle et un test unitaire.

```
$ rails generate model \
    product \
        title:string \
        description:text \
        price:decimal
```

RAD !

## Exercice 4

Corrigez le test qui échoue en corrigeant les *fixtures*.

```
$ rails db:rollback

$ git reset --hard
$ git clean -fd
```

```
$ git checkout product
$ rails db:migrate

$ rails test
```

## Test unitaire

```
# test/models/product_test.rb

class ProductTest < ActiveSupport::TestCase
  test 'T-shirt has a price' do
    product = Product.find_by(title: 'T-shirt')
    assert 0 < product.price
  end
end
```

## Solution

```
# test/fixtures/products.yml

tshirt:
  title: T-shirt
  description: Superbe maillot de corps
  price: 9.99
```

## Validation

Selon Ruby on Rails, la logique métier ne doit pas se trouver dans la base de données.

```
# app/models/product.rb

class Product < ActiveRecord::Base
  validates :title, presence: true
  validates :price, numericality: { greater_than: 0 }
end
```

## Exercice 5

Testez les règles de validations ci-dessus en ajoutant des tests.

```
$ git reset --hard
$ git checkout validation
$ rails test
```

## Solution

```
# test/models/product_test.rb

test 'must have a title' do
  assert_not Product.create(price: 10).valid?
end

test 'must have a price greater than zero' do
  assert_raise do
    Product.create!(title: 'Untitled', price: 0)
  end
end
```

## Contrôleur

```
$ rails g controller products index

app/assets/javascripts/products.coffee
  /stylesheets/products.scss
  /controllers/products_controller.rb      # def index; end
  /helpers/products_helper.rb
  /views/products/index.html.erb          # index.html.erb

config/routes.rb                          # get 'products/index'

test/controllers/products_controller_test.rb # should get index
```

Par convention, un modèle est au singulier et un contrôleur au pluriel.

## Test unitaire

```
# test/controllers/products_controller_test.rb

test 'should get products on /' do
  get '/'

  assert_response :success
  assert_not_nil assigns(:products)
end
```

## Exercice 6

Corrigez le test du contrôleur.

```
$ git reset --hard
$ git clean -fd
$ git checkout controller

$ rails test
```

## Solution

```
# config/routes.rb
root 'products#index'

# app/controllers/products_controller
def index
  @products = Product.all
end

# app/views/products/index.html.erb
<% @products.each do |product| %>
  <h2><%= product.title %></h2>
<% end %>
```

## Taille

Création d'un modèle pour les tailles de nos t-shirts.

```
$ rails generate model size name:string
```

## Exercice 7

Créez un seeder pour les tailles allant de XS à XXL.

```
$ git reset --hard
$ git clean -fd
$ git checkout sizes
$ rails db:migrate

$ rails db:seed
```



```
# Test
$ rails console
> pp Size.all
```

## Solution

```
# db/seeds.rb

Size.create([
  {name: 'XS'},
  {name: 'S'},
  {name: 'M'},
  {name: 'L'},
  {name: 'XL'},
  {name: 'XXL'}
])
```

## Relation Produits - Tailles

```
$ rails g migration associate_products_and_sizes
```

```
# db/migrate/..._associate_products_and_sizes.rb

create_table :products_sizes do |t|
  t.references :product, :index => true
  t.references :size, :index => true
end
```

## Many-to-many

Dans chaque modèle.

```
# app/models/product.rb
has_and_belongs_to_many :sizes, uniq: true

# app/models/size.rb
has_and_belongs_to_many :products, uniq: true
```

## Test

```
$ git reset --hard
$ git clean -fd
$ git checkout habtm
```

Tests depuis la console.

```
$ rails console
> xxl = Size.find_by(name: 'XXL')
> xxl.products.size
=> 0
```

```
xxl.products.create(title : `A`, description : `B`, price : 10)
```

## Administration

```
$ more Gemfile
```

```
# Automagic admin interface.
gem 'rails_admin', '~> 1.1'
```

```
$ bundle install
$ rails g rails_admin:install
$ touch tmp/restart.txt
```

```
$ git reset --hard
$ git checkout admin
$ bundle install
```

## Image

Ajoutez une image à vos produits

```
$ more Gemfile
```

```
# Thoughtbot's paperclip to upload files
gem 'paperclip', '~> 5.1'
```

```
$ bundle install
```

## Migration

```
$ rails g migration add_image_to_product
```

```
def change
  change_table :products do |t|
    t.attachment :image
  end
end
```

## Exercice 8

Faites qu'on puisse attacher une image depuis l'interface d'administration.

```
$ git reset --hard
$ git clean -fd
$ git checkout images
$ rails db:migrate
```

Indice : lire la documentation de paperclip.

## Solution

```
# app/models/product.rb

has_attached_file: image
validates_attachment_content_type :image, \
  content_type: /\Aimage/
validates_attachment_file_name :image, \
  matches: [/png\z/, /jpe?g\z/]
```

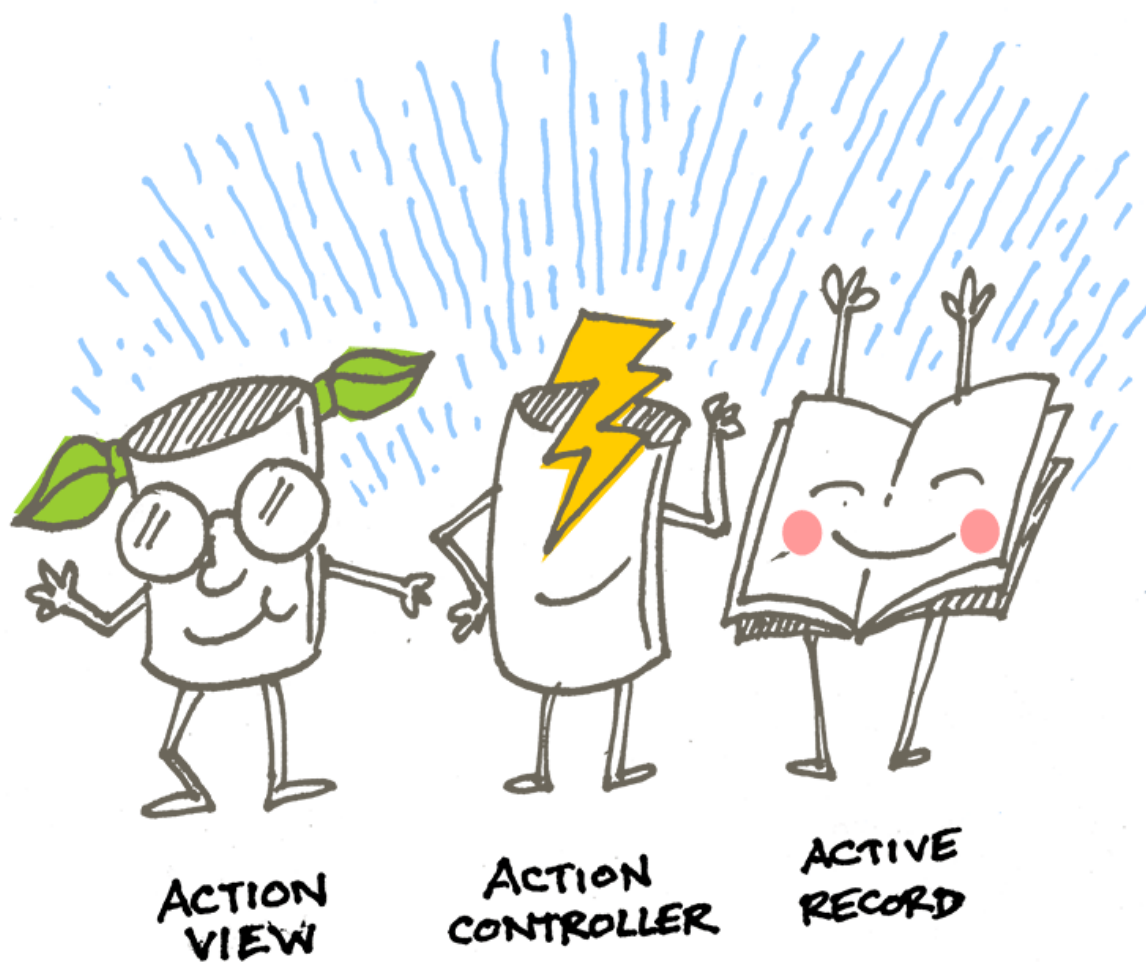
## Ressource

Il aurait été possible de créer modèle, contrôleur et routes de type REST.

```
$ rails generate resource person
$ rails routes
...
```

Testez !

## Détails intéressants de Rails



### CSS et JavaScript

- foundation-rails
- bootstrap-sass, ~> 3.3.7
- bootstrap, ~> 4.0.0.alpha6
- basscss-rails
- bulma-rails
- mui-sass
- etc.

Voir [Asset Pipeline](#)

Rails 5.1 proposera de gérer ces éléments-là via webpack ou yarn. D'ici là, il nous faut passer par les gems associées.

## ActionCable

La nouveauté de Rails 5.0.

Gestion simplifiée des WebSocket permettant d'incorporer des fonctionnalités « temps-réel ».

Voir [Action Cable Overview](#)

## ActiveJob

Gestion des tâches de fond, comme envoyer des e-mails, redimensionner des images, ...

Voir [Active Jobs Basics](#)

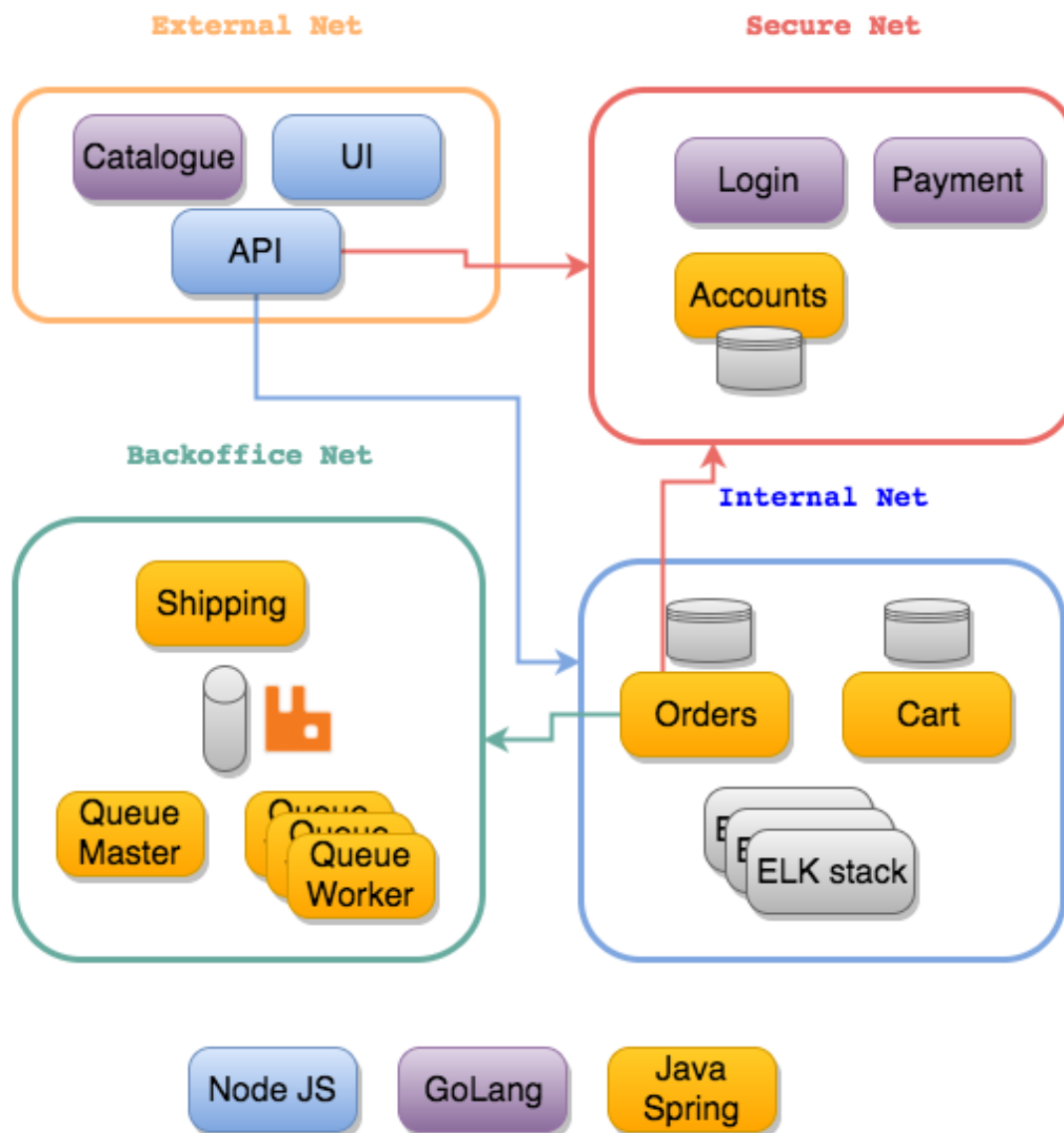
## ActionView

La bonne méthode pour créer des formulaires et les lier à des données.

```
<%= form_for @article, url: {action: 'create'} do |f| %>
  <%= f.text_field :title %>
  <%= f.submit 'Create' %>
<% end %>
```

Voir [Form Helpers](#)

## Problème avec Ruby on Rails



## Conclusion

- Laravel tire son inspiration première de Ruby on Rails.
- Rails est plus cohérent dans son ensemble tirant partie des fonctionnalités de Ruby.



**DHH** ✓  
@dhh

2017: Start fewer things, but finish them.

RETWEETS  
**202**

LIKES  
**393**



5:52 PM - 1 Jan 2017

(1)

## Difficultés pour vous

- Construisez un produit au fur et à mesure
- Déployez souvent
- Essayer des bibliothèques
- Et ayez un plan !



## Sources

1. HEINEMEIER HANSSON, David. 2017 : Start fewer things, but finish them. [en ligne]. 2017. [Consulté le 7 février 2017]. Disponible à l'adresse : <https://twitter.com/dhh/status/>

815601578329575424