# LES FRAMEWORKS

## DÉFINITION DE FRAMEWORK

**Framework** peut se traduire par "cadre de travail", le terme anglais désigne un ensemble de **librairies** et **d'outils** ayant pour objectif principal d'améliorer la **productivité** en programmation informatique. Cela se traduit par un énorme gain de temps.

En effet l'idée est de permettre aux développeurs d'accéder à des codes **préétablis** et **réutilisables** pour le développement aussi bien d'un site internet, que d'un jeu, d'une extension ou d'une application mobile.

Cependant il faut bien comprendre qu'un Framework utilise un **langage de programmation** qu'il faut connaître afin de maîtriser le Framework, le but étant de comprendre ce que le Framework fait pour nous et ce qu'il se passe derrière.

De plus cela nous permet d'agir sur le Framework. Il faut donc ne pas griller les étapes et ne pas se lancer à corps perdu dans un Framework sans avoir étudié le langage associé avant. Exemple, ce n'est pas parce qu'on connaît trois bouts de Javascript qu'il faut se lancer dans du React.

Il existe différents types de Framework dans le développement web, certains sont créés pour la réalisation de projets de grande envergure tandis que d'autres sont essentiels pour mener à bien de petites tâches. On utilisera donc pas le même Framework que l'on veuille réaliser une application web complexe ou un site vitrine type cv numérique par exemple. Il ne faut pas utiliser un bazooka pour tuer une mouche.

Il peut être nécessaire de combiner plusieurs Framework lors de la réalisation d'un projet, exemple :

On utilise Laravel (Framework orienté back-end Php) pour réaliser son API,
Reactjs (Framework front-end Javascript) pour réaliser la partie front-end du site que l'on combine à Tailwindcss (Framework CSS).

## POURQUOI UTILISER UN FRAMEWORK ?

Voici les principaux avantages d'utiliser un Framework :

- Un gain de temps énorme lié au fait que vous partiez d'une structure établie qui permet de se concentrer sur l'essentiel, les composants spécifiques de votre code.

On considère en moyenne qu'un projet développé en 10h en vanilla se fait en 3 grâce à un Framework associé au langage.

- Une organisation optimisée grâce à la structure du projet, ce qui permet d'avoir une organisation des fichiers cohérente et faciliter la relecture de code. In fine cela représente également un gain de productivité.

- Un gros gain en termes de sécurité dès lors que le Framework est compris, bien utilisé et maintenu une fois déployé.

- Un travail collaboratif facilité, un Framework permet de bien segmenter les tâches et éviter les futurs conflits sur Git.

- Une communauté qui représente une source d'informations intarissable ainsi qu'une aide possible en cas de soucis.

- Le coût, en effet les frameworks les plus populaires sont open source donc gratuit. De plus l'open source est un bon moyen de contribuer et ainsi participer à l'évolution du framework.

## L'HISTOIRE DES FRAMEWORKS

Les Frameworks sont apparus rapidement avec le boom de l'internet, au début des années **2000**. On peut considérer qu'ils ont suivis le développement des différents langages de programmation et l'évolution des technologies. En effet un framework étant ni plus ni moins qu'un outil créé par des développeurs pour se faciliter le travail il est normal que les Frameworks suivent l'évolution de ceux-ci. Lorsque la communauté pratique un langage, elle créé des outils pour ceux-ci.

Ainsi des Frameworks comme Hibernate et Spring pour le langage Java sont apparus en 2001 et 2003. Le framework Php Symfony lui est apparut en 2005. Les Frameworks Javascript sont eux apparus plus récemment, en même temps que l'augmentation de l'utilisation du langage. AngularJs est né en 2010 par exemple, son évolution qui est celle utilisée Angular en 2016.

De nos jours les Frameworks Javascript se multiplient et prennent de plus en plus d'ampleur. Il était encore inconcevable de réaliser son back-end avec du Javascript il y a une décennie. Plus les besoins augmentent et plus le nombre de Frameworks augmentent.

## LES DIFFÉRENTS TYPES DE FRAMEWORKS

### FRAMEWORK D'INFRASTRUCTURE SYSTÈME

Ces Frameworks sont faits pour réaliser des **systèmes d'exploitation**, des interfaces graphiques, des outils de communication.

Exemple : Framework.Net, Eclipse, Netbeans, Struts

### FRAMEWORK D'INTÉGRATION INTERGICIELLE

Ces Frameworks fusionnent **plusieurs technologies** dans une interface unique.

Exemple: Ampoliros, SOAP, XML

### FRAMEWORKS D'ENTREPRISE

Ils permettent de développer des applications pour un secteur défini d'une entreprise spécifique. Souvent basé sur des frameworks existants, ils permettent souvent de préparer un environnement de développement courant dans l'entreprise. Certaines entreprises mettent même ces Frameworks en open-source.

Exemple : Sylius pour Symfony spécialisé e-commerce.

## FRAMEWORKS DE GESTION DE CONTENU

Ces Frameworks permettent de poser les fondations d'un système de gestion de contenu. Permet de concevoir, gérer et mettre à jours des sites Web ou des applications mobiles de manière **dynamique**. Ils doivent pouvoir être utilisés par plusieurs personnes simultanément, proposer une chaîne de publication de contenu et permettre de gérer séparément la forme et le contenu.

Exemple : Wordpress, Drupal.

## LES TYPES D'USAGE

Selon le projet et les besoins qui lui sont associés, que ce soit pour mettre en ligne une page web, développer une application, il existe différents types d'usage pour un Framework. On peut les résumer ainsi :

- Les frameworks applicatifs pour le développement d'application web (Symfony, Ruby on Rails ou Jango)
- Les frameworks de présentation de contenu web (Bootstrap, Tailwindcss, Semantic UI, Foundation)
- Les frameworks de développement de bureau (Cocoa, Qt)
- Les frameworks de logging (Log4j)
- Les frameworks de persistance (SQLAlchemy, Propel)

## COMMENT CHOISIR UN FRAMEWORK

Tout d'abord il reste toujours nécessaire de se demander dans un premier temps si l'utilisation d'un Framework est pertinente sur le projet, par exemple un site vitrine utilisant peu de Javascript et de CSS peut être réalisé plus simplement sans Framework qui alourdissent la structure du projet ainsi que ses performances.

Cependant dans la majorité des cas vous devrez faire appel à des Frameworks lorsque vous réaliserez des projets. Sur de gros projets un Frameworks va apporter de la facilité dans le

développement, un gain en maintenabilité et de propreté de code. En équipe, tout. le monde doit respecter le cadre du Framework ce qui se traduit par une uniformisation des méthodes de travail naturelle et contrainte.

Pour rappel comme nous avons vu précédemment, bien souvent nous utiliserons plusieurs Frameworks dans un même projet. Mais comment savoir lequel privilégier plutôt qu'un autre.

Le but ici est de profiter au mieux des bénéfices du Framework, plus il est adapté à la situation, plus le développeur pourra en tirer les bénéfices notamment en termes de gain de productivité.

Pour se faire il faut prendre plusieurs critères en compte.

## LE LANGAGE DE PROGRAMMATION

Dans un premier temps le critère principal sera le langage de programmation utilisé par le Framework. On peut aussi tenir compte des technologies associées à ce Framework, langage de base donnée, moteur de template. Il est certain que si on ne maîtrise pas le C#, on ne choisira pas framework.NET pour développer le projet.

Il est donc important de lister tous les Frameworks pour lesquels on a les compétences en termes de langage.

Il reste aussi possible d'apprendre un langage nouveau afin de travailler avec un Framework voulut par l'entreprise.

## LES CARACTÉRISTIQUES TECHNIQUES

Il s'agit ici d'évaluer si le Framework est adapté à la structure du projet et à ses besoins techniques. Il est souvent utile d'effectuer une veille informative sur les différents Frameworks au travers de la documentation afin de comprendre les spécificités de celui-ci.

Il est naturellement plus facile d'utiliser un cadre de travail conçut pour répondre à un besoin spécifique plutôt que d'essayer d'en adapter un à ses besoins.

Il faut donc évaluer les frameworks en termes de performance, d'objectifs à atteindre et de spécificité technique du projet.

Il faut aussi tenir compte du coût, en effet certains Frameworks ne sont pas open source est peuvent engendrer des coûts importants, il faut en tenir compte dans le choix.

## LA POPULARITÉ

Plus un Framework est populaire, plus sa communauté est importante, réactive. Vous aurez donc plus facilement accès à des réponses face à des problèmes. Il ne faut cependant pas tomber dans les effets de mode, notamment dans les frameworks Javascript qui sont nombreux maintenant.

Choisir un framework célèbre c'est la garantie d'avoir un outil plus évolutif, plus complet et plus facilement maintenable. Mais il faut garder en tête que miser sur un framework jeune et en évloution si il est abandonné et plus maintenu pourrait engendrer un gros coût futur.

## QUEL EST LE MEILLEUR ? L'ETERNEL DEBAT...

Comme toujours l'être humain a besoin de classifier, à la façon des consoles de jeux laquelle est la meilleure. Au final c'est celle qui a les jeux qui vous conviennent. Pour les Frameworks c'est la même chose celui qui répond le mieux à mes besoins.

Mais comment départager deux Frameworks répondant à mes besoins de façon similaire.

Prenons le cas de Symfony et Laravel, deux Frameworks Php qui permettent de réaliser absolument tous les types de projet web. Laravel utilisant lui-même de nombreux composants de Symfony quand on a essayé les deux on se rend bien compte des similarités.

Dans ce cas le choix doit être personnel, en fonction de vos goûts en termes de syntaxe, de façon de coder.

# LES FRAMEWORKS CSS

Nous allons passer en revue les deux Frameworks CSS les plus populaires **Bootstrap** et **Tailwindcss**, ils sont liés au rendu visuel du site et à sa mise en forme par le CSS.

Bien souvent ce sont des systèmes de classes en **HTML** qui permettent d'appeler des ensembles de propriétés CSS. Ainsi en une seule classe dans le HTML on peut créer un élément stylisé, exemple la classe `card` de **Bootstrap**.

Il existe de nombreux Frameworks CSS, ils sont généralement assez simple à appréhender et souvent liés à l'utilisation d'un Framework, par exemple Semantic Ui qui est React friendly.

## BOOTSTRAP

Bootstrap est un **framework CSS** qui a été créé en 2011 par Mark Otto et Jacob Thornton, deux salariés de l'entreprise **Twitter**. Nous en sommes actuellement à la version 4 mais la version 5 est imminente.

Bootstrap est une collection d'outils **HTML, CSS** et **JavaScript** destinés à aider les développeurs de sites et d'applications web.

Bootstrap est devenu « le framework front-end » le plus populaire pour développer des projets responsive et mobile-first sur le web.

Dans les faits il permet avec des classes HTML de générer des ensembles de propriétés CSS qui permettent de styliser rapidement une page web mais aussi de facilement positionner les différents éléments HTML.

Ses avantages :

- Facilité d'apprentissage
- Très large communauté, nombreuses ressources
- Le plus populaire à l'heure actuelle, le plus utilisé dans les entreprises

Ses inconvénients :

- Sa rigidité qui peut engendrer un manque d'originalité
- Le risque de ne pas utiliser ses compétences de CSS
- Sa popularité qui donne une impression de déjà vu, exemple le menu hamburger

Lien vers la doc de Bootstrap

## TAILWINDCSS

Tailwindcss est un Framework CSS qui permet tout comme Bootstrap d'appeler des propriétés CSS grâce à des classes HTML. Nous en sommes à la version 3 sortie très récemment.

Cependant il n'envisage pas les choses comme Bootstrap, en effet les classes ne permettent pas d'obtenir de rendu visuel pour un élément HTML mais d'appeler uniquement du CSS.

Il est entièrement customisable avec une grand souplesse et facilité.

Sa popularité est grandissante avec le boom des Frameworks Javascript avec lesquels il est très adapté.

Ses avantages :

- Permet de ne se fixer aucune limite en terme de rendu
- Grand souplesse
- Originalité des possibilités

Ses inconvénients :

- Requiert plus de compétences CSS
- Plus difficile à mettre en place selon les projets
- Technologie émergente donc moins fournie en terme de ressources

Lien vers le site de Tailwind

## LES FRAMEWORKS FRONT-END

Il ne faut pas confondre **moteur de template** et Framework front-end, Twig ou handlebars sont des moteurs de template. Ils ne permettent d'implémenter que peu de logique à l'instar des Frameworks front-end qui eux peuvent implémenter autant de logique que nécessaire.

Les Frameworks front-end actuels utilisent **Javascript,** nous allons en étudier 3 qui sont les plus utilisés actuellement.

Lien vers les Tech Trends 2021

Nous allons donc passer en revue Reactjs, Angular et Vuejs mais il en existe plein d'autres.

Lien vers une liste de frameworks front-end

## REACTJS

Il a été développé en **2013** par **Facebook** et est actuellement un des plus populaires, il est utilisé par Netflix, Yahoo ou Instagram notamment. Reactjs est considéré comme une librairie plutôt qu'un Framework, cependant elle est tellement complète qu'elle s'appréhende comme un Framework.

Il permet de réaliser des développements front-end et la réalisation d'interfaces utilisateurs. Il se sert du langage Javascript et du JSX également pour générer le DOM. On mélange donc le HTML et le Javascript. Il est basé sur un système de composants autonomes et réutilisables. Le framework React Native permet de construire des applications mobiles avec du Reactjs.

Il est comparable à Vuejs mais n'est qu'une partie d'Angular.

Ses avantages :

- Fonctionnement très simple
- Souplesse de syntaxe
- Performance

Ses inconvénients :

- Peu devenir difficile à lire à cause de sa souplesse en terme d'architecture et de syntaxe
- Peu de documentation officielle
- Pas de moteur pour injection de dépendances

Lien vers la doc Reactjs

## ANGULAR

Initialement nommé AngularJs et créé par **Google** en **2009**, il était à cette époque très ressemblant aux possibilités de Reactjs puis en 2016 avec Angular, il est devenu capable de créer des applications très complexes.

Il prend en charge le développement d'interfaces utilisateurs en manipulant le DOM, il gère le data-binding ou encore l'injection de dépendance. Il utilise Typescript et Javascript et est également orienté composant. En utilisant Typescript Angular se veut plus structuré que Reactjs par exemple. Il impose un code structuré c'est aussi pour ça qu'il utilise Typescript qui permet lui aussi de produire du Javascript plus structuré. Angular est très apprécié dans le monde de l'entreprise.

Ses avantages :

- Framework complet qui permet d'envisager tous types de projet
- Les avantages liés à Typescript typage du Javascript, gain de sécurité
- Une grande communauté et des améliorations constantes

Ses inconvénients :

- Lourdeur du langage, complexité de la syntaxe
- Apprentissage de Typescript nécessaire
- Difficulté de migration pour la montée en version

Lien vers la doc Angular

## VUEJS

C'est le dernier né des Frameworks front-end Javascript, il a été créé en 2014 par Evan You, à ce jour il est maintenu par lui et son équipe. On peut le résumer comme simple, puissant et très agréable à pratiquer. Ce sont ces raisons qui en font le Framework tendance.

Il permet de créer des interfaces utilisateurs, cependant il se différencie de Reactjs et Angular car il ne nécessite pas de réaliser le projet entier avec le Framework et peut être utilisé uniquement sur les parties du projet où il est nécessaire. On peut donc implanter Vuejs dans un projet existant et lui donner un coup de neuf sans être pour autant obligé de tout recoder en respectant l'architecture d'un Framework. Tout comme React et Angular il permet le data-binding, la manipulation du DOM et est orienté composant.

Ses avantages :

- Facilité d'apprentissage
- Simplicité de mise en place
- Performances, meilleurs que React et Angular

Ses inconvénients :

- Sa jeunesse qui offre donc moins de ressources
- Plutôt orienté pour réaliser des SPA moins pour de gros projets
- Ne permet pas de créer d'applications mobiles car Vue Native reste méconnue et nécessite beaucoup d'améliorations

Lien vers la doc Vuejs

## LES FRAMEWORKS BACK-END

Les Frameworks **back-end** utilisent des langages informatiques côté serveur comme Php, Python, Ruby, nodeJS, C#, Java. Il faut savoir qu'au moins 80% du web utilise du Php pour son back-end ce qui veut dire que Php représente une des technologies les plus utilisées en entreprise encore à l'heure actuelle. De nos jours il est possible d'utiliser du Javascript pour la réalisation du back-end avec NodeJS et expressJS par exemple le stack **MERN** (MongoDB, ExpressJS, ReactJS, NodeJS).

Les Frameworks back-end peuvent permettre de renvoyer du html grâce notamment aux moteurs de template, ce qui en fait des Frameworks "Full-stack", prenant donc en charge la partie back-end et front-end.

Cependant avec l'avènement des SPA Javascript et les Frameworks associés comme ReactJS, Angular, VueJS, il est fréquent d'utiliser un Framework backend afin d'uniquement fournir les data par le biais d'une API ou non au front-end ainsi que de gérer les aspects métiers de l'application comme la sécurité.

Nous allons passer en revue quelques Frameworks back-end et étudier leur points forts et leur points faibles.

### DJANGO

Djanjo est un Framework back-end open-source basé sur le langage Python. Il suit le modèle MVC(Model View Controller). Il convient aux développements de sites web sophistiqués et riches en fonctionnalités basés sur les bases de données.

Il facilite une connectivité optimale, un code réduit, une grande ré-utilisabilité, et un développement rapide. Il utilise Python pour toutes les opérations dans Django et a même en option une interface d'administration pour faciliter les CRUD.

Il est utilisé notamment par Mozilla, Disqus, Washington Times.

Ses avantages :

- Facilité d'apprentissage et d'utilisation
- Richesse des fonctionnalités (authentification, sitemap, administration contenu...) ce qui lui permet d'envisager tous types d'applications
- La sécurité, permet de prévenir de nombreux problèmes (faille XSS, injection SQL, cross-site scripting)

Ses inconvénients :

- Lourd pour de petites applications ce qui engendre des problèmes de performance qu'il faudra donc combler
- Ne permet pas de traiter des multiples requêtes simultanées
- Le système ORM manque de fonctionnalités par rapport à la concurrence

Lien vers la doc Django

## RUBY ON RAILS

Il est également connu sous le nom de Rails, est un Framework d'application web côté serveur et utilisant le langage Ruby. C'est un logiciel libre proposé sous licence du MIT (gratuit et obtenu via Git). Il respecte l'architecture MVC, et encourage l'utilisation de standards du web comme XML, JSON pour les transfert de données, le CSS, HTML et Javascript pour l'interface utilisateur.

Ses avantages :

- Uniformité lié au système de stockage de fichier et aux conventions de programmation
- Développement de qualité grâce à des outils tels que Minitest, il est donc TDD friendly
- Adapté pour gérer des gros trafics

Ses inconvénients :

- Vitesse et performances d'exécution lentes qui nécessitent beaucoup de compétences pour être résolus
- L'interdépendance des composants entre eux peut engendrer des problèmes grave en terme d'architecture
- Peu de place à la créativité, dépendance aux bibliothèques et outils qui demande de se tenir à jour en permanence

Lien vers la doc de Ruby on rails

## ASP.NET CORE

C'est un framework open-source gratuit qui suit les traces d'ASP.NET, modulaire et qui peut fonctionner sur l'ensemble du Framework .NET. Il utilise le langage C# et permet de créer tous types d'applications web multi-plateforme.

Ses avantages :

- Codage minimum car ASP.NET Core utilise une technologie qui nécessite moins de code et est donc plus rapide pour créer l'application
- Maintenabilité facilité, de part le peu de code à implémenter la maintenance s'en trouve simplifiée et l'optimisation aussi
- Les performances, le compilateur intégré de l'ASP.NET et capable d'améliorer le code lorsque le framework est re compilé avec le code

Ses inconvénients :

- Difficulté d'apprentissage, le C# étant plus orienté vers la programmation d'OS et de logiciels associés
- ASP.NET n'est pas open-source
- Ne fonctionne que sur un serveur Microsoft

Lien vers la page Github de ASP.NET Core

## EXPRESSJS

C'est un framework d'application web Node.js open-source disponible sous licence du MIT. On l'utilise pour construire des API et des applications web.

Il est considéré comme un framework Node.js standard. Node.js en tant que back-end est un JavaScript complet pour les applications côté serveur et côté client.

ExpressJS permet de la programmation rapide côté serveur et implémente le routage de manière plus sophistiqué que Node.js car il permet de gérer des URL dynamiques. Le débogage est grandement facilité par ExpressJS ce qui accroît la productivité, un programme n'étant qu'une suite de bug corrigé comme on aime souvent le dire pour plaisanter.

Ses avantages :

- Apprentissage facile, JavaScript étant un des langages de programmation les plus répandus. Node.js demande peu de temps pour être utilisé

- Un seul langage pour le front-end et le back-end, JavaScript
- Performances élevées et universalité du langage. JavaScript est prit en charge par la majeure partie des navigateurs et compilé de manière efficace

Ses inconvénients :

- La flexibilité et le peu de cadre peuvent engendrer un code difficile à relire et à maintenir
- L'usage des librairies npm, rend la maintenabilité difficile et accroît les risques en terme de sécurité
- Nécessite l'usage de l'asynchrone afin de garantir les performances

Lien vers la doc Node.js

Lien vers la doc ExpressJS

## SYMFONY

C'est un framework open-source utilisant le langage Php qui permet de créer tous types d'application web, de plus c'est un framework français créé par Fabien Potencier. Laravel est son pendant mais plutôt pour la communauté anglophone, il utilise de nombreux composants de Symfony et fonctionne de manière très similaire.

Le Framework en est à sa version 6 qui est sortie en fin d'année 2021, il est orienté MVC, très flexible et possède une barre de débogage intégrale qui facilite le développement et les tests.

De base il utilise le moteur de template Twig afin d'assurer la partie front-end du projet web mais on peut tout à fait utiliser Symfony afin de créer une API et faire communiquer un front-end indépendant réalisé avec un Framework JavaScript.

Symfony fonctionne avec l'ORM Doctrine qui est très performant, de plus il existe des bundles créés et maintenu par l'équipe Sensio Lab tels que easy-admin qui permet d'avoir une administration en quelques lignes de commandes.

Ses avantages :

- Très grande communauté francophone
- Productivité accrue grâce au maker de Symfony et sa cli
- Organisation du code qui facilite le travail en équipe

Ses inconvénients :

- Difficulté d'apprentissage même si facilité par la quantité de ressources
- Évolution constante, il faut pratiquer le Framework au quotidien et suivre son actualité
- Lourd pour la réalisation de petits projets

Lien vers la doc Symfony