**Piste de Documentation API REST ET SYMFONY API PLATFORM**

**1)API REST**

**● Notion d’API**

**● Protocole HTTP (fonctionnement, les verbes)**

**● Architecture Rest basé sur le protocole HTTP**

**● Contrainte Architecture de API REST**

**● API RESTFULL: Modèle de Maturité de Richardson**

**2)Notion de Serialisation**

**● Notion de Decode et Encode**

**● Notion de sérialisation et désérialisation**

**● Notion de normalisation et dénormalisation**

**3)Notion de Fixtures**

**4)Installation Postman pour tester les API**

**5)Bundle API PlatForm**

**● Installation**

**● Référence circulaire**

**● Api Auto-decouvrable**

**● Notion de Tri**

**● Notion de Filter(RangeFilter,SearchFilter, OrderFilter,PropertyFilter)**

**● Pagination**

**● Notion de Api SubRessource**

**● itemsOperation**

**● CollectionsOperation**

**6)Gestion de l’authentification**

**● Notion JWT**

**● Architecture du JWT**

**● Notion Provider**

**● Firewall**

**● Encoder**

**● Access Control**

**7)API Platform et Evenements Doctrine**

**● Evenements Doctrine**

**● Methodes PrePersist,PreUpdate,PreRemove**

**● PreSubscriber**

**8)API Platform et Controllers Personnalisés**

[**API Platform**](https://api-platform.com/) est un framework de **pile complète** puissant mais facile à utiliser dédié aux projets pilotés par API. Il contient une bibliothèque **PHP** pour créer des API complètes prenant en charge les normes leaders de l'industrie ( [**JSON-LD**](https://json-ld.org/) et [**Hydra**](https://www.hydra-cg.com/) , **[GraphQL](https://graphql.org/" \t "_blank)** , **[OpenAPI](https://www.openapis.org/" \t "_blank)** ...), fournit des outils **JavaScript** ambitieux pour consommer ces API en un clin d'œil (admin, PWA et générateurs d'applications mobiles, client hypermédia ...) et est livré avec une belle intégration [**Docker**](https://www.docker.com/) et **[Kubernetes](https://kubernetes.io/" \t "_blank)** pour développer et déployer instantanément sur le cloud.

**Notion d’API**

API signifie Application Programming Interface. Le mot le plus important est “interface”, et c’est le mot le plus simple, car nous utilisons tous des interfaces au quotidien.

Nous utilisons beaucoup d'interfaces sans nous en rendre compte. Les sites web et les applications ont besoin de la même chose pour communiquer et échanger des données. C’est là que vient le “AP” d’API : “Application Programming” Interface. Une API est une interface pour les applications, car un logiciel n’a pas de mains ni d’yeux pour interagir avec les interfaces physiques !

**● Protocole HTTP (fonctionnement, les verbes)**

L'***Hypertext Transfer Protocol*** (**HTTP**, littéralement « protocole de transfert [hypertexte](https://fr.wikipedia.org/wiki/Hypertexte) ») est un [protocole de communication](https://fr.wikipedia.org/wiki/Protocole_de_communication) [client-serveur](https://fr.wikipedia.org/wiki/Client-serveur) développé pour le [*World Wide Web*](https://fr.wikipedia.org/wiki/World_Wide_Web). [HTTPS](https://fr.wikipedia.org/wiki/HyperText_Transfer_Protocol_Secure) (avec S pour *secured*, soit « sécurisé ») est la variante du HTTP *sécurisée* par l'usage des [protocoles](https://fr.wikipedia.org/wiki/Protocole_de_communication) [SSL](https://fr.wikipedia.org/wiki/Transport_Layer_Security) ou [TLS](https://fr.wikipedia.org/wiki/Transport_Layer_Security).

HTTP est un protocole de la [couche application](https://fr.wikipedia.org/wiki/Couche_application). Il peut fonctionner sur n'importe quelle connexion fiable, dans les faits on utilise le protocole [TCP](https://fr.wikipedia.org/wiki/Transmission_Control_Protocol) comme couche de transport. Un [serveur HTTP](https://fr.wikipedia.org/wiki/Serveur_HTTP) utilise alors par défaut le [port](https://fr.wikipedia.org/wiki/Port_(logiciel)) 80 (443 pour HTTPS).

Les [clients HTTP](https://fr.wikipedia.org/wiki/Client_HTTP) les plus connus sont les [navigateurs Web](https://fr.wikipedia.org/wiki/Navigateur_web) permettant à un utilisateur d'accéder à un serveur contenant les données. Il existe aussi des systèmes pour récupérer automatiquement le contenu d'un site tel que les [aspirateurs de site Web](https://fr.wikipedia.org/wiki/Aspirateur_de_site_Web) ou les [robots d'indexation](https://fr.wikipedia.org/wiki/Robot_d%27indexation).

**● Contrainte Architecture de API REST**

les API “REST” sont propres et polyvalentes pour beaucoup de types d’échange d’information entre logiciels. REST signifie “Representational State Transfer”.

Le standard REST a été créé en 2000 par Roy Fielding dans sa thèse "Architectural Styles and the Design of Network-based Software Architectures".

### **les verbes HTTP comme identifiant des opérations**

La seconde règle d’une architecture REST est d’utiliser les verbes HTTP existants plutôt que d’inclure l’opération dans l’URI de la ressource. Ainsi, généralement pour une ressource, il y a 4 opérations possibles (CRUD) :

* Créer (create)
* Afficher (read)
* Mettre à jour (update)
* Supprimer (delete)

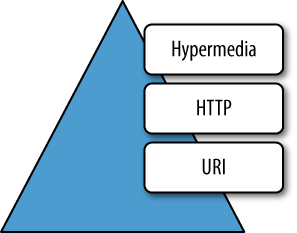
HTTP propose les verbes correspondant :

* Créer (create) => **POST**
* Afficher (read) => **GET**
* Mettre à jour (update) => **PUT**
* Supprimer (delete) => **DELETE**

**●Modèle de maturité Richardson**

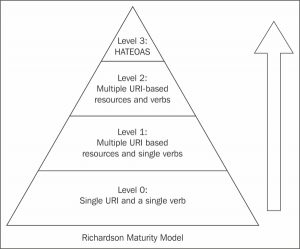
Leonard Richardson a analysé une centaine de conceptions de services Web différentes et les a divisées en quatre catégories en fonction de leur conformité REST. Ce modèle de division des services REST pour identifier leur niveau de maturité - est appelé **Richardson Maturity Model** .

Richardson a utilisé trois facteurs pour décider de la maturité d'un service, à savoir l' [URI](https://restfulapi.net/resource-naming/) , [les méthodes HTTP](https://restfulapi.net/http-methods/) et [HATEOAS](https://restfulapi.net/hateoas/) ( [Hypermedia](https://restfulapi.net/hateoas/) ). Plus un service utilise ces technologies, plus il doit être mûr.

Les niveaux de maturité selon le modèle de Richardson

Dans cette analyse, Richardson a décrit ces niveaux de maturité comme suit:

* [Niveau zéro](https://restfulapi.net/richardson-maturity-model/#level-zero)
* [Premier niveau](https://restfulapi.net/richardson-maturity-model/#level-one)
* [Niveau deux](https://restfulapi.net/richardson-maturity-model/#level-two)
* [Niveau trois](https://restfulapi.net/richardson-maturity-model/#level-three)

Modèle de maturité Richardson

**Niveau zéro**

Le niveau zéro de maturité n'utilise aucune des fonctionnalités URI, Méthodes HTTP et HATEOAS.

Ces services ont un seul URI et utilisent une seule méthode HTTP (généralement POST). Par exemple, la plupart des services basés sur les services Web (WS - \*) utilisent un seul URI pour identifier un point de terminaison et HTTP POST pour transférer des charges utiles basées sur SOAP, ignorant efficacement le reste des verbes HTTP.

De même, les services basés sur XML-RPC qui envoient des données en tant que XML ancien simple (POX). Il s'agit de la manière la plus primitive de créer des applications SOA avec une seule méthode POST et d'utiliser XML pour communiquer entre les services.

**Premier niveau**

Le premier niveau de maturité **utilise des URI** hors URI, des méthodes HTTP et HATEOAS.

Ces services utilisent de nombreux URI mais un seul verbe HTTP - généralement HTTP POST. Ils donnent à chaque ressource individuelle de leur univers un URI. Chaque ressource est identifiée séparément par un URI unique - et cela les rend meilleures que le niveau zéro.

**Niveau deux**

Le niveau de maturité deux **utilise les URI et HTTP** hors URI, les méthodes HTTP et HATEOAS.

Les services de niveau deux hébergent de nombreuses ressources adressables par URI. Ces services prennent en charge plusieurs des verbes HTTP sur chaque ressource exposée - services CRUD (Create, Read, Update and Delete). Ici, l'état des ressources, représentant généralement des entités commerciales, peut être manipulé sur le réseau.

Ici, le concepteur de services s'attend à ce que les gens mettent un certain effort à maîtriser les API - généralement en lisant la documentation fournie.

Le niveau 2 est le bon cas d'utilisation des principes REST, qui préconisent l'utilisation de différents verbes basés sur les méthodes de requête HTTP et le système peut avoir plusieurs ressources.

**Niveau trois**

Le niveau de maturité trois **utilise les trois URI et HTTP et HATEOAS** .

Il s'agit du niveau le plus mature du modèle de Richardson qui encourage la découverte facile et permet aux réponses de s'expliquer facilement en utilisant HATEOAS.

Le service guide les consommateurs à travers une traînée de ressources, provoquant des transitions d'état d'application en conséquence.

**2)Notion de Serialisation**

En [informatique](https://fr.wikipedia.org/wiki/Informatique), la **sérialisation** (de l'[anglais américain](https://fr.wikipedia.org/wiki/Anglais_am%C3%A9ricain) *serialization*) est le codage d'une information sous la forme d'une **suite** d'informations plus petites (dites *atomiques*, voir l'étymologie de [atome](https://fr.wiktionary.org/wiki/atome)) pour, par exemple, sa sauvegarde ([persistance](https://fr.wikipedia.org/wiki/Persistance_(informatique))) ou son transport sur le réseau ([proxy](https://fr.wikipedia.org/wiki/Serveur_mandataire), [RPC](https://fr.wikipedia.org/wiki/Remote_procedure_call)…). L'activité réciproque, visant à décoder cette suite pour créer une copie conforme de l'information d'origine, s'appelle la **désérialisation** (ou *unmarshalling*).

Le terme *marshalling* (mobilisation, canalisation, organisation) est souvent employé de façon synonyme, de même que le terme *linéarisation*. Les termes *marshalling* et *unmarshalling* s'emploient le plus souvent dans le contexte d'échanges entre [programmes informatiques](https://fr.wikipedia.org/wiki/Application_informatique), alors que les termes *sérialisation* et *désérialisation* sont plus généraux[1](https://fr.wikipedia.org/wiki/S%C3%A9rialisation#cite_note-1).

D'apparence simple, ces opérations posent en réalité un certain nombre de problèmes, comme la gestion des références entre [objets](https://fr.wikipedia.org/wiki/Objet_(informatique)) ou la portabilité des encodages. Par ailleurs, les choix entre les diverses techniques de sérialisation ont une influence sur les critères de performances comme la taille des suites d'octets sérialisées ou la vitesse de leur traitement.

**● Notion de Decode et Encode**

## Encodage[[modifier](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&veaction=edit&section=2) | [modifier le code](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&action=edit&section=2)]

Le choix de base est entre [format binaire](https://fr.wikipedia.org/wiki/Fichier_binaire) et [format texte](https://fr.wikipedia.org/wiki/Fichier_texte) :

* les fichiers binaires sont généralement plus compacts, le code pour *[parser](https://fr.wikipedia.org/wiki/Parser" \o "Parser)* ce type de données est plus simple à mettre au point, la lecture et l'écriture sont moins exigeantes en ressources processeurs ;
* les fichiers textes sont plus simples à vérifier ou à modifier manuellement, ils posent moins de problèmes de portabilité, ils sont plus simples à maintenir et à faire évoluer en fonction des besoins.

### Codages binaires **:**

L'une des contraintes des codages binaires est la portabilité. Par exemple une machine utilisant un autre modèle de processeur que l'ordinateur d'origine doit pouvoir désérialiser un bloc de données, en prenant en compte les problèmes d'[alignement de données](https://fr.wikipedia.org/wiki/Alignement_de_donn%C3%A9es) et d’*[endianness](https://fr.wikipedia.org/wiki/Endianness" \o "Endianness)*. C'est pourquoi, même si l'objet ne comporte pas de pointeurs, la simple copie de l'empreinte mémoire d'un objet n'est généralement pas une solution acceptable.

Il faut donc ici aussi utiliser des encodages conventionnels. Il est assez courant d'utiliser les conventions suivantes : aucun alignement; encodage des [types C](https://fr.wikipedia.org/wiki/Langage_C#Types) entiers en fonction de leur empreinte mémoire tous au format *big-endian*, les nombres en virgule flottante utilisent la norme [IEEE 754](https://fr.wikipedia.org/wiki/IEEE_754).

Des protocoles comme [GIOP](https://fr.wikipedia.org/wiki/IIOP) de [CORBA](https://fr.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture) ou [RMI](https://fr.wikipedia.org/wiki/Remote_method_invocation_(Java)) de [Java](https://fr.wikipedia.org/wiki/Java_(technologie)) emploient tous deux des codages binaires.

### Codages textes **:**

Définir un codage textuel nécessite de choisir un protocole pour séparer les champs, pour encoder des données binaires (par exemple [uuencode](https://fr.wikipedia.org/wiki/Uuencode" \o "Uuencode), [base64](https://fr.wikipedia.org/wiki/Base64) ou échappement des caractères non [ASCII](https://fr.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange))…

Il était relativement courant d'utiliser un dérivé du format [XML](https://fr.wikipedia.org/wiki/Extensible_Markup_Language). En 2019, le format de codage texte le plus courant est le format [JSON](https://fr.wikipedia.org/wiki/JavaScript_Object_Notation)

Les protocoles basés sur des fichiers textes XML sont [SOAP](https://fr.wikipedia.org/wiki/Simple_Object_Access_Protocol), [FIX](https://fr.wikipedia.org/w/index.php?title=Financial_Information_eXchange&action=edit&redlink=1) et [XML-RPC](https://fr.wikipedia.org/wiki/XML-RPC). Le format binaire de SOAP a pour nom [Common Data Representation](https://fr.wikipedia.org/w/index.php?title=Common_Data_Representation&action=edit&redlink=1)

La sérialisation [PHP](https://fr.wikipedia.org/wiki/PHP) utilise un format spécifique.

**● Notion de sérialisation et désérialisation**

### Sérialisation et encodage sur internet**[**[**modifier**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&veaction=edit&section=5)**|**[**modifier le code**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&action=edit&section=5)**]**

Le transfert de fichier texte avec l'apparition de l'internet a laissé place à des protocoles client/Serveur gérant le transfert de données sous forme de classes. Les anciens clients avaient des [*cookies*](https://fr.wikipedia.org/wiki/Cookie_(informatique)) dont la taille et l'origine étaient limités. Les objets sont l'évolution des *cookies* et peuvent ou non être sauvegardés dans l'espace de travail du navigateur web.

* AMF (*Action Message Format*) est utilisé dans de nombreux projets libres et commerciaux. Différentes implémentations existent : [PHP](https://fr.wikipedia.org/wiki/PHP:_Hypertext_Preprocessor)[2](https://fr.wikipedia.org/wiki/S%C3%A9rialisation#cite_note-2), [Java](https://fr.wikipedia.org/wiki/Java_(langage))[3](https://fr.wikipedia.org/wiki/S%C3%A9rialisation#cite_note-3),[4](https://fr.wikipedia.org/wiki/S%C3%A9rialisation#cite_note-4), PErl[5](https://fr.wikipedia.org/wiki/S%C3%A9rialisation#cite_note-5). AMF est utilisé par flash pour son serveur média et s'interface facilement avec les objets java. AMF 0 apparaît en 2001 dans le *Player Flash 6* (as 1.0) et évolue en AMF 3.0 avec le *Player 9* (as 3.0). AMF est un format d'échange binaire mettant en jeu des vidéos Flash ou un serveur et un client Flash.
* [*Google Gears*](https://fr.wikipedia.org/wiki/Google_Gears) est un plug in [AJAX](https://fr.wikipedia.org/wiki/Asynchronous_JavaScript_and_XML) pour navigateur web. Il permet de façon transparente de sauvegarder des données localement dans une base de données *SQLite* durant une connexion internet. Ces données pourront être utilisées en mode non connecté. Il est fourni [par défaut](https://fr.wikipedia.org/wiki/Par_d%C3%A9faut) avec [Google Chrome](https://fr.wikipedia.org/wiki/Google_Chrome). Les services web en ligne [*Google Reader*](https://fr.wikipedia.org/wiki/Google_Reader) et [*Remember the Milk*](https://fr.wikipedia.org/w/index.php?title=Remember_the_Milk&action=edit&redlink=1) sont compatibles Google Gears
* EOF et WOF sont des objets utilisés par le serveur java [WebObjects](https://fr.wikipedia.org/wiki/WebObjects" \o "WebObjects) d'[Apple](https://fr.wikipedia.org/wiki/Apple) pour une visualisation html. WOF (*WebObjects Framework*) s'occupe du rendu html tandis que EOF ([*Enterprise Objects Framework*](https://fr.wikipedia.org/wiki/Enterprise_Objects_Framework)) simplifie l'accès aux bases de données. L'abandon de webobjects et la gratuité a permis à Apple de se concentrer sur le moteur, et de voir naître des outils open source (Eclipse IDE avec plug in WOLIPs).
* WCF de [Microsoft](https://fr.wikipedia.org/wiki/Microsoft) permet de sérialiser des données par plusieurs objets (Xml DataContract netDataContract). Après la sérialisation, l'encodage se fait dans différents formats (Binary, Text, MTOM). Le client Silverlight support WCF.
* *zodb* dans *zope* utilise le module *pickle* de [python](https://fr.wikipedia.org/wiki/Python_(langage)) pour sérialiser des objets côté serveur. Rien n'existe côté client où les objets sont rendus visualisables (html, flash, pdf…).
* [COM](https://fr.wikipedia.org/wiki/Component_Object_Model), [Rewire](https://fr.wikipedia.org/wiki/Rewire" \o "Rewire) permet l'échange de données entre deux applications sans sauvegarde ; il n'y a pas sérialisation.

## Sérialisation d'un objet atomique[[modifier](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&veaction=edit&section=6) | [modifier le code](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&action=edit&section=6)]

Un objet atomique est un objet qui ne comporte aucune référence vers d'autres objets.

### Type de l'objet**[**[**modifier**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&veaction=edit&section=7)**|**[**modifier le code**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&action=edit&section=7)**]**

Selon les possibilités du langage, la réanimation pourra utiliser un mécanisme de [métaclasse](https://fr.wikipedia.org/wiki/M%C3%A9taclasse" \o "Métaclasse) apporté par le langage, ou une [fabrique](https://fr.wikipedia.org/w/index.php?title=Fabrique_(informatique)&action=edit&redlink=1) spécifique. Dans tous les cas, il est nécessaire de conserver les informations qui permettront de sélectionner le type d'objet à créer.

Si le nombre de types d'objets à sérialiser est connu à l'avance, les informations de type peuvent être codées de manière très compacte (par exemple, sur un simple octet si celles-ci n'excèdent pas 256).

Sinon, il sera nécessaire d'utiliser des conventions, comme celle des packages du langage [Java](https://fr.wikipedia.org/wiki/Java_(langage)). Ces noms conventionnels pouvant être volumineux, il pourra être utile de prévoir un mécanisme d'alias pour éviter les répétitions lors du traitement de plusieurs objets du même type.

Il est aussi possible de transmettre directement le code mettant en œuvre le type encodé. C'est le cas par exemple du module *marshal* de la bibliothèque standard du langage Python, et c'est un mécanisme qui est plus généralement supporté par tous les langages interprétés supportant la mise en cache de leur [byte code](https://fr.wikipedia.org/wiki/Bytecode).

### Données**[**[**modifier**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&veaction=edit&section=8)**|**[**modifier le code**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&action=edit&section=8)**]**

Chaque type de donnée est responsable de l'archivage et de la restauration de ses données membres. Pour les types composites, il s'agit de sérialiser chacun des champs dans un ordre prédéfini.

### Types hiérarchiques**[**[**modifier**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&veaction=edit&section=9)**|**[**modifier le code**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&action=edit&section=9)**]**

En [programmation orientée objet](https://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_objet), il est nécessaire de traiter les données gérées par le type de base avant d'archiver les données du type dérivé.

## Parcours d'un graphe d'objets[[modifier](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&veaction=edit&section=10) | [modifier le code](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&action=edit&section=10)]

C'est une problématique qui est assez commune et que l'on retrouve par exemple quand on cherche à mettre en œuvre un [clonage](https://fr.wikipedia.org/wiki/Clonage_(informatique)) ou un [ramasse-miettes](https://fr.wikipedia.org/wiki/Ramasse-miettes_(informatique)).

Des algorithmes plus ou moins performants peuvent être choisis suivant les a priori que l'on peut faire sur la topologie du graphe :

* arbre ;
* graphe uniquement connecté aux feuilles ;
* graphe quelconque.

### Parcours manuel**[**[**modifier**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&veaction=edit&section=11)**|**[**modifier le code**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&action=edit&section=11)**]**

Dans le cas général il est nécessaire de mémoriser les objets parcourus pour détecter les cycles.

Ce n'est pas une bonne idée d'utiliser les objets eux-mêmes pour pointer leur statut visité :

* la méthode de sérialisation se met à muter les objets, ce qui perturbe les optimisations de type [*Copy-On-Write*](https://fr.wikipedia.org/wiki/Copy-On-Write) utilisé par exemple lors d'un [*fork*](https://fr.wikipedia.org/wiki/Fork_(programmation)) sous UNIX ;
* cela pose des problèmes de réentrance.

Il est préférable d'utiliser une [table de hachage](https://fr.wikipedia.org/wiki/Table_de_hachage) (adresse de l'objet, compteur) qui sera par ailleurs utilisée pour mettre en œuvre la mutation de pointeur. Il faut alors veiller à retenir chaque objet pour éviter les collisions d'adresses avec de nouveaux objets.

### Parcours par introspection**[**[**modifier**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&veaction=edit&section=12)**|**[**modifier le code**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&action=edit&section=12)**]**

Les langages qui supportent l'[introspection](https://fr.wikipedia.org/wiki/Introspection_(informatique)) peuvent fournir un mécanisme de sérialisation par défaut.

## Désérialisation[[modifier](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&veaction=edit&section=13) | [modifier le code](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&action=edit&section=13)]

La désérialisation pose également un certain nombre de problèmes comme la réanimation d'un [objet immuable](https://fr.wikipedia.org/wiki/Objet_immuable). Les objets ne peuvent pas être utilisés pendant la désérialisation.

La désérialisation pose aussi des problèmes de [sûreté du typage](https://fr.wikipedia.org/wiki/S%C3%BBret%C3%A9_du_typage).

## Sécurité[[modifier](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&veaction=edit&section=14) | [modifier le code](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&action=edit&section=14)]

Pour éviter les problèmes de sécurité, la sérialisation, et surtout la désérialisation doivent être utilisés avec une grande prudence.

### Désérialisations de données non fiable**[**[**modifier**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&veaction=edit&section=15)**|**[**modifier le code**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&action=edit&section=15)**]**

Un attaquant contrôlant les données fournies à une application puis désérialisées peut, de fait, injecter ses propres objets dans l'application. En fonction des classes disponibles et des méthodes appelées sur les données désérialisées, les risques vont de la divulgation de données sensibles à l'exécution de code arbitraire[6](https://fr.wikipedia.org/wiki/S%C3%A9rialisation#cite_note-6).

Les vulnérabilités de ce type sont regroupées dans la catégorie CWE-502[7](https://fr.wikipedia.org/wiki/S%C3%A9rialisation#cite_note-7). Bien que moins nombreuses que d'autres types de vulnérabilités, on en comptait quand même 252 publiées début 2020[8](https://fr.wikipedia.org/wiki/S%C3%A9rialisation#cite_note-8).

### Sérialisation de données sensibles**[**[**modifier**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&veaction=edit&section=16)**|**[**modifier le code**](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&action=edit&section=16)**]**

Réciproquement, un attaquant pouvant obtenir la désérialisation de classes contenant des données sensibles pourrait en obtenir la divulgation. Ces classes doivent donc interdire leur sérialisation ou, au moins, protéger les données sensibles en surchargeant les méthodes correspondantes.

Ces vulnérabilités sont regroupées dans la catégorie CWE-499[9](https://fr.wikipedia.org/wiki/S%C3%A9rialisation#cite_note-9).

## Gestion des versions[[modifier](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&veaction=edit&section=17) | [modifier le code](https://fr.wikipedia.org/w/index.php?title=S%C3%A9rialisation&action=edit&section=17)]

Il est souvent nécessaire de garantir une [compatibilité ascendante ou descendante](https://fr.wikipedia.org/wiki/Compatibilit%C3%A9_ascendante_et_descendante), c’est-à-dire la possibilité de relire ses données avec une nouvelle version du logiciel ou de permettre à une ancienne version du logiciel de lire des données créées à partir d'une version plus récente. Cela nécessite d'une part un mode de versionnage qui permet de connaître les versions compatibles et un moyen pour les versions les plus anciennes d'ignorer les données qu'elle ne savent pas interpréter.

La normalisation et la dénormalisation sont les méthodes utilisées dans les bases de données. Les termes sont différenciables lorsque la **normalisation** est une technique permettant de minimiser les anomalies d'insertion, de suppression et de mise à jour en éliminant les données redondantes. D'autre part, la **dénormalisation** est le processus inverse de la normalisation dans lequel la redondance est ajoutée aux données afin d'améliorer les performances de l'application spécifique et l'intégrité des données.

La normalisation évite le gaspillage d'espace disque en minimisant ou en éliminant la redondance.

### Tableau de comparaison

|  |  |  |
| --- | --- | --- |
| **Base de comparaison** | **Normalisation** | **Dénormalisation** |
| De base | La normalisation est le processus de création d'un schéma d'ensemble pour stocker des données cohérentes et non redondantes. | La dénormalisation est le processus consistant à combiner les données afin de pouvoir les interroger rapidement. |
| Objectif | Réduire la redondance et l'incohérence des données. | Pour accélérer l'exécution des requêtes en introduisant la redondance. |
| Utilisé dans | Le système OLTP, où l’accent est mis sur l’accélération de l’insertion, la suppression et la mise à jour des anomalies et le stockage des données de qualité. | Système OLAP, où l’accent est mis sur l’accélération de la recherche et de l’analyse. |
| Intégrité des données | Entretenu | Peut ne pas retenir |
| Redondance | Éliminé | Ajoutée |
| Nombre de tables | Augmente | Des diminutions |
| Espace disque | Utilisation optimisée | Gaspillage |

### Définition de la normalisation

**La normalisation** est la méthode pour organiser efficacement les données dans la base de données. Cela implique la construction de tables et la mise en place de relations entre ces tables en fonction de certaines règles. La redondance et la dépendance incohérente peuvent être supprimées à l'aide de ces règles afin de le rendre plus flexible.

Les données redondantes gaspillent de l'espace disque, augmentent l'incohérence des données et ralentissent les requêtes DML. Si les mêmes données sont présentes dans plusieurs emplacements et qu'une mise à jour est validée pour ces données, la modification doit être reflétée dans tous les emplacements. Des données incohérentes peuvent rendre la recherche et l'accès aux données plus difficiles en perdant le chemin d'accès.

La normalisation a plusieurs raisons, par exemple pour éviter la redondance, la mise à jour des anomalies, le codage inutile, la conservation des données dans un format permettant une modification plus simple et plus précise et l’application de la contrainte de données.

La normalisation comprend l'analyse des dépendances fonctionnelles entre les attributs. Les relations (tables) sont **décomposées** avec des anomalies pour générer des relations avec une structure. Cela aide à décider quels attributs doivent être regroupés dans une relation.

La normalisation est fondamentalement basée sur les concepts de **formes normales** . Une table de relations est dite de forme normale si elle remplit un certain ensemble de contraintes. Il existe 6 formes normales définies: 1NF, 2NF, 3NF, BCNF, 4NF et 5NF. La normalisation devrait éliminer la redondance, mais pas au détriment de l'intégrité.

### **Définition de la dénormalisation**

**La dénormalisation** est le processus inverse de la normalisation, dans lequel le schéma normalisé est converti en un schéma contenant des informations redondantes. Les performances sont améliorées en utilisant la redondance et en maintenant les données redondantes cohérentes. La dénormalisation s'explique par les **frais généraux** générés dans le processeur de requêtes par une structure sur-normalisée.

La dénormalisation peut également être définie comme la méthode de stockage de la jointure des relations de forme normale supérieure en tant que relation de base, qui est sous une forme normale inférieure. Cela réduit le nombre de tables et les jointures de table complexes, car un nombre plus élevé de jointures peut ralentir le processus. Il existe différentes techniques de dénormalisation telles que: stockage des valeurs dérivables, tables de pré-jointure, valeurs codées en dur et conservation des détails avec le maître, etc.

Ici, l’approche de dénormalisation met l’accent sur le concept selon lequel, en plaçant toutes les données en un seul endroit, il n’est plus nécessaire de rechercher ces multiples fichiers pour collecter ces données. La stratégie de base suivie dans la dénormalisation consiste à sélectionner le processus le plus décisif pour examiner les modifications qui amélioreront les performances. Et la modification la plus élémentaire consiste à ajouter plusieurs attributs à la table existante afin de réduire le nombre de jointures.

## Principales différences entre normalisation et dénormalisation

1. La normalisation est la technique consistant à diviser les données en plusieurs tables afin de réduire la redondance et l'incohérence des données et d'assurer leur intégrité. D'autre part, la dénormalisation est la technique qui consiste à combiner les données dans une seule table pour accélérer l'extraction des données.
2. La normalisation est utilisée dans le système **OLTP**, qui met l'accent sur l'accélération de l'insertion, de la suppression et de la mise à jour des anomalies. Par contre, la dénormalisation est utilisée dans le système **OLAP**, qui met l'accent sur l'accélération de la recherche et de l'analyse.
3. L'intégrité des données est maintenue dans le processus de normalisation tandis que dans la dénormalisation, l'intégrité des données est plus difficile à conserver.
4. Les données redondantes sont éliminées lors de la normalisation tandis que la dénormalisation augmente les données redondantes.
5. La normalisation augmente le nombre de tables et de jointures. En revanche, la dénormalisation réduit le nombre de tables et de jointures.
6. La dénormalisation gaspille de l’espace disque car les mêmes données sont stockées à des emplacements différents. Au contraire, l’espace disque est optimisé dans une table normalisée.

L'application [Postman](http://www.getpostman.com/" \t "_blank) est un outil pratique pour tester une API REST dans API Gateway. Les instructions suivantes vous guident dans les principales étapes d'utilisation de l'application Postman pour appeler une API. Pour plus d'informations, consultez [l'aide](https://www.getpostman.com/docs/" \t "_blank) Postman.

**4)Installation Postman pour tester les API**

1. Lancez Postman.
2. Saisissez l'URL de point de terminaison d'une demande dans la barre d'adresse et sélectionnez la méthode HTTP appropriée dans la liste déroulante à gauche de la barre d'adresse.
3. Si nécessaire, sélectionnez l'onglet **Authorization**. Sélectionnez le type d'autorisation **AWS Signature** dans le champ **Type**. Saisissez l'ID de clé d'accès de l'utilisateur AWS IAM dans le champ de saisie **AccessKey**. Saisissez la clé secrète de l'utilisateur IAM dans le champ **SecretKey**. Spécifiez une région AWS appropriée correspondant à la région spécifiée dans l'URL d'appel. Saisissez **execute-api** dans le champ **Service Name (Nom du service)**.
4. Sélectionnez l'onglet **Headers**. Supprimez éventuellement tous les en-têtes existants. Cela peut effacer des paramètres obsolètes susceptibles de provoquer des erreurs. Ajoutez tous les en-têtes personnalisés requis. Par exemple, si les clés d'API sont activées, vous pouvez définir ici la paire nom-valeur **x-api-key:***{api\_key}*.
5. Sélectionnez **Send** pour envoyer la demande et recevoir une réponse.

Pour obtenir un exemple d'utilisation de Postman, consultez [Appelez une API avec les mécanismes d'autorisation API Gateway Lambda](https://docs.aws.amazon.com/fr_fr/apigateway/latest/developerguide/call-api-with-api-gateway-lambda-authorization.html).

5)Bundle API PlatForm

API Platform Core est livré avec un pont pour **[FOSUserBundle](https://github.com/FriendsOfSymfony/FOSUserBundle" \t "_blank)** . Si l'ensemble FOSUser est activé, ce pont l'utilisera UserManagerpour créer, mettre à jour et supprimer des ressources utilisateur.

Remarque: FOSUserBundle n'est pas bien adapté aux API. Nous vous encourageons fortement à utiliser le [**fournisseur d'utilisateurs Doctrine**](https://symfony.com/doc/current/security/user_provider.html#entity-user-provider) fourni avec Symfony ou à [**créer un fournisseur d'utilisateurs personnalisé**](https://symfony.com/doc/current/security/user_provider.html#creating-a-custom-user-provider) au lieu d'utiliser cet ensemble.

## Installation du bundle

La procédure d'installation de FOSUserBundle est décrite [**dans les principaux documents Symfony**](https://symfony.com/doc/master/bundles/FOSUserBundle/index.html)

Vous pouvez:

* Ignorez l' [**étape 3 (créer votre classe d'utilisateurs)**](https://symfony.com/doc/master/bundles/FOSUserBundle/index.html#step-3-create-your-user-class) et utilisez la classe fournie dans le paragraphe suivant pour configurer les groupes de sérialisation de la bonne manière
* Ignorez l' [**étape 4 (configurer security.yml de votre application)**](https://symfony.com/doc/master/bundles/FOSUserBundle/index.html#step-4-configure-your-application-s-security-yml) si vous prévoyez d' [**utiliser une authentification basée sur JWT à l'aide deLexikJWTAuthenticationBundle**](https://api-platform.com/docs/core/jwt/)

Si vous utilisez l'API Platform Standard Edition, vous devrez activer les services de formulaire dans les options de configuration du framework symfony:

# api/config/packages/framework.yaml

framework:

form: { enabled: true }

## Activation du pont

Pour activer le pont fourni avec FOSUserBundle, vous devez ajouter la configuration suivante à API Platform:

# api/config/packages/api\_platform.yaml

api\_platform:

enable\_fos\_user: true

## Création d'une Userentité avec des groupes de sérialisation

Voici un exemple de déclaration d'une [**classe d'utilisateurs Doctrine ORM**](https://github.com/FriendsOfSymfony/FOSUserBundle/blob/master/Resources/doc/index.rst#a-doctrine-orm-user-class) . Il y a aussi un exemple pour un [**ODM Doctrine MongoDB**](https://github.com/FriendsOfSymfony/FOSUserBundle/blob/master/Resources/doc/index.rst#b-mongodb-user-class) . Vous devez utiliser des groupes de sérialisation pour masquer certaines propriétés comme plainPassword(uniquement en lecture) et password. Les propriétés affichées sont gérées avec **[normalization\_context](https://api-platform.com/docs/core/serialization/" \l "normalization)**, tandis que les propriétés que vous pouvez modifier sont gérées avec **[denormalization\_context](https://api-platform.com/docs/core/serialization/" \l "denormalization)**.

Créez votre entité utilisateur avec des groupes de sérialisation:

<?php

// api/src/Entity/User.php

namespace App\Entity;

use ApiPlatform\Core\Annotation\ApiResource;

use Doctrine\ORM\Mapping as ORM;

use FOS\UserBundle\Model\User as BaseUser;

use FOS\UserBundle\Model\UserInterface;

use Symfony\Component\Serializer\Annotation\Groups;

/\*\*

\* @ORM\Entity

\* @ORM\Table(name="fos\_user")

\* @ApiResource(

\* normalizationContext={"groups"={"user", "user:read"}},

\* denormalizationContext={"groups"={"user", "user:write"}}

\* )

\*/

class User extends BaseUser

{

/\*\*

\* @ORM\Id

\* @ORM\Column(type="integer")

\* @ORM\GeneratedValue(strategy="AUTO")

\*/

protected $id;

/\*\*

\* @Groups({"user"})

\*/

protected $email;

/\*\*

\* @ORM\Column(type="string", length=255, nullable=true)

\* @Groups({"user"})

\*/

protected $fullname;

/\*\*

\* @Groups({"user:write"})

\*/

protected $plainPassword;

/\*\*

\* @Groups({"user"})

\*/

protected $username;

public function setFullname(?string $fullname): void

{

$this->fullname = $fullname;

}

public function getFullname(): ?string

{

return $this->fullname;

}

public function isUser(?UserInterface $user = null): bool

{

return $user instanceof self && $user->id === $this->id;

}

}

## [Configuration du projet](https://symfonycasts.com/screencast/api-platform/install#project-setup)

Quoi qu'il en soit, pour devenir le héros de l'API dont nous avons tous besoin, vous devez totalement coder avec moi en téléchargeant le code du cours à partir de cette page. Après l'avoir décompressé, vous trouverez un start/répertoire à l'intérieur avec le même code que vous voyez ici ... qui n'est en fait qu'un nouveau projet de squelette Symfony 4.2: il n'y a rien de spécial installé ou configuré pour le moment. Suivez le README.mdfichier pour les instructions de configuration.

La dernière étape sera d'ouvrir un terminal, de se déplacer dans le projet et de démarrer le serveur Symfony avec:

symfony serve -d

Cela utilise l' symfonyexécutable - un petit outil de développement génial que vous pouvez obtenir sur <https://symfony.com/download> . Cela démarre un serveur Web sur le port 8000 qui s'exécute en arrière-plan. Ce qui signifie que nous pouvons trouver notre navigateur, localhost:8000voir et voir ... enfin, pratiquement rien! Juste la belle page d'accueil que vous voyez dans une application Symfony vide.

## [Installation de la plate-forme API](https://symfonycasts.com/screencast/api-platform/install#installing-api-platform)

Maintenant que nous avons notre application Symfony vide, comment pouvons-nous installer API Platform? Oh, c'est tellement génial. Trouvez votre terminal et lancez:

composer require api

C'est tout. Vous remarquerez que cela installe quelque chose appelé api-platform/api-pack. Si vous vous souvenez de notre série Symfony, un "pack" est une sorte de "fausse" bibliothèque qui aide à installer plusieurs choses à la fois.

Heck, vous pouvez le voir sur https://github.com/api-platform/api-pack: c'est un composer.jsonfichier unique qui nécessite plusieurs bibliothèques, comme Doctrine, un bundle CORS dont nous parlerons plus tard, des annotations, la plate-forme API elle-même et quelques parties de Symfony, comme le système de validation, le composant de sécurité et même twig, qui est utilisé pour générer une documentation vraiment cool que nous verrons dans une minute.

Mais, il n'y a rien qui intéressant encore: tout simplement API plate - forme et quelques paquets standards Symfony.

De retour au terminal, c'est fait! Et a quelques détails sur la façon de commencer. Quelques recettes ont également été exécutées qui nous ont donné quelques fichiers de configuration. Avant de faire quoi que ce soit d' autre, revenir au navigateur et la tête pour https://localhost:8000/apivoir ... Woh! Nous avons une documentation API! Eh bien, nous ne même pas a une API encore ... donc il n'y a rien ici. Mais cela va être une énorme fonctionnalité gratuite que vous obtenez avec API Platform: au fur et à mesure que nous construisons notre API, cette page sera automatiquement mise à jour.

● Api Auto-decouvrable

Ce niveau 3 reprend le principe de base du Web : les liens. Le fait de pouvoir se balader d'une page à une autre est rendu plus facile lorsque cette page web affiche à l'utilisateur les liens hypertexte cliquables et accessibles à l'internaute. Eh bien, il s'agit de faire exactement la même chose pour votre API ! Nous allons donc fournir aux utilisateurs de notre API les liens hypertexte des ressources qu'il est possible d'obtenir au fur et à mesure.

Commençons par indiquer ce qui est possible de faire avec un article grâce l'API : le modifier et le supprimer. Nous allons donc ajouter les liens pour que les utilisateurs aient accès à l'information directement.

### HATEOAS - Hypermedia As The Engine Of Application State

La représentation des informations peut se faire de différentes manières. Des standards ont été mis en place par des collectifs indépendants.

Nous n'allons pas faire le tour de tous les standards. Il s'agit de ne se concentrer que sur celui qui a été implémenté par la librairie que nous verrons dans ce cours.

Néanmoins, si le sujet vous intéresse, il existe d'autres standards :

* [**JSON - LD (Linked Data)**](http://json-ld.org/) (implémenté par [**API Platform**](https://api-platform.com/))
* [**SIREN**](https://github.com/kevinswiber/siren)
* [**JSON API**](http://jsonapi.org/format)
* [**Hydra**](http://www.hydra-cg.com/)

Si l'un des standards vous semble plus approprié pour votre API, ne vous privez pas ! Il s'agit de faire en sorte que les informations partagées soient les plus lisibles possible et adoptées de plus. De plus, en fonction des librairies/bundles actuellement sur le marché, vous verrez que les implémentations diffèrent.

Pour faciliter nos développements, nous allons utiliser une librairie ainsi qu'un bundle qui permet l'intégration de cette librairie. J'ai nommé : [BazingaHateoasBundle](https://github.com/willdurand/BazingaHateoasBundle" \t "_blank). Ce bundle intègre la librairie de [William Durand](https://williamdurand.fr/)[Hateoas](https://packagist.org/packages/willdurand/hateoas).

Cette librairie implémente le standard **HAL**.

API Platform Core fournit un système générique pour appliquer des filtres et des critères de tri sur les collections. Des filtres utiles pour Doctrine ORM, MongoDB ODM et ElasticSearch sont fournis avec la bibliothèque.

Vous pouvez également créer des filtres personnalisés qui répondent à vos besoins spécifiques. Vous pouvez également ajouter la prise en charge du filtrage à vos [**fournisseurs de données**](https://api-platform.com/docs/core/data-providers/) personnalisés en implémentant les interfaces fournies par la bibliothèque.

Par défaut, tous les filtres sont désactivés. Ils doivent être activés explicitement.

Lorsqu'un filtre est activé, il apparaît automatiquement dans les documentations **[OpenAPI](https://api-platform.com/docs/core/swagger/)** et **[GraphQL](https://api-platform.com/docs/core/graphql/)** . Il est également documenté automatiquement en tant que hydra:searchpropriété pour les réponses JSON-LD.

### ● Notions de base

Les filtres sont des services, et ils peuvent être liés à une ressource de deux manières

### Filtre de recherche

Si la prise en charge Doctrine ORM ou MongoDB ODM est activée, l'ajout de filtres est aussi simple que l'enregistrement d'un service de filtrage dans le api/config/services.yamlfichier et l'ajout d'un attribut à votre configuration de ressources.

Les supports de filtres exact, partial, start, endet les word\_startstratégies correspondantes:

* partialstratégie utilise LIKE %text%pour rechercher les champs qui contiennent text.
* startla stratégie utilise LIKE text%pour rechercher les champs commençant par text.
* endla stratégie utilise LIKE %textpour rechercher les champs qui se terminent par text.
* word\_startLa stratégie utilise LIKE text% OR LIKE % text%pour rechercher des champs contenant des mots commençant par text.

Ajoutez la lettre iau filtre si vous souhaitez qu'elle soit insensible à la casse. Par exemple ipartialou iexact. Notez que cela utilisera la LOWERfonction et **aura un** impact sur les performances [**s'il n'y a pas d'index approprié**](https://api-platform.com/docs/core/performance/#search-filter) .

L'insensibilité à la casse peut déjà être appliquée au niveau de la base de données en fonction du [**classement**](https://en.wikipedia.org/wiki/Collation) utilisé. Si vous utilisez MySQL, notez que le utf8\_unicode\_ciclassement couramment utilisé (et son frère utf8mb4\_unicode\_ci) sont déjà insensibles à la casse, comme indiqué par la \_cipartie dans leurs noms.

Remarque: Les filtres de recherche avec la exactstratégie peuvent avoir plusieurs valeurs pour la même propriété (dans ce cas, la condition sera similaire à une clause SQL IN).

Syntaxe: ?property[]=foo&property[]=bar

Dans l'exemple suivant, nous verrons comment autoriser le filtrage d'une liste d'offres de commerce électronique:

<?php

// api/src/Entity/Offer.php

namespace App\Entity;

use ApiPlatform\Core\Annotation\ApiResource;

use ApiPlatform\Core\Annotation\ApiFilter;

use ApiPlatform\Core\Bridge\Doctrine\Orm\Filter\SearchFilter;

/\*\*

\* @ApiResource()

\* @ApiFilter(SearchFilter::class, properties={"id": "exact", "price": "exact", "description": "partial"})

\*/

class Offer

{

// ...

}

http://localhost:8000/api/offers?price=10retournera toutes les offres avec un prix exact 10. http://localhost:8000/api/offers?description=shirtrenverra toutes les offres avec une description contenant le mot "chemise".

Les filtres peuvent être combinés ensemble: http://localhost:8000/api/offers?price=10&description=shirt

Il est également possible de filtrer sur les relations, si Offera une Productrelation:

<?php

// api/src/Entity/Offer.php

namespace App\Entity;

use ApiPlatform\Core\Annotation\ApiResource;

use ApiPlatform\Core\Annotation\ApiFilter;

use ApiPlatform\Core\Bridge\Doctrine\Orm\Filter\SearchFilter;

/\*\*

\* @ApiResource()

\* @ApiFilter(SearchFilter::class, properties={"product": "exact"})

\*/

class Offer

{

// ...

}

Avec cette définition de service, il est possible de retrouver toutes les offres appartenant au produit identifié par un IRI donné. Essayez ce qui suit: http://localhost:8000/api/offers?product=/api/products/12. L'utilisation d'un ID numérique est également prise en charge:http://localhost:8000/api/offers?product=12

Les URL ci - dessus renverra toutes les offres pour le produit ayant les éléments suivants IRI comme identifiant JSON-LD ( @id): http://localhost:8000/api/products/12.

### Filtre de date

Le filtre de date permet de filtrer une collection par intervalles de date.

Syntaxe: ?property[<after|before|strictly\_after|strictly\_before>]=value

La valeur peut prendre n'importe quel format de date pris en charge par le [**\DateTimeconstructeur**](https://www.php.net/manual/en/datetime.construct.php) .

Les filtres afteret beforefiltreront en incluant la valeur tandis que strictly\_afteret strictly\_beforefiltreront en excluant la valeur.

Comme les autres filtres, le filtre de date doit être explicitement activé:

<?php

// api/src/Entity/Offer.php

namespace App\Entity;

use ApiPlatform\Core\Annotation\ApiFilter;

use ApiPlatform\Core\Annotation\ApiResource;

use ApiPlatform\Core\Bridge\Doctrine\Orm\Filter\DateFilter;

/\*\*

\* @ApiResource

\* @ApiFilter(DateFilter::class, properties={"createdAt"})

\*/

class Offer

{

// ...

}

Étant donné que le point final de collecte est /offers, vous pouvez filtrer les offres par jour avec la requête suivante: /offers?createdAt[after]=2018-03-19.

Il retournera toutes les offres où createdAtest supérieur ou égal à 2018-03-19.

En haut de notre classe, ajoutez un autre filtre avec PropertyFilter::class. Déplacez-vous vers le haut, tapez use PropertyFilteret appuyez sur l'onglet pour compléter automatiquement. Cette fois, il n'y a qu'une seule de ces classes.

 164 lines  [src/Entity/CheeseListing.php](https://symfonycasts.com/screencast/api-platform/property-filter" \l "codeblock-c2673c06b2" \o "src/Entity/CheeseListing.php)

|  |  |
| --- | --- |
|  | *... lines 1 – 9* |
|  | use ApiPlatform\Core\Serializer\Filter\PropertyFilter; |
|  | *... lines 11 – 15* |
|  | /\*\* |
|  | \* @ApiResource( |
|  | *... lines 18 – 29* |
|  | \* @ApiFilter(PropertyFilter::class) |
|  | *... line 31* |
|  | \*/ |
|  | class CheeseListing |
|  | *... lines 34 – 164* |

Ce filtre *ne* possède quelques options, mais il fonctionne parfaitement bien sans rien faire d' autre.

Allez rafraîchir nos documents. Hmm, cela ne fait aucune différence ici ... ce n'est pas une fonctionnalité de notre API qui peut être exprimée dans le document de spécification OpenAPI.

Mais, cette ressource dans notre API *ne* possède une nouvelle super-puissance. Dans l'autre onglet, choisissez les propriétés exactes avec lesquelles vous souhaitez ?properties[]=title&properties[]=shortDescription. Frappe le! Belle! Nous obtenons toujours les champs JSON-LD standard, mais nous *ne* récupérons *que* ces deux champs. Cette idée est parfois appelée un "jeu de champs épars", et c'est un excellent moyen de permettre à votre client API de demander *exactement* ce qu'il veut, tout en organisant tout autour de ressources API concrètes.

Oh, et l'utilisateur ne peut pas essayer de sélectionner de *nouveaux* champs qui ne font pas partie de nos données d'origine - vous ne pouvez pas essayer d'obtenir isPublished- cela ne fonctionne tout simplement pas, bien que vous *puissiez l'* activer.

Ensuite: parlons de la pagination. Oui, les API ont *totalement* besoin de pagination! Si nous avons 10 000 listes de fromages dans la base de données, nous *ne pouvons pas tous les* renvoyer à la fois.

**● Pagination**

Tout d'abord, cela peut être modifié globalement dans votre config/packages/api\_platform.yamlfichier. Je ne le montrerai pas maintenant, mais rappelez-vous toujours que vous pouvez exécuter:

php bin/console debug:config api\_platform

pour voir une liste de toutes les configurations valides pour ce fichier et leurs valeurs actuelles. Cela révélerait une collection.paginationsection *pleine* de config.

Mais nous pouvons également contrôler le nombre d'éléments par page, ressource par ressource. À l'intérieur de l' @ApiResourceannotation, ajoutez attributes={}... qui est une clé qui contient une variété de configurations aléatoires pour API Platform. Et puis, "pagination\_items\_per\_page": 10.

 167 lines  [src/Entity/CheeseListing.php](https://symfonycasts.com/screencast/api-platform/pagination" \l "codeblock-62704bc366" \o "src/Entity/CheeseListing.php)

|  |  |
| --- | --- |
|  | *... lines 1 – 15* |
|  | /\*\* |
|  | \* @ApiResource( |
|  | *... lines 18 – 25* |
|  | \* attributes={ |
|  | \* "pagination\_items\_per\_page"=10 |
|  | \* } |
|  | \* ) |
|  | *... lines 30 – 34* |
|  | \*/ |
|  | class CheeseListing |
|  | *... lines 37 – 167* |

J'ai mentionné plus tôt que beaucoup de plateformes API apprennent exactement *ce que* vous pouvez configurer à l' intérieur de cette annotation et comment. *Ceci* est un parfait exemple.

Revenez à la documentation - pas besoin de rafraîchir. Appuyez simplement sur Exécuter. Voyons voir ... le nombre total d'articles est de 11 ... mais si vous avez compté, cela n'affiche que *10* résultats! Bonjour pagination! Nous avons également une nouvelle hydra:viewpropriété. Cette annonce qu'il se passe et la pagination comment nous pouvons « parcourir » à travers les autres pages: nous pouvons suivre hydra:first, hydra:lastet hydra:next d'aller à la première, dernière ou page suivante. Les URL exactement comme je veux: ?page=1, ?page=2et ainsi de suite.

Ouvrez un nouvel onglet et revenez à /api/cheeses.jsonld. Oui, les 10 premiers résultats. Maintenant, ajoutez ?page=2... pour voir le dernier résultat.

Le filtrage fonctionne *également* toujours. Essayez .jsonld?title=cheese. Cela renvoie ... seulement 10 résultats ... donc pas de pagination! Ce n'est pas amusant. Revenons aux documents, ouvrons le point de terminaison POST et ajoutons-en quelques autres. Oh, mais assurons-nous d'en ajouter un avec "fromage" dans le titre. Appuyez sur Exécuter plusieurs fois.

*Maintenant,* actualisez l'opération de collecte GET avec ?title=cheese. Agréable! Nous avons 13 résultats au total et cela montre les 10 premiers. Ce qui est *vraiment* bien, c'est que les liens de pagination *incluent* le filtre! C'est *super* utile en JavaScript: vous n'avez pas besoin d'essayer de pirater l'URL manuellement en combinant les pageinformations et les filtres: il suffit de lire les liens d'Hydra et de les utiliser.

Sous-ressources

Une sous-ressource est une collection ou un élément qui appartient à une autre ressource. La plate-forme API facilite la création de telles opérations.

Le point de départ d'une sous-ressource doit être une relation sur une ressource existante. Par exemple, créons deux entités (Question, Réponse) et configurons une sous-ressource pour que / question / 42 / réponse nous donne la réponse à la question 42:

<?php

// api/src/Entity/Answer.php

namespace App\Entity;

use ApiPlatform\Core\Annotation\ApiResource;

use Doctrine\ORM\Mapping as ORM;

/\*\*

\* @ORM\Entity

\* @ApiResource

\*/

class Answer

{

/\*\*

\* @ORM\Column(type="integer")

\* @ORM\Id

\* @ORM\GeneratedValue(strategy="AUTO")

\*/

private $id;

/\*\*

\* @ORM\Column

\*/

public $content;

/\*\*

\* @ORM\OneToOne(targetEntity="Question", mappedBy="answer")

\*/

public $question;

public function getId(): ?int

{

return $this->id;

}

// ...

}

// api/src/Entity/Question.php

namespace App\Entity;

use ApiPlatform\Core\Annotation\ApiResource;

use ApiPlatform\Core\Annotation\ApiSubresource;

use Doctrine\ORM\Mapping as ORM;

/\*\*

\* @ORM\Entity

\* @ApiResource

\*/

class Question

{

/\*\*

\* @ORM\Column(type="integer")

\* @ORM\Id

\* @ORM\GeneratedValue(strategy="AUTO")

\*/

private $id;

/\*\*

\* @ORM\Column

\*/

public $content;

/\*\*

\* @ORM\OneToOne(targetEntity="Answer", inversedBy="question")

\* @ORM\JoinColumn(referencedColumnName="id", unique=true)

\* @ApiSubresource

\*/

public $answer;

public function getId(): ?int

{

return $this->id;

}

// ...

}

Notez que tout ce que nous avions à faire est de configurer @ApiSubresource sur la relation Question :: answer. Parce que la réponse est une relation un à un, nous savons que cette sous-ressource est un élément. Par conséquent, la réponse ressemblera à ceci:

{

"@context": "/contexts/Answer",

"@id": "/answers/42",

"@type": "Answer",

"id": 42,

"content": "Life, the Universe, and Everything",

"question": "/questions/42"

}

Si vous placez la sous-ressource sur une relation qui est à plusieurs, vous récupérerez une collection.

Enfin et surtout, les sous-ressources peuvent être imbriquées, de telle sorte que / questions / 42 / réponse / commentaires obtienne la collection de commentaires pour la réponse à la question 42.

Remarque: seules les opérations GET sont prises en charge pour le moment

Utilisation de groupes de sérialisation

Vous pouvez souhaiter des groupes personnalisés sur les sous-ressources, vous pouvez définir normalization\_context ou denormalization\_context sur cette opération. Pour ce faire, ajoutez un nœud subresourceOperations. Par exemple:

<?php

// api/src/Entity/Answer.php

namespace App\Entity;

use ApiPlatform\Core\Annotation\ApiResource;

/\*\*

\* @ApiResource(subresourceOperations={

\* "api\_questions\_answer\_get\_subresource"={

\* "method"="GET",

\* "normalization\_context"={"groups"={"foobar"}}

\* }

\* })

\*/

class Answer

{

// ...

}

Dans les exemples précédents, l'attribut method est obligatoire, car le nom de l'opération ne correspond pas à une méthode HTTP prise en charge.

Notez que le nom de l'opération, ici api\_questions\_answer\_get\_subresource, est le mot clé important. Il sera automatiquement défini sur $ resources\_ $ subresource (s) \_get\_subresource. Pour trouver le nom d'opération correct, vous pouvez utiliser debug bin / console debug: router.

Utilisation de chemins personnalisés

Vous pouvez contrôler le chemin des sous-ressources avec l'option chemin du paramètre subresourceOperations:

<?php

// api/src/Entity/Question.php

/\*\*

\* ...

\* @ApiResource(

\* subresourceOperations={

\* "api\_questions\_answer\_get\_subresource"={

\* "method"="GET",

\* "path"="/questions/{id}/all-answers"

\* },

\* },

\* )

\*/

class Question

{

}

**itemsOperation**

La commande **ItemOperations** agit comme un conteneur pour l' élément **Fetch** (section [2.2.3.67.1](https://docs.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-ascmd/7782504c-43f2-4cef-9147-2d61ce8aa4e3) ), l' élément **EmptyFolderContents**(section [2.2.3.58](https://docs.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-ascmd/bae7d1c9-959a-4a2e-8316-4dc7f4cc8bb5) ) et l' élément **Move** (section [2.2.3.117.1](https://docs.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-ascmd/fa51531c-650d-4baa-bf36-82845e5e89ea) ) pour fournir une gestion en ligne par lots de ces opérations contre le serveur.

Les opérations contenues dans l' élément **ItemOperations** (section [2.2.3.94](https://docs.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-ascmd/aa329d4e-497b-4f53-88d6-26ce392dd99e) ) DOIVENT être exécutées par le serveur dans l'ordre spécifié. Le serveur DOIT signaler l'état par opération au client. Par conséquent, le client corrèle ces réponses à l'opération initiale et procède de manière appropriée.

La commande **ItemOperations** ne peut pas effectuer d'opérations sur les éléments dans le cache d'informations du destinataire.

L'opération d'extraction est destinée à être utilisée sur la technologie Microsoft Windows SharePoint Services, pour [l'](https://docs.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-ascmd/2fcc6436-1569-4416-adcf-85e3a64e9b78#gt_c9507dca-291d-4fd6-9cba-a9ee7da8c908) accès [UNC (Universal Naming Convention)](https://docs.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-ascmd/2fcc6436-1569-4416-adcf-85e3a64e9b78#gt_c9507dca-291d-4fd6-9cba-a9ee7da8c908) ou pour l'accès aux boîtes aux lettres des utilisateurs. Les métadonnées du document, les résultats de la recherche et les éléments et pièces jointes de la boîte aux lettres peuvent être récupérés.

L' opération **EmptyFolderContents** permet au client de vider un dossier de tous ses éléments. Les clients utilisent généralement l' opération **EmptyFolderContents** pour effacer tous les éléments du [dossier Éléments supprimés](https://docs.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-ascmd/2fcc6436-1569-4416-adcf-85e3a64e9b78#gt_3f0c7497-a422-4de2-8f2a-d9bd0f63b659) si l'utilisateur manque de quota de stockage.

L' opération **Déplacer** déplace une [conversation](https://docs.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-ascmd/2fcc6436-1569-4416-adcf-85e3a64e9b78#gt_0aec5fa3-827f-4725-9d37-4b5bff86d6e1) vers un dossier de destination.

L' **espace de** noms **ItemOperations** est l'espace de noms principal de cette section. Les éléments référencés dans cette section qui ne sont pas définis dans l' **espace de** noms **ItemOperations** utilisent les préfixes d'espace de noms définis dans la section [2.2.4](https://docs.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-ascmd/718fe481-c75b-4ff7-bbf6-5cf3dd0f43d7) .

CollectionsOperation

API Platform Core repose sur le concept d'opérations. Les opérations peuvent être appliquées à une ressource exposée par l'API. Du point de vue de la mise en œuvre, une opération est un lien entre une ressource, une route et son contrôleur associé.

API Platform enregistre automatiquement les opérations CRUD typiques et les décrit dans la documentation exposée (Hydra et Swagger). Il crée et enregistre également des routes correspondant à ces opérations dans le système de routage Symfony (s'il est disponible). Le comportement des opérations intégrées est brièvement présenté dans le guide Mise en route. La liste des opérations activées peut être configurée pour chaque ressource. La création d'opérations personnalisées sur des itinéraires spécifiques est également possible. Il existe deux types d'opérations: les opérations de collecte et les opérations d'élément. Les opérations de collecte agissent sur une collection de ressources. Par défaut, deux routes sont implémentées: POST et GET. Les opérations sur les articles agissent sur une ressource individuelle. Trois itinéraires par défaut sont définis: GET, PUT et DELETE (PATCH est également pris en charge lors de l'utilisation du format JSON: API, comme requis par la spécification).

6)Gestion de l’authentification

La gestion de l'authentification permet l'implémentation de procédures de connexion avec des jetons physiques d'authentification (carte à puce, carte USB, badges RFID), de la biométrie et des smartphones, en plus de l'authentification standard par identifiant et mot de passe. La gestion de l'authentification est mise œuvre par [Evidian Authentication Manager](https://www.evidian.com/fr/produits/authentification-manager/). La vidéo présente plusieurs méthodes d'authentification proposées par Authentication Manager.

## Améliorer la sécurité avec le 2FA- Two factor authentication/authentification à deux facteurs

Authentication Manager permet de combiner des méthodes d'authentification forte pour renforcer la sécurité d’accès à une session Windows. Avec l’authentification à deux facteurs, vos utilisateurs devront utiliser deux méthodes d’authentification pour se connecter à leur session Windows. Vous pouvez combiner toutes les méthodes d'authentification forte prises en charge par Evidian Authentication Manager, telles que la biométrie et le code PIN ou la RFID et le code PIN.

Avec Enterprise Single Sign-On et Authentication Manager, vous pouvez utiliser l'authentification à deux facteurs pour la réauthentification lors de l'accès à des applications sensibles ou critiques.

● Notion JWT

**JSON Web Token** (JWT) est un standard ouvert défini dans la RFC 7519[1](https://fr.wikipedia.org/wiki/JSON_Web_Token#cite_note-1). Il permet l'échange sécurisé de jetons (tokens) entre plusieurs parties. Cette sécurité de l’échange se traduit par la vérification de l’intégrité des données à l’aide d’une signature numérique. Elle s’effectue par l'algorithme [HMAC](https://fr.wikipedia.org/wiki/Keyed-Hash_Message_Authentication_Code) ou [RSA](https://fr.wikipedia.org/wiki/Chiffrement_RSA).

Un jeton se compose de trois parties :

* Un *en-tête* (header), utilisé pour décrire le jeton. Il s'agit d'un objet JSON.
* Une *charge utile* (payload) qui représente les informations embarquées dans le jeton. Il s'agit également d'un objet JSON.
* Une *signature* numérique.

## Obtention de la signature.

Pour obtenir la signature, il faut tout d'abord encoder séparément l'en-tête et la charge utile avec [Base64url](https://fr.wikipedia.org/wiki/Base64#base64url) défini dans la RFC 4648[2](https://fr.wikipedia.org/wiki/JSON_Web_Token#cite_note-2). Ensuite, on les concatène ensemble en les séparant avec un point. On obtient la signature de ce résultat avec l'algorithme choisi. Cette signature est ajoutée au résultat de la même manière (encodée et séparée par un point).

À noter que pour l'encodage en [Base64url](https://fr.wikipedia.org/wiki/Base64#base64url), le caractère de remplissage '=' n'est pas obligatoire[3](https://fr.wikipedia.org/wiki/JSON_Web_Token#cite_note-3) et ne sera pas utilisé dans la création du JSON Web Token pour faciliter la transmission dans une URL.

● Firewall

Un **pare-feu**[1](https://fr.wikipedia.org/wiki/Pare-feu_(informatique)#cite_note-FranceTerme-1),[2](https://fr.wikipedia.org/wiki/Pare-feu_(informatique)#cite_note-GDT-2) (de l'[anglais](https://fr.wikipedia.org/wiki/Anglais) *firewall*[3](https://fr.wikipedia.org/wiki/Pare-feu_(informatique)#cite_note-3)) est un [logiciel](https://fr.wikipedia.org/wiki/Logiciel) et/ou un [matériel](https://fr.wikipedia.org/wiki/Mat%C3%A9riel_informatique) permettant de faire respecter la [politique de sécurité du réseau](https://fr.wikipedia.org/wiki/Politique_de_s%C3%A9curit%C3%A9_du_r%C3%A9seau_informatique), celle-ci définissant quels sont les types de communications autorisés sur ce [réseau informatique](https://fr.wikipedia.org/wiki/R%C3%A9seau_informatique). Il surveille et contrôle les applications et les flux de données ([paquets](https://fr.wikipedia.org/wiki/Paquet_(r%C3%A9seau))).

Un pare-feu est parfois appelé *coupe-feu*, *garde-barrière*, *barrière de sécurité*, ou encore *firewall*. Traduction littérale : mur de feu[[réf. souhaitée]](https://fr.wikipedia.org/wiki/Aide:R%C3%A9f%C3%A9rence_n%C3%A9cessaire).

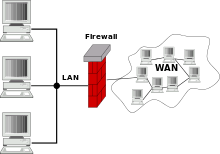
Dans un environnement [Unix](https://fr.wikipedia.org/wiki/Unix) BSD ([Berkeley Software Distribution](https://fr.wikipedia.org/wiki/Berkeley_Software_Distribution)), un pare-feu est aussi appelé *[packet filter](https://fr.wikipedia.org/wiki/Packet_Filter" \o "Packet Filter)*.

## Origine du terme[[modifier](https://fr.wikipedia.org/w/index.php?title=Pare-feu_(informatique)&veaction=edit&section=2) | [modifier le code](https://fr.wikipedia.org/w/index.php?title=Pare-feu_(informatique)&action=edit&section=2)]

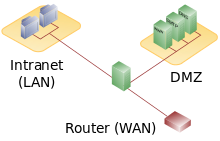
Selon le contexte, le terme peut revêtir différentes significations :

* dans le domaine de la lutte contre les [incendies de forêt](https://fr.wikipedia.org/wiki/Incendies_de_for%C3%AAt), il se réfère aux allées [pare-feu](https://fr.wikipedia.org/wiki/Pare-feux_(lutte_contre_l%27incendie)) destinées à contenir l'extension des feux de forêts ;
* au [théâtre](https://fr.wikipedia.org/wiki/Th%C3%A9%C3%A2tre), le déclenchement d'un mécanisme « *pare-feu* » (ou « *coupe-feu* ») permet d'éviter la propagation du [feu](https://fr.wikipedia.org/wiki/Feu) de la salle vers la scène ;
* dans le domaine de l'[architecture](https://fr.wikipedia.org/wiki/Architecture), il fait référence aux [portes coupe-feu](https://fr.wikipedia.org/wiki/Porte_coupe-feu) ou à tout autre dispositif constructif destiné à contenir l'extension d'un incendie ;
* en informatique, l'usage du terme « pare-feu » est donc métaphorique. Il représente un mur virtuel qui bloque tout ce qui tente d'entrer avec l'intention de nuire dans une machine ou un réseau. Il établit une barrière de protection contre les intrusions et les contaminations venant de l’extérieur.

## Fonctionnement général[[modifier](https://fr.wikipedia.org/w/index.php?title=Pare-feu_(informatique)&veaction=edit&section=3) | [modifier le code](https://fr.wikipedia.org/w/index.php?title=Pare-feu_(informatique)&action=edit&section=3)]

[](https://commons.wikimedia.org/wiki/File:Gateway_firewall.svg?uselang=fr)

Pare-feu passerelle entre LAN et WAN.

[](https://commons.wikimedia.org/wiki/File:DMZ_network_diagram_1_firewall.svg?uselang=fr)

Pare-feu routeur, avec une zone DMZ.

Le pare-feu est jusqu'à ces dernières années considéré comme une des pierres angulaires de la [sécurité](https://fr.wikipedia.org/wiki/S%C3%A9curit%C3%A9_du_syst%C3%A8me_d%27information) d'un réseau informatique (il perd en importance au fur et à mesure que les communications basculent vers le [HTTP](https://fr.wikipedia.org/wiki/HTTP) sur [TLS](https://fr.wikipedia.org/wiki/Transport_Layer_Security), court-circuitant tout filtrage). Il permet d'appliquer une politique d'accès aux ressources réseau ([serveurs](https://fr.wikipedia.org/wiki/Serveur_informatique)).

Il a pour principale tâche de contrôler le trafic entre différentes zones de confiance, en filtrant les flux de données qui y transitent. Généralement, les zones de confiance incluent [Internet](https://fr.wikipedia.org/wiki/Internet) (une zone dont la confiance est nulle) et au moins un [réseau interne](https://fr.wikipedia.org/wiki/R%C3%A9seau_local) (une zone dont la confiance est plus importante).

Le but est de fournir une connectivité contrôlée et maîtrisée entre des zones de différents niveaux de confiance, grâce à l'application de la [politique de sécurité](https://fr.wikipedia.org/wiki/Politique_de_s%C3%A9curit%C3%A9) et d'un modèle de connexion basé sur le [principe du moindre privilège](https://fr.wikipedia.org/wiki/S%C3%A9paration_des_privil%C3%A8ges).

Le filtrage se fait selon divers critères. Les plus courants sont :

* l'origine ou la destination des paquets ([adresse IP](https://fr.wikipedia.org/wiki/Adresse_IP), [ports](https://fr.wikipedia.org/wiki/Port_(logiciel)) [TCP](https://fr.wikipedia.org/wiki/Transmission_Control_Protocol) ou [UDP](https://fr.wikipedia.org/wiki/User_Datagram_Protocol), interface réseau, etc.) ;
* les options contenues dans les données (fragmentation, validité, etc.) ;
* les données elles-mêmes (taille, correspondance à un motif, etc.) ;
* les utilisateurs pour les plus récents.

Un pare-feu fait souvent office de [routeur](https://fr.wikipedia.org/wiki/Routeur) et permet ainsi d'isoler le réseau en plusieurs zones de sécurité appelées [zones démilitarisées ou DMZ](https://fr.wikipedia.org/wiki/Zone_d%C3%A9militaris%C3%A9e_(informatique)). Ces zones sont séparées suivant le niveau de confiance qu'on leur porte.

Enfin, le pare-feu est également souvent situé à l'extrémité de tunnel [IPsec](https://fr.wikipedia.org/wiki/Internet_Protocol_Security" \o "Internet Protocol Security) ou TLS. L'intégration du filtrage de flux et de la gestion du tunnel est en effet nécessaire pour pouvoir à la fois protéger le trafic en confidentialité et intégrité et filtrer ce qui passe dans le tunnel. C'est le cas notamment de plusieurs produits du commerce nommés dans la liste ci-dessous.

## Catégories de pare-feu[[modifier](https://fr.wikipedia.org/w/index.php?title=Pare-feu_(informatique)&veaction=edit&section=4) | [modifier le code](https://fr.wikipedia.org/w/index.php?title=Pare-feu_(informatique)&action=edit&section=4)]

Les pare-feux sont un des plus vieux équipements de [sécurité informatique](https://fr.wikipedia.org/wiki/S%C3%A9curit%C3%A9_informatique) et, en tant que tel, ont subi de nombreuses évolutions. Suivant la génération du pare-feu ou son rôle précis, on peut les classer en différentes catégories.

### Pare-feu sans état (*stateless firewall*)**[**[**modifier**](https://fr.wikipedia.org/w/index.php?title=Pare-feu_(informatique)&veaction=edit&section=5)**|**[**modifier le code**](https://fr.wikipedia.org/w/index.php?title=Pare-feu_(informatique)&action=edit&section=5)**]**

C'est le plus vieux dispositif de filtrage réseau, introduit sur les routeurs. Il regarde chaque paquet indépendamment des autres et le compare à une liste de règles préconfigurées.  
Ces règles peuvent avoir des noms très différents en fonction du pare-feu :

* « [ACL](https://fr.wikipedia.org/wiki/Access_Control_List) » pour *Access Control List* (certains pare-feux [Cisco](https://fr.wikipedia.org/wiki/Cisco_Systems)),
* politique ou *policy* (pare-feu [Juniper](https://fr.wikipedia.org/wiki/Juniper_Networks" \o "Juniper Networks)/Netscreen),
* filtres,
* règles ou *rules*,
* etc.

La configuration de ces dispositifs est souvent complexe et l'absence de prise en compte des machines à états des protocoles réseaux ne permet pas d'obtenir une finesse du filtrage très évoluée. Ces pare-feux ont donc tendance à tomber en désuétude mais restent présents sur certains routeurs ou systèmes d'exploitation.

Article connexe : [Serveur sans état](https://fr.wikipedia.org/wiki/Serveur_sans_%C3%A9tat).

### Pare-feu à états (*stateful firewall*)**[**[**modifier**](https://fr.wikipedia.org/w/index.php?title=Pare-feu_(informatique)&veaction=edit&section=6)**|**[**modifier le code**](https://fr.wikipedia.org/w/index.php?title=Pare-feu_(informatique)&action=edit&section=6)**]**

Certains protocoles dits « à états » comme [TCP](https://fr.wikipedia.org/wiki/Transmission_Control_Protocol) introduisent une notion de connexion. Les pare-feux à états vérifient la conformité des [paquets](https://fr.wikipedia.org/wiki/Paquet_(logiciel)) à une connexion en cours. C’est-à-dire qu'ils vérifient que chaque paquet d'une connexion est bien la suite du précédent paquet et la réponse à un paquet dans l'autre sens. Ils savent aussi filtrer intelligemment les paquets [ICMP](https://fr.wikipedia.org/wiki/Internet_Control_Message_Protocol) qui servent à la signalisation des flux [IP](https://fr.wikipedia.org/wiki/Internet_Protocol).

Enfin, si les [ACL](https://fr.wikipedia.org/wiki/Access_Control_List) autorisent un paquet [UDP](https://fr.wikipedia.org/wiki/User_Datagram_Protocol) caractérisé par un quadruplet (ip\_src, port\_src, ip\_dst, port\_dst) à passer, un tel pare-feu autorisera la réponse caractérisée par un quadruplet inversé, sans avoir à écrire une ACL inverse. Ceci est fondamental pour le bon fonctionnement de tous les protocoles fondés sur l'[UDP](https://fr.wikipedia.org/wiki/User_Datagram_Protocol), comme [DNS](https://fr.wikipedia.org/wiki/Domain_Name_System) par exemple. Ce mécanisme apporte en fiabilité puisqu'il est plus sélectif quant à la nature du trafic autorisé. Cependant dans le cas d'UDP, cette caractéristique peut être utilisée pour établir des connexions directes ([P2P](https://fr.wikipedia.org/wiki/Pair-%C3%A0-pair)) entre deux machines (comme le fait [Skype](https://fr.wikipedia.org/wiki/Skype) par exemple).

Article connexe : [Pare-feu à états](https://fr.wikipedia.org/wiki/Pare-feu_%C3%A0_%C3%A9tats).

### Pare-feu applicatif**[**[**modifier**](https://fr.wikipedia.org/w/index.php?title=Pare-feu_(informatique)&veaction=edit&section=7)**|**[**modifier le code**](https://fr.wikipedia.org/w/index.php?title=Pare-feu_(informatique)&action=edit&section=7)**]**

Dernière génération de pare-feu, ils vérifient la complète conformité du paquet à un protocole attendu. Par exemple, ce type de pare-feu permet de vérifier que seul le protocole [HTTP](https://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol) passe par le port [TCP](https://fr.wikipedia.org/wiki/Transmission_Control_Protocol) 80. Ce traitement est très gourmand en temps de calcul dès que le débit devient très important. Il est justifié par le fait que de plus en plus de protocoles réseaux utilisent un [tunnel](https://fr.wikipedia.org/wiki/Tunnel_(r%C3%A9seau_informatique)) TCP afin de contourner le filtrage par [ports](https://fr.wikipedia.org/wiki/Port_(logiciel)).

Une autre raison de l'inspection applicative est l'ouverture de [ports](https://fr.wikipedia.org/wiki/Port_(logiciel)) dynamique. Certains protocoles comme [FTP](https://fr.wikipedia.org/wiki/File_Transfer_Protocol), en mode passif, échangent entre le client et le serveur des adresses IP ou des ports TCP/UDP. Ces protocoles sont dits « **à contenu sale** » ou « [passant difficilement les pare-feux](https://fr.wikipedia.org/wiki/Protocole_r%C3%A9seau_passant_difficilement_les_pare-feu) » car ils échangent au niveau applicatif (FTP) des informations du niveau IP (échange d'adresses) ou du niveau TCP (échange de ports). Ce qui transgresse le principe de la séparation des [couches réseaux](https://fr.wikipedia.org/wiki/Couche_de_r%C3%A9seau). Pour cette raison, les protocoles « à contenu sale » passent difficilement voire pas du tout les règles de [NAT](https://fr.wikipedia.org/wiki/Network_address_translation) ...dynamiques, à moins qu'une inspection applicative ne soit faite sur ce protocole.

Chaque type de pare-feu sait inspecter un nombre limité d'applications. Chaque application est gérée par un module différent pour pouvoir les activer ou les désactiver. La terminologie pour le concept de module est différente pour chaque type de pare-feu : par exemple : Le protocole HTTP permet d'accéder en lecture sur un serveur par une commande GET, et en écriture par une commande PUT. Un pare-feu applicatif va être en mesure d'analyser une connexion HTTP et de n'autoriser les commandes PUT qu'à un nombre restreint de machines.

### Pare-feu identifiant **:**

Un pare-feu réalise l’identification des [connexions](https://fr.wikipedia.org/wiki/Protocole_r%C3%A9seau) passant à travers le filtre IP. L'administrateur peut ainsi définir les règles de filtrage par utilisateur et non plus par [adresse IP](https://fr.wikipedia.org/wiki/Adresse_IP) ou [adresse MAC](https://fr.wikipedia.org/wiki/Adresse_MAC), et ainsi suivre l'activité réseau par utilisateur.

Plusieurs méthodes différentes existent qui reposent sur des associations entre IP et utilisateurs réalisées par des moyens variés. On peut par exemple citer [authpf](http://www.openbsd.org/faq/pf/authpf.html) [[archive](http://archive.wikiwix.com/cache/?url=http%3A%2F%2Fwww.openbsd.org%2Ffaq%2Fpf%2Fauthpf.html)] (sous [OpenBSD](https://fr.wikipedia.org/wiki/OpenBSD" \o "OpenBSD)) qui utilise [ssh](https://fr.wikipedia.org/wiki/Secure_shell" \o "Secure shell) pour faire l'association. Une autre méthode est l'identification connexion par connexion (sans avoir cette association IP = utilisateur et donc sans compromis sur la sécurité), réalisée par exemple par la suite [NuFW](https://fr.wikipedia.org/wiki/NuFW" \o "NuFW), qui permet d'identifier également sur des machines multi-utilisateurs.  
On pourra également citer Cyberoam qui fournit un pare-feu entièrement basé sur l'identité (en réalité en réalisant des associations adresse MAC = utilisateur) ou [Check Point](https://fr.wikipedia.org/wiki/Check_Point) avec l'option NAC Blade qui permet de créer des règles dynamiques basée sur l'authentification [Kerberos](https://fr.wikipedia.org/wiki/Kerberos_(protocole)" \o "Kerberos (protocole)) d'un utilisateur, l'identité de son poste ainsi que son niveau de sécurité (présence d'antivirus, de [patchs](https://fr.wikipedia.org/wiki/Patch_(informatique)) particuliers).

7)API Platform et Evenements Doctrine

Remarque: l'utilisation de l'événement Kernel avec la plate-forme API doit être principalement limitée à la modification de la réponse HTTP générée. De plus, GraphQL n'est **pas pris en charge** . [**Pour la plupart des cas d'utilisation, de meilleurs points d'extension, fonctionnant à la fois avec REST et GraphQL, sont disponibles**](https://api-platform.com/docs/core/extending/) .

API Platform Core implémente le modèle [**Action-Domain-Responder**](https://github.com/pmjones/adr) . Cette implémentation est traitée en détail dans le chapitre [**Création d'opérations et de contrôleurs personnalisés**](https://api-platform.com/docs/core/operations/#creating-custom-operations-and-controllers) .

Fondamentalement, API Platform Core exécute une classe d'actions qui retournera une entité ou une collection d'entités. Ensuite, une série d'écouteurs d'événements est exécutée pour valider les données, les conserver dans la base de données, les sérialiser (généralement dans un document JSON-LD) et créer une réponse HTTP qui sera envoyée au client.

Pour ce faire, API Platform Core exploite les [**événements déclenchés par le noyau HTTP Symfony**](https://symfony.com/doc/current/reference/events.html#kernel-events) . Vous pouvez également connecter votre propre code à ces événements. Des points d'extension pratiques et puissants sont disponibles à tous les stades du cycle de vie de la demande.

Si vous utilisez Doctrine, les événements de cycle de vie ( [**ORM**](https://www.doctrine-project.org/projects/doctrine-orm/en/current/reference/events.html#lifecycle-events) , **[MongoDB ODM](https://www.doctrine-project.org/projects/doctrine-mongodb-odm/en/latest/reference/events.html" \l "lifecycle-events" \t "_blank)** ) sont également disponibles si vous souhaitez vous connecter au cycle de vie des objets de la couche de persistance.

8)API Platform et Controllers Personnalisés

Remarque: l'utilisation de contrôleurs personnalisés avec API Platform est **déconseillée** . De plus, GraphQL n'est **pas pris en charge** . [**Pour la plupart des cas d'utilisation, de meilleurs points d'extension, fonctionnant à la fois avec REST et GraphQL, sont disponibles**](https://api-platform.com/docs/core/design/) .

La plate-forme API peut tirer parti du système de routage Symfony pour enregistrer les opérations personnalisées liées aux contrôleurs personnalisés. Ces contrôleurs personnalisés peuvent être n'importe quel [**contrôleur Symfony**](http://symfony.com/doc/current/book/controller.html) valide , y compris les contrôleurs Symfony standard étendant la [**Symfony\Bundle\FrameworkBundle\Controller\AbstractController**](http://api.symfony.com/4.1/Symfony/Bundle/FrameworkBundle/Controller/AbstractController.html) classe d'assistance.

Cependant, API Platform recommande d'utiliser **des classes d'actions** au lieu des contrôleurs Symfony classiques. En interne, API Platform implémente le modèle [**Action-Domain-Responder**](https://github.com/pmjones/adr) (ADR), un raffinement spécifique au Web de [**MVC**](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) .

La distribution d'API Platform facilite également la mise en œuvre du modèle ADR: il enregistre automatiquement les classes d'actions stockées en api/src/Controllertant que services câblés automatiquement.

Grâce à la fonctionnalité de [**câblage**](http://symfony.com/doc/current/components/dependency_injection/autowiring.html) automatique du conteneur Symfony Dependency Injection, les services requis par une action peuvent être indiqués dans son constructeur, il sera automatiquement instancié et injecté, sans avoir à le déclarer explicitement.

Dans les exemples suivants, l' GETopération intégrée est enregistrée ainsi qu'une opération personnalisée appelée post\_publication.

Par défaut, API Platform utilise la première GETopération définie dans itemOperationspour générer l'IRI d'un élément et la première GETopération définie dans collectionOperationspour générer l'IRI d'une collection.

Si vous créez une opération personnalisée, vous souhaiterez probablement la documenter correctement. Voir la partie **[OpenAPI](https://api-platform.com/docs/core/swagger/)** de la documentation pour le faire.

Commençons par créer votre opération personnalisée:

<?php

// api/src/Controller/CreateBookPublication.php

namespace App\Controller;

use App\Entity\Book;

class CreateBookPublication

{

private $bookPublishingHandler;

public function \_\_construct(BookPublishingHandler $bookPublishingHandler)

{

$this->bookPublishingHandler = $bookPublishingHandler;

}

public function \_\_invoke(Book $data): Book

{

$this->bookPublishingHandler->handle($data);

return $data;

}

}

Cette opération personnalisée se comporte exactement comme l'opération intégrée: elle renvoie un document JSON-LD correspondant à l'ID passé dans l'URL.

Ici, nous considérons que le [**câblage automatique**](https://symfony.com/doc/current/service_container/autowiring.html) est activé pour les classes de contrôleur (valeur par défaut lors de l'utilisation de la distribution API Platform). Cette action sera automatiquement enregistrée en tant que service (le nom du service est le même que le nom de la classe:) App\Controller\CreateBookPublication.

La plate-forme API récupère automatiquement l'entité PHP appropriée à l'aide du fournisseur de données, puis désérialise les données utilisateur, POSTet PUTdemande et met à jour l'entité avec les données fournies par l'utilisateur.

**Attention: le \_\_invoke()paramètre de méthode**[**DOIT être appelé$data**](https://symfony.com/doc/current/components/http_kernel.html#getting-the-controller-arguments) , sinon, il ne sera pas rempli correctement!

Les services ( $bookPublishingHandlerici) sont automatiquement injectés grâce à la fonction de câblage automatique. Vous pouvez taper n'importe quel service dont vous avez besoin et il sera également câblé automatiquement.

La \_\_invokeméthode de l'action est appelée lorsque l'itinéraire correspondant est atteint. Il peut renvoyer soit une instance de Symfony\Component\HttpFoundation\Response(qui sera immédiatement affichée au client par le noyau Symfony), soit, comme dans cet exemple, une instance d'une entité mappée en tant que ressource (ou une collection d'instances pour les opérations de collecte). Dans ce cas, l'entité passera par [**tous les écouteurs d'événements intégrés**](https://api-platform.com/docs/core/events/#built-in-event-listeners) d'API Platform. Il sera automatiquement validé, conservé et sérialisé en JSON-LD. Ensuite, le noyau Symfony enverra le document résultant au client.

Le routage n'a pas encore été configuré car nous l'ajouterons au niveau de la configuration des ressources:

PHPYAMLXML

<?php

// api/src/Entity/Book.php

use ApiPlatform\Core\Annotation\ApiResource;

use App\Controller\CreateBookPublication;

/\*\*

\* @ApiResource(itemOperations={

\* "get",

\* "post\_publication"={

\* "method"="POST",

\* "path"="/books/{id}/publication",

\* "controller"=CreateBookPublication::class,

\* }

\* })

\*/

class Book

{

//...

}

Il est obligatoire de définir les attributs method, pathet controller. Ils permettent à l'API Platform de configurer respectivement le chemin de routage et le contrôleur associé.

**Utilisation de groupes de sérialisation**

Vous pouvez souhaiter différents groupes de sérialisation pour vos opérations personnalisées. Il suffit de configurer le bon normalization\_contextet / ou denormalization\_contextdans votre opération:

PHPYAMLXML

<?php

// api/src/Entity/Book.php

use ApiPlatform\Core\Annotation\ApiResource;

use App\Controller\CreateBookPublication;

use Symfony\Component\Serializer\Annotation\Groups;

/\*\*

\* @ApiResource(itemOperations={

\* "get",

\* "post\_publication"={

\* "method"="POST",

\* "path"="/books/{id}/publication",

\* "controller"=CreateBookPublication::class,

\* "normalization\_context"={"groups"={"publication"}},

\* }

\* })

\*/

class Book

{

//...

/\*\*

\* @Groups("publication")

\*/

public $isbn;

// ...

}

**Récupération de l'entité**

Si vous souhaitez contourner la récupération automatique de l'entité dans votre opération personnalisée, vous pouvez définir "read"=falsedans l'attribut d'opération:

PHPYAMLXML

<?php

// api/src/Entity/Book.php

use ApiPlatform\Core\Annotation\ApiResource;

use App\Controller\CreateBookPublication;

/\*\*

\* @ApiResource(itemOperations={

\* "get",

\* "post\_publication"={

\* "method"="POST",

\* "path"="/books/{id}/publication",

\* "controller"=CreateBookPublication::class,

\* "read"=false,

\* }

\* })

\*/

class Book

{

//...

}

De cette façon, il sautera le ReadListener. Vous pouvez faire de même pour certains autres écouteurs intégrés. Voir [**Écouteurs d'événements intégrés**](https://api-platform.com/docs/core/events/#built-in-event-listeners) pour plus d'informations.

**Méthode alternative**

Il existe une autre façon de créer une opération personnalisée. Cependant, nous n'encourageons pas son utilisation. En effet, celui-ci disperse la configuration à la fois dans le routage et la configuration des ressources.

L' post\_publicationopération fait référence à la route Symfony nommée book\_post\_publication.

Depuis la version 2.3, vous pouvez également utiliser le nom de la route comme nom d'opération par convention, comme indiqué dans l'exemple suivant book\_post\_discontinuationlorsque ni methodni les route\_nameattributs ne sont spécifiés.

Commençons par créer votre configuration de ressources:

PHPYAMLXML

<?php

// api/src/Entity/Book.php

use ApiPlatform\Core\Annotation\ApiResource;

/\*\*

\* @ApiResource(itemOperations={

\* "get",

\* "post\_publication"={"route\_name"="book\_post\_publication"},

\* "book\_post\_discontinuation",

\* })

\*/

class Book

{

//...

}

La plate-forme API mappera automatiquement cette post\_publicationopération sur l'itinéraire book\_post\_publication. Créons une action personnalisée et son itinéraire associé à l'aide d'annotations:

<?php

// api/src/Controller/CreateBookPublication.php

namespace App\Controller;

use App\Entity\Book;

use Symfony\Component\Routing\Annotation\Route;

class CreateBookPublication

{

private $bookPublishingHandler;

public function \_\_construct(BookPublishingHandler $bookPublishingHandler)

{

$this->bookPublishingHandler = $bookPublishingHandler;

}

/\*\*

\* @Route(

\* name="book\_post\_publication",

\* path="/books/{id}/publication",

\* methods={"POST"},

\* defaults={

\* "\_api\_resource\_class"=Book::class,

\* "\_api\_item\_operation\_name"="post\_publication"

\* }

\* )

\*/

public function \_\_invoke(Book $data): Book

{

$this->bookPublishingHandler->handle($data);

return $data;

}

}

Il est obligatoire de définir \_api\_resource\_classet \_api\_item\_operation\_name(ou \_api\_collection\_operation\_namepour une opération de collecte) dans les paramètres de l'itinéraire ( defaultsclé). Il permet à API Platform de fonctionner avec le système de routage Symfony.

Alternativement, vous pouvez également utiliser un contrôleur Symfony traditionnel et des déclarations de route YAML ou XML. L'exemple suivant fait la même chose que l'exemple précédent:

<?php

// api/src/Controller/BookController.php

namespace App\Controller;

use App\Entity\Book;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

class BookController extends AbstractController

{

public function createPublication(Book $data, BookPublishingHandler $bookPublishingHandler): Book

{

return $bookPublishingHandler->handle($data);

}

}

# api/config/routes.yaml

book\_post\_publication:

path: /books/{id}/publication

methods: ['POST']

defaults:

\_controller: App\Controller\BookController::createPublication

\_api\_resource\_class: App\Entity\Book

\_api\_item\_operation\_name: post\_publication