Drupal découplé | Headless Drupal

Le 11 avril 2018 à Montpellier

Motivations

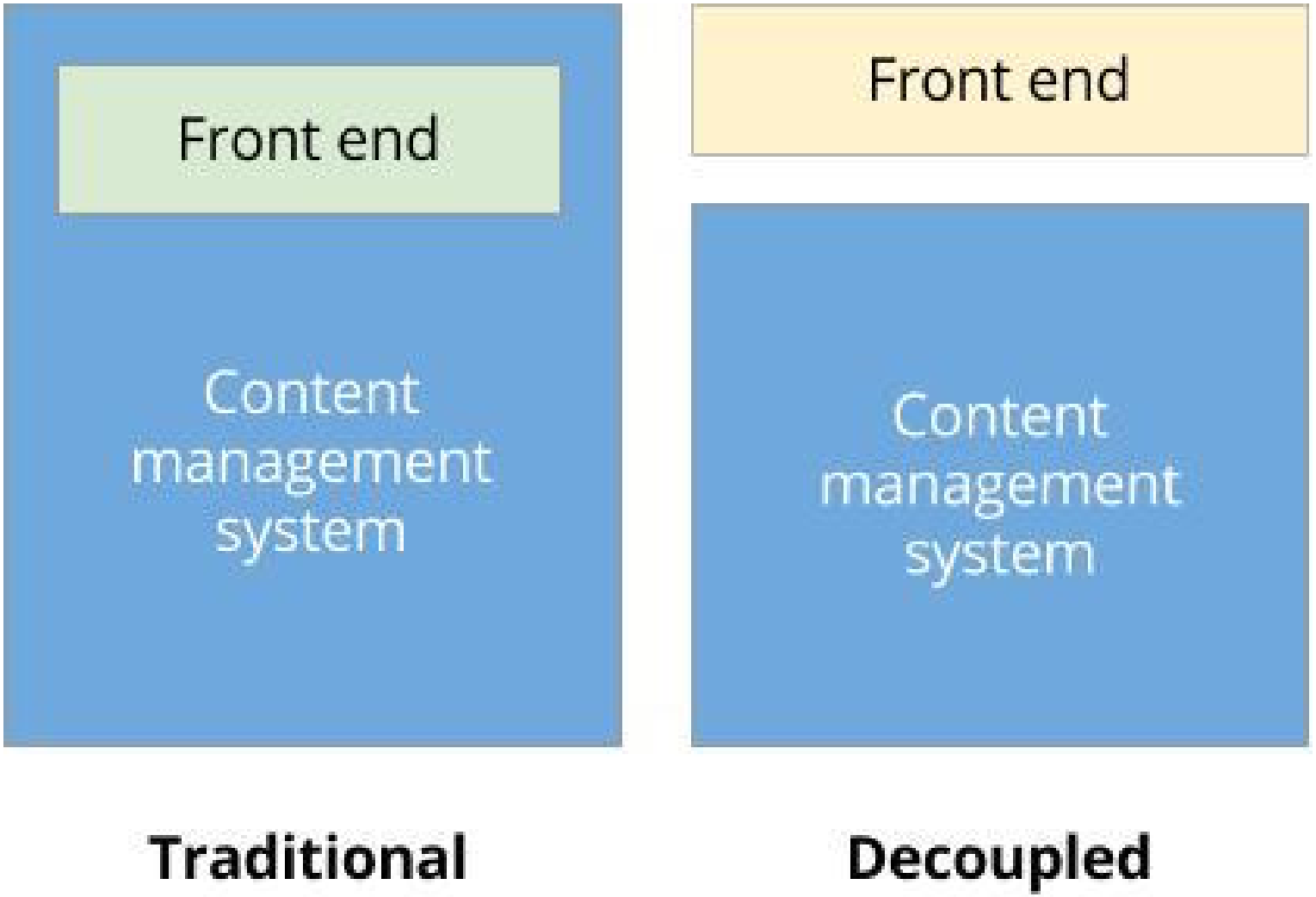
Applications côté client.

- Séparer les aspects client et serveur, évolutions à des rythmes différents.
- Gestion des interfaces avec des technologies Javascript : Ember, Angular, VueJS, React, etc.

Cycle de vie des contenus

- De plus en plus complexe.
- Accessibles via diverses technologies et dispositifs : sites web, applications mobiles natives, interfaces vocales ou conversationnelles, technologie portable (« wearables »), internet des objets.
- « Publish once, access everywhere. »

Drupal découplé, « headless »



Avantages et inconvénients

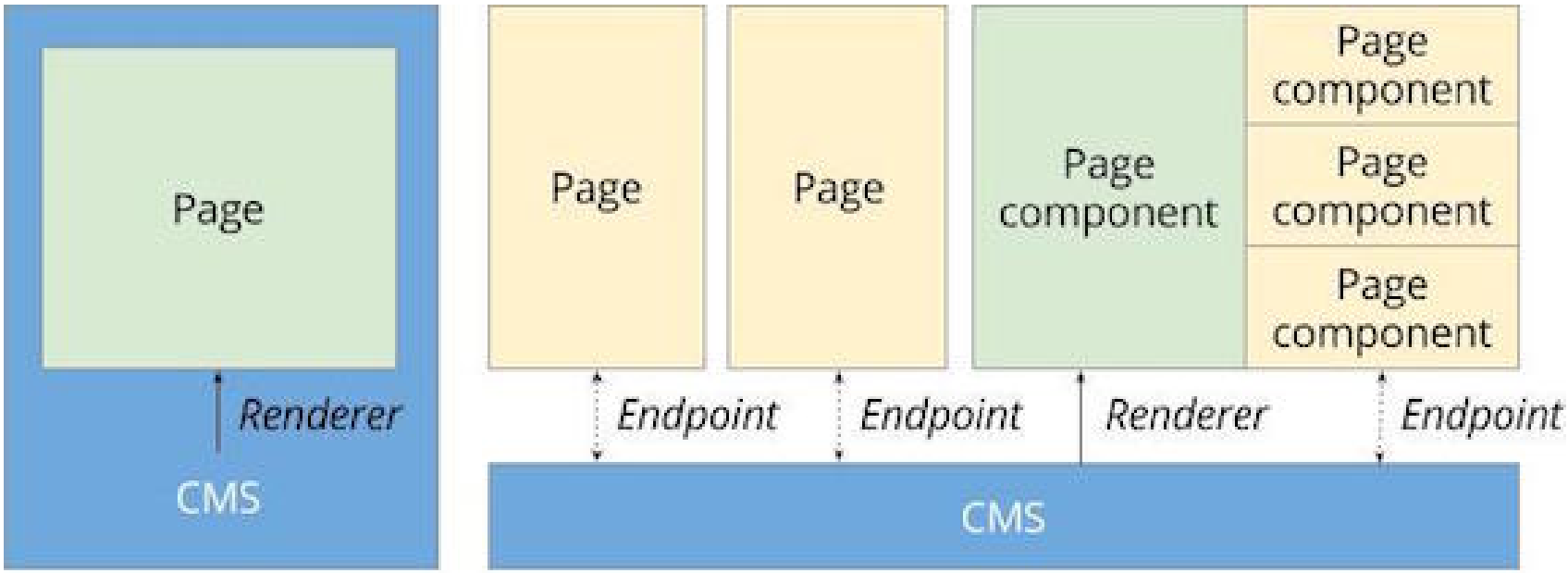
Avantages

- Travail séparé sur le front-end et le back-end : moins « monolithique » ?
- Rythmes d'évolution différents.
- Séparation des responsabilités.
- Réflexion centrée sur le contenu.
- Plus de flexibilité ?

Inconvénients

- Perte des fonctionnalités de mise en page, de la Form API...
- Thématiques de sécurité.
- Traduction des contenus
- Efficacité (allers-retours et bootstrap si mal pensé)
- Tracasseries concernant le référencement.
- Moins de flexibilité ?

Découplement progressif



Traditional

Progressively decoupled (many heads)

« Drupal is API-first, not API-only »

Drupal can be the brain behind all of your systems to deliver content everywhere

Representational state transfer (REST)

« Style d'architecture pour les systèmes hypermédia distribués, créé par Roy Fielding en 2000 dans le chapitre 5 de sa thèse de doctorat. »

- Client-serveur.
- *Sans état*.
- Gestion du cache.
- Interface uniforme, identification des ressources.

API RESTful :

- Basées sur HTTP.
- URL de base.
- Type MIME (auto-description).
- Verbes HTTP : GET, POST, PUT, DELETE...

JSON API

- JSON : mécanisme de sérialisation des données structurées (comme XML).
- JSON API : *spécification* détaillant le formatage d'une réponse en JSON.
- On expose des objets de manière générique et indépendante de la technologie sous-jacente : inutile de connaître Drupal pour exploiter une réponse.
- Gestion normalisée des relations entre entités (`relationships`).
- Le module « JSON API » pour Drupal « remplace » la couche de services web du cœur : il ne se sert que de *Serialization* (apparu après la sortie du cœur).

GraphQL

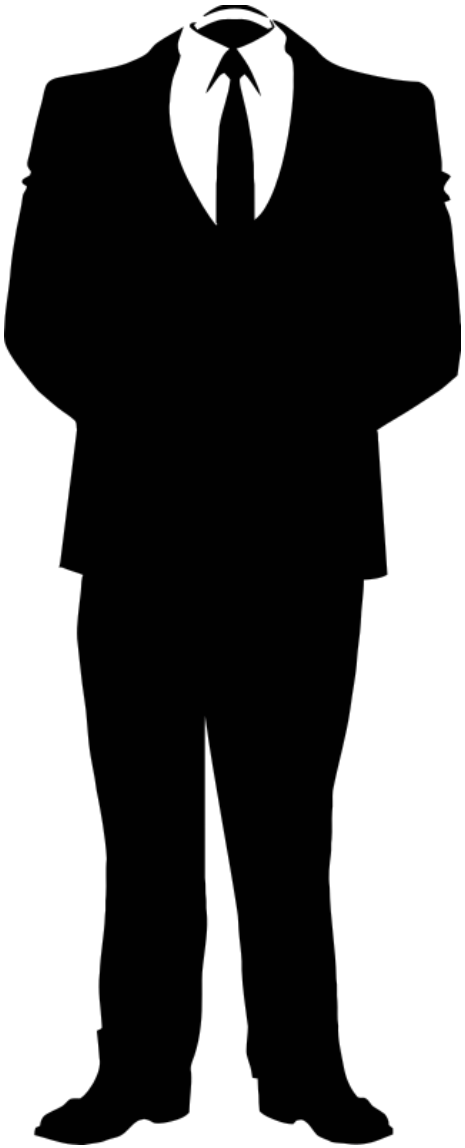
- Développé par Facebook et publié en 2015.
- Fonctionne via HTTP.
- Requêter à travers la structure des données (entités) d'un site
- REST 2.0 ?

Distributions et SDKs

Contenta Une distribution « API-first » proposée par la communauté (Mateu Aguiló Bosch & al.)

Reservoir Dans la même veine, moins de choix, mais plus guidés.

Waterwheel SDK (software development kit) pour s'interfacer avec un drupal sans tête, sans avoir à connaître son fonctionnement interne. Disponible en JavaScript et en Swift.



Merci !

Références (1/3)

- Buytaert, Dries, *Headless Drupal*, `https://dri.es/tag/headless-drupal` (consulté le 06/04/2018)
- Acquia, *Decoupled Drupal*, `https://www.acquia.com/fr/products-services/decoupled` (consulté le 06/04/2018)
- Georges, Simon, *Comment mettre en place un site Drupal « headless » ?*, `https://makina-corpus.com/blog/metier/2017/comment-mettre-en-place-un-site-drupal-headless` (consulté le 06/04/2018)

Références (2/3)

- Collectif, *Headless CMS*, `https://en.wikipedia.org/wiki/Headless_CMS` (consulté le 06/04/2018)
- Lander, Michael, *What is Headless Drupal ?*, `https://www.elevatedthird.com/news-insights/what-headless-drupal` (consulté le 06/04/2018)
- Collectif, *Decoupled CMS : Why "Going Headless" Is Becoming So Popular*, `https://pantheon.io/decoupled-cms` (consulté le 06/04/2018)

Références (3/3)

- Paris, Michael, *REST 2.0 Is Here and Its Name Is GraphQL*, `https://www.sitepoint.com/rest-2-0-graphql/` (consulté le 07/04/2018)
- Stubailo, Sashko, *GraphQL vs. REST*, `https://dev-blog.apollodata.com/graphql-vs-rest-5d425123e34b` (consulté le 07/04/2018)
- Buytaert, Dries, *GraphQL demo*, `https://www.youtube.com/watch?v=ZjDYg6NrAys` (consulté le 07/04/2018)