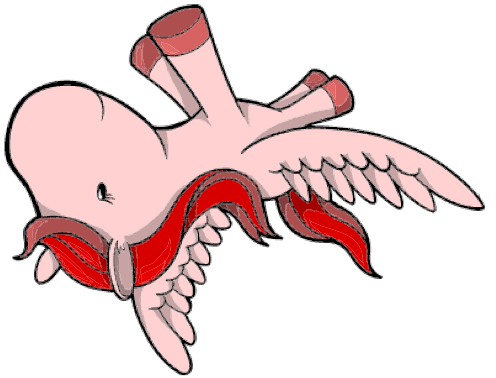
Dans l'enfer de la Bascule

Mathieu Pillard
@dioxmat



Libération.fr

Introduction

- Libération a basculé 90% de son site web en août 2011 (enfin... sans le back-office, voir conf de Souen tout à l'heure :)
- Passage d'un site « statique » (pré-génération de pages) en PHP à un site « dynamique » en Python / Django 1.3
- Quels enseignements techniques en tirer ? (Enfin, qu'est ce qu'on peut faire tenir en 12 minutes?)

Libération.fr

Benchmarks, mensonges et profiling

- Faites des benchs dans les conditions les plus réalistes possibles
- Utilisez vos logs pour faire des scénarios, déterminer ce qu'il faut optimiser en priorité, avoir une idée du volume de données, etc.
- Attention à django-debug-toolbar :
  - Avec tous les panels activés, ça représente beaucoup de HTML, et donc ça influe sur la vitesse de chargement
  - Panel de profiling buggé : il fait exécuter la view courante en double
  - Panels de SQL/Cache pratiques mais lents, en partie à cause des stackstraces
    - Même si le middleware est désactivé, juste si l'app est dans INSTALLED_APPS ! L'enlever en prod

Enseignements : Templates Django

- Le moteur de templates de Django est vieillissant, lourd et lent.
- On a choisi de l'utiliser quand même, pour être compatible avec la tonne de petites applications externes qu'on utilise
- Si vous n'avez pas cette contrainte, regardez du coté de Jinja2
- Pièges :
  - L'abus d'héritage de templates ou d'includes peut coûter très cher en performance : regarder du coté de django-pancake ou django-template-preprocessor
  - Ne pas oublier d'utiliser django.template.loaders.cached.Loader
- Les développeurs sont au courant des problèmes, et ça devrait être résolu... un jour

Libération.fr

Enseignements : SQL

- L'ORM de Django est sympa, mais parfois dangereux
- Il est facile de faire des requêtes inutiles ou coûteuses
- Utiliser django-debug-toolbar et applications du même genre, mais se méfier : l'optimisation de requêtes est un art délicat, car il dépend de la volumétrie, du hardware, etc.
- Toujours revérifier l'impact en production, via logs ou autre
- La configuration de la base de données est tout aussi importante que les requêtes envoyées dessus

Libération.fr

Enseignements : SQL, pièges classiques de l'ORM

- Construire des objets python est lourd. Utiliser .values(), .values_list()
- Essayer d'utiliser .only() et .defer()
- Les bases de données n'aiment pas les jointures dans tous les sens ; L'argument depth de select_related() est souvent contre-productif
- Attention à l'héritage « concret », qui peut provoquer beaucoup de jointures. Toujours se poser la question de si abstract=True ne serait pas plus indiqué
- Dé-normaliser n'est pas un crime, il faut juste ne pas en abuser.
- Votre base vous remerciera si elle n'a pas besoin de faire de distinct()
- Minimiser les count(), les cacher

Libération.fr

Enseignements : Pagination

- Le Paginator de base de django peut faire très mal :
  - Fait un count() obligatoire
  - Utilise OFFSET/LIMIT pour parcourir la base jusqu'au bout : plus on va loin dans les pages, plus la base doit travailler
- Solutions possibles :
  - InfinitePaginator (pas de count(), mais en contrepartie pas de détail du nombre de pages)
  - clamp_pagination() pour limiter la pagination à un nombre de pages pré-défini
  - Seek Based Pagination (pas de OFFSET, fait un LIMIT <x> combiné à un filtre sur la valeur de la pk ou autre champ indexé)

Libération.fr

Enseignements : Query Caching

- Diverses applis permettent de cacher de manière transparente des QuerySets
- « There are only two hard things in Computer Science: cache invalidation and naming things. »
- 2 stratégies différentes pour l'invalidation:
  - Invalidation très fine, dépendante des champs utilisés, voire des objets retournés par une requête (django-cache-machine, django-cachebot)
    - Quelques limitations (dépend de l'appli, aucune n'est totalement transparente)
    - Nécessite un vrai contrôle sur les modèles pour être certain de ne pas avoir d'effets de bord
  - Invalidation très agressive, dès qu'une entrée dans une table liée à une requête change (johnny-cache)
    - Totalement transparent pour le développeur
    - Suffit dans la majorité des cas (tant que vous avez plus de lectures que d'écritures...), mais attention aux « tempêtes d'invalidation »
- Attention à ne pas produire de requêtes impossibles à cacher, type .filter(champ_lte=datetime.now()) ... Utiliser F() ou .extra() !

Et voilà, c'est déjà fini !

- Un grand merci à :
  - Tous les développeurs de Django
  - Tous ceux qui ont participé de près ou de loin à notre bascule
  - Tous les développeurs de plus ou moins petites applications Django qu'on ré-utilise
  - Tous ceux qui liront ces slides jusqu'au bout
  - Etc, etc

Libération.fr