Vider le cache django en developpement aprés chaque changement de source .

> Vider le cache django en developpement aprés chaque changement de source .

- Compensation statique fichier python manage.py collectstatic --noinput --clear . Cela nettoiera la statique à l'avance.
- Effacer le cache du navigateur
- ajouter une chaîne aléatoire après le fichier JS include, E. g jquery.js? rand = 23423423, avec chaque charge.

> vous devez détruire le cache du navigateur. Cette balise template affichera un uuid basé sur le temps quand DEBUG=True .
> Sinon, il cherchera une variable d'environnement PROJECT_VERSION . Si ce n'est pas trouvé,
> il affichera un numéro de version statique.

```
rom django import template
from django.conf import settings

register = template.Library()

@register.simple_tag(name='cache_bust')
def cache_bust():

    if settings.DEBUG:
        version = uuid.uuid1()
    else:
        version = os.environ.get('PROJECT_VERSION')
        if version is None:
            version = '1'

    return '__v__={version}'.format(version=version)
```

> vous utiliseriez dans un modèle comme celui-ci:

```
{% load cache_bust %}

<link rel="stylesheet" href="{% static "css/project.css" %}?{% cache_bust %}"/>
```

et voici le résultat:

```
<link rel="stylesheet" href="/static/css/project.css?__v__=7d88de4e-7258-11e7-95a7-0242ac1300
```

> J'ai aussi lutté avec ce problème pendant des heures. J'ai essayé d'injecter des chaînes aléatoires en utilisant Javascript, mais cette méthode semble stupide et laide. Une façon possible de traiter ce problème est d'introduire une balise personnalisée. Voir ce document pour plus de détails:

plus précisément, vous devez créer un paquet appelé templatetags dans toutes les applications que vous avez créées (ou créer un nouveau paquet si vous le souhaitez). Et vous créez n'importe quel fichier dans ce paquet, et écrivez quelque chose comme ceci:

```
from django import template
from django.utils.crypto import get_random_string
from django.templatetags import static

register = template.Library()


class StaticExtraNode(static.StaticNode):

    def render(self, context):
        return super().render(context) + '?v=' + get_random_string(32)


@register.tag('static_no_cache')
def do_static_extra(parser, token):
    return StaticExtraNode.handle_token(parser, token)


def static_extra(path):
    return StaticExtraNode.handle_simple(path)
```

alors vous pouvez utiliser la balise {% static_no_cache '.../.../path...' %} pour créer un chemin avec des arguments aléatoires!

j'espère que cela aidera!

> pour rafraîchir les fichiers statiques, vous devez lancer python manage.py collectstatic à nouveau.

> si rien d'autre ne fonctionne, recherchez le nom du fichier dans le projet et recherchez une copie inattendue. Si vous avez sauvegardé au mauvais endroit (application différente) à un moment donné, ou découpé une nouvelle application à partir d'une ancienne, la priorité de charge peut vous jouer des tours.

> au lieu d'utiliser des solutions compliquées, vous pouvez ajouter des paramètres supplémentaires à vos includes dans les modèles.

pour statique comprend:

```
<script src="{% static 'js/polls/polls.js' %}?version=1"></script>
Pour direct comprend:

<link rel="stylesheet" type="text/css" href="/site_media/css/style.css?version=1" />
noter le ?version=1 dans le code. Chaque fois que vous modifiez le fichier css/js, changez ce

Et si vous voulez éviter la mise en cache à tout cela pour une raison inconnue, vous pouvez u

<link rel="stylesheet" type="text/css" href="/site_media/css/style.css?{% now "U" %}" />
```

> si vous ne voulez pas rafraîchir le cache du navigateur à chaque fois que vous changez vos fichiers CSS et JavaScript, ou tout en stylisant des images, vous devez définir STATIC_URL de façon dynamique avec un composant de chemin variable. Avec L'URL qui change dynamiquement, chaque fois que le code est mis à jour, le navigateur du visiteur va forcer le chargement de tous les nouveaux fichiers statiques non mis en cache. Dans cette recette, nous allons définir un chemin dynamique pour STATIC_URL en utilisant le temps de la dernière édition dans os.

```
import os
from datetime import datetime

def get_file_changeset(absolute_path):
    timestamp = max(map(lambda x: os.path.getmtime(x[0]), os.walk(os.path.join(absolute_path,
    try:
        timestamp = datetime.utcfromtimestamp(int(timestamp))
    except ValueError:
        return ""
    changeset = timestamp.strftime('%Y%m%d%H%M%S')
    return changeset
```

```
et changement suivant dans votre SETTINGS :

from utils.misc import get_file_changeset
STATIC_URL = "/static/%s/" % get_file_changeset(BASE_DIR)
```

> Comment cela fonctionne:
>
> La fonction get_file_changeset() prend le répertoire absolute_path comme paramètre et appelle le os.path.getmtime() à chaque fichier dans chaque répertoire imbriqué et trouve le dernier fichier édité (avec son temps d'édition max). L'horodatage est analysé; converti en chaîne de caractères comprenant l'année, le mois, le jour, l'heure, les minutes et secondes; retournées; et incluses dans la définition de STATIC_URL .

[id]### Conclusion

> avec ceci vous devez recharger dev server chaque fois que vous éditez vos fichiers statiques.
> ou alors
> votre navigateur cache des images et des fichiers (javascript inclus). Tout d'abord, effacez juste vos images et fichiers mis en cache. Puis utilisez le mode incognito dans chrome ou la navigation privée dans firefox pendant que vous faites des changements à votre .les fichiers js donc vous les voyez instantanément après un rafraîchissement de page

```
1.
django django-staticfiles
```

- installation VM
  - partition disque
  - install virtualBox
  - config VM
- base de donnee
  - install base mysql
  - import data

PP