django-gcframe Documentation

Release 1.1

Ben Spaulding

Sep 27, 2017

Contents

CHAPTER 1

Overview

`django-gcframe` is a set of Django utilities for working with Google Chrome Frame.

Specifically, the package provides configurable middleware and decorators for adding and removing the `X-UA-Compatible` HTTP header in view responses. Said header conditionally activates Google Chrome Frame when installed on Microsoft Internet Explorer, and conditionally enables a compatibility mode when GCF (Google Chrome Frame) is not activated.

This package does not install Google Chrome Frame, though in the future it may have features that would promote installation.

CHAPTER 2

Contents

Getting Started

Overview

Google Chrome Frame is a plug-in for Internet Explorer that causes the browser to behave like Google Chrome, using the Webkit rendering engine.

Because many sites are made to work for Internet Explorer and rely on its proprietary or broken technologies, GCF is not activated by default. Rather, it is activated when the requested resource has a particlar `X-UA-Compatible` HTTP response header.

This application does not install the GCF plug-in. It merely allows for simple and configurable sending of the necessary header to activate the plug-in for those Internet Explorer users with it installed, and for those who don't it makes use of the highest compatibility mode available.

For further information on GCF, see the developer documentation, particularly the Getting Started article. More information regarding IE compatibility modes can be found on MSDN.

Requirements

`django-gcframe` requires Python 2.6 or newer and Django 1.3 or newer.

Installation & Usage

To install this package, run the following command at the root of the package directory:

```
python setup.py install
```

If you have the Python `easy_install` utility available, you can also type the following to download and install in one step:

```
easy_install django-gcframe
```

Or if you're using `pip` (and you should be):

```
pip install django-gcframe
```

Or if you'd prefer you can simply place the included `gcframe` directory somewhere on your Python path, or symlink to it from somewhere on your Python path; this is useful if you're working from a Git checkout.

You can then begin using the middleware and/or decorators in your Django project.

Middleware

To send the default `X-UA-Compatible` HTTP header site-wide simply add the middleware to your `MIDDLEWARE_CLASSES` setting:

```
MIDDLEWARE_CLASSES = (
    ...
    'gcframe.middleware.GoogleChromeFrameIEMiddleware',
    ...
)
```

For further details, see *Middleware*.

Decorators

There is a decorator for adding or overriding the `X-UA-Compatible` HTTP header on individual views. There is also one for removing it from individual views when the middleware is adding it site-wide.

For further details, see *Decorators*.

Configuration

There are two settings that allow for modification of the `X-UA-Compatible` HTTP header. Find them at *Settings*.

Support

Bugs and feature requests can be submitted to the GitHub issue tracker.

Features Not Yet Included

There are a few things that this middleware could do which it does not. This is because I know of no data indicating that the features are worth the time they would take to implement, and they could be difficult to do properly because they involve inspecting the User Agent string.

- Sending the header only to IE.
- Requiring all IE users to have GCF installed.
- Requiring only IE users matching the condition of activation to have GCF installed.

I am not opposed to these features, I just don't feel like implementing them right now. Feel free to do it yourself or try to convince me to.

There is one more feature that I do think could be useful. It would be a set of tools that would allow a site to gracefully prompt IE users to install GCF. It would likely involve some User Agent string inspection, a context processor that would set a context variable to indicate the state of the IE, and session fu that would allow an IE user to be prompted once to install the plug-in. If you are interested in pursuing this yourself, see the Chrome developer guide for detecting and prompting to install.

Development

To run tests:

```
django-admin.py test gcframe --settings="gcframe.tests.settings"
```

For more information on tests, and for building the documentation, please see *Development*.

Middleware

Overview

The `GoogleChromeFrameIEMiddleware` middleware adds an `X-UA-Compatible` HTTP header to the response of all views. When using default settings (which you should), this header activates GCF in all versions of Internet Explorer that have it installed. Those that do not have GCF installed will instead be directed to use the highest compatibility mode available to them.

Tip: Views can be exempt from the header addition by using the `gcframe_exempt()` decorator.

Installation

Install the middleware by adding it your `MIDDLEWARE_CLASSES` setting:

```
MIDDLEWARE_CLASSES = (
    ...
    'gcframe.middleware.GoogleChromeFrameIEMiddleware',
    ...
)
```

Put it somewhere in the middle-ish of the list. An exact ordering position is not necessary (to my knowledge), but putting in the middle will at least ensure that it's not outside of other middleware that require being near the start or end.

Tip: The content of the `X-UA-Compatible` header can be altered using by overriding settings in *`gcframe.settings`*.

Decorators

Overview

These decorators allow you to apply or remove the `X-UA-Compatible` HTTP header on individual views.

Available Decorators

gcframe

This decorator applies the `X-UA-Compatible` HTTP header to individual views, rather than site-wide as *GoogleChromeFrameIEMiddleware* does.

```
from gcframe.decorators import gcframe

@gcframe()
def some_view(request):
    ...
```

Note: The trailing `()` are required on this decorator, even when no arguments are passed.

It accepts the key-word arguments `compat_mode` and `act_method`. These arguments correspond to *GCF_IE_COMPATIBILITY_MODE* and *GCF_IE_ACTIVATION_METHOD*, respectively.

Using these arguments is useful when you wish to set the `X-UA-Compatible` HTTP header to something different than the default that is being used by this decorator on other views or by the middleware site-wide.

```
from gcframe.decorators import gcframe

@gcframe(act_method='IE7')
def some_view(request):
    ...
```

gcframe_exempt

This decorator instructs the *GoogleChromeFrameIEMiddleware* to **not** set the `X-UA-Compatible` HTTP header.

```
from gcframe.decorators import gcframe_exempt

@gcframe_exempt
def some_view(request):
    ...
```

Obviously this will only affect change when the middleware is installed. It is harmless if the middleware is not installed.

Settings

Overview

There are a couple of configurable settings that define the content of the `X-UA-Compatible` HTTP response header. Realistically, virtually no one will need to modify these. But if you do, simply set a custom value for them in your `settings.py`.

Available Settings

GCF_IE_COMPATIBILITY_MODE

The default value is `'Edge'`. This means that if GCF is not activated IE (Internet Explorer) will use the highest compatibility mode available. Other values include:

- `None`
- `'5'`
- `'6'`
- `'7'`
- `'EmulateIE7'`
- `'8'`

A value of `None`, will turn off compatibility mode changing. Other values correspond to various Internet Explorer browser modes. Virtually everyone will be best served by using the default value of `'Edge'`.

See also:

More information on compatibility modes can be found in the MSDN article META Tags and Locking in Future Compatibility.

GCF_IE_ACTIVATION_METHOD

Default value is `'1'`. This means that all versions of IE with GCF installed will activate it. Other options include:

- `'IE6'`
- `'IE7'`
- `'IE8'`

These values activate GCF only on the named IE version and older. Newer versions will behave according to the value of *GCF_IE_COMPATIBILITY_MODE*. Again, few people will need to change this setting. A value of `'1'` will likely serve you best.

See also:

The GCF developer documentation has more information regarding activation methods.

Development

Source Code

gcframe source code is managed using Git, and can be found on GitHub. Feel free to clone, fork, and contribute.

Documentation

The documentation is written in plain text, viewable practially anywhere. An HTML version of the docs can be found online at Read the Docs. If you want to build a local version of these, you can install Sphinx, and then from the doc directory in this repository, run:

```
make html
```

You will find the built docs in the docs/_build/html directory.

Tests

gcframe has a decent test suite, which can and will improve in time.

Current build status can be found at Travis CI.

To run tests, be sure Django is installed, then run:

```
django-admin.py test gcframe --settings="gcframe.tests.settings"
```

If you have not installed gcframe, but are working from a Git checkout, you will need to either have it on your PYTHONPATH or run the above command from the root of the repository.

Note: In order for gcframe tests to run in your project, you will need to add gcframe to your INSTALLED_APPS. (Mentioned here because no other gcframe functionality requires this.)

CHAPTER 3

Indices, etc.

- genindex
- modindex
- search

Python Module Index

g

Index

G