CSPObject Documentation

Release 0.1

Adam Brenecki

May 07, 2018

Contents:

CSPObject is an object-oriented representation of Content Security Policies. You can create CSP objects either using keyword arguments or by passing a string to the `.parse` class method, and convert back to strings by calling `str()` on them.

```
from cspobject import CSPObject

csp1 = CSPObject(
    default_src=("'self'",),
    img_src=("'self'", "https://flickr.com"),
)
assert str(csp1) == "default-src 'self'; img-src 'self' https://flickr.com"

csp2 = CSPObject.parse("default-src 'self'; img-src 'self' https://flickr.com")
assert csp2 == csp1
```

This might not seem very useful by itself, but you can also **combine** CSP objects together. Let's say we've started building a website, and we want to allow any kind of resource from `'self'` and nowhere else.

```
>>> from cspobject import CSPObject
>>> my_site = CSPObject(default_src=("'self'",))
```

But then, we decide we want to load fonts from Typekit. We can make a separate CSP object which allows Typekit to run JavaScript and put CSS, fonts and images on our site...

```
>>> typekit = CSPObject(
...     script_src=("use.typekit.net",),
...     style_src=("'unsafe-inline'", "use.typekit.net"),
...     font_src=("use.typekit.net", "fonts.typekit.net"),
...     img_src=("p.typekit.net",)
... )
```

...and combine them using the union (`|`) operator.

```
>>> my_site | typekit
CSPObject(
    default_src={"'self'"},
    font_src={'use.typekit.net', 'fonts.typekit.net', "'self'"},
    img_src={'p.typekit.net', "'self'"},
    script_src={'use.typekit.net', "'self'"},
    style_src={"'unsafe-inline'", 'use.typekit.net', "'self'"}
)
```

The resulting CSP object is written to allow through any resources that would also have been allowed through by any of the CSP objects that were passed in. For example, the `my_site` policy allows scripts from `'self'`, because `'self'` is in `default-src`, and no `script-src` is present. Here, CSPObject has added `'self'` to `script-src`, so that scripts from `'self'` are still allowed.

Note: If you're familiar with set theory, you can think of CSP objects as sets of allowed requests, and the union operator as taking the union of those sets. If you're not familiar with set theory, don't worry, you don't need to be.

Let's say we wanted to add Google Analytics as well. We could continue to use `|`, but we can also use the `.union` class method, which can combine more than two CSPObjects at once, and also accepts strings.

```
>>> combined = CSPObject.union(my_site, typekit, "script-src www.google-analytics.com;
↪ img-src www.google-analytics.com")
>>> str(combined)
```

```
"default-src 'self'; font-src use.typekit.net fonts.typekit.net 'self'; img-src p.
↪typekit.net www.google-analytics.com 'self'; script-src use.typekit.net www.google-
↪analytics.com 'self'; style-src 'unsafe-inline' use.typekit.net 'self'"
```

CHAPTER 1

Indices and tables

- genindex
- modindex
- search