Integrating JavaScript with Python Web Frameworks

Jen Zajac (http://www.littlegreenpixel.com), @jenofdoom

Slide notes:

- *who am I?*
- *frontend dev with some pretty rusty backend dev knowledge*
- *have been building a giant django/angularjs web app for a gov ministry for the last year*
- *so that's my particular area of expertise, am trying to keep this talk a bit more general though*

Why do we want to use JS?

- To offload some work to the client?
- To make the app or site more responsive to the user?

Slide notes:

- *making your site or app more responsive is the killer feature*
- *ability to update smaller parts of the page*
- *no need for a reload*

How can we use JS to augment our Python app or site?

- Light interactivity
  - e.g. as you type form validation on one page
  - just use jQuery
- App-like behaviour
  - e.g. client side routes (HTML5 push state)
  - at this point we want a JS framework

Slide notes:
JS frameworks are a good idea because they provide structure for your code and templating so you don't end up with lots of content strings and markup in your code.

Okay, so I want to use a JS framework with a Python framework, what things do I need to think about?

Many JS libraries use the {{ somevar }} pattern in their templating

Many Python libraries *also* use the {{ somevar }} pattern in their templating

Some Python templating engines should be fine:

- templetor (web.py)
- mako (pyramid, bottle)
- chameleon (pyramid)
- cheetah (bottle)
- genshi (cherrypy)

Slide notes:
On the JS side, backbone should work fine with anything as its template delimiters are configurable and in fact don't use curly braces as default

Some, not so fine:

- jinja2 (bottle, flask, many others)
- SimpleTemplate Engine (bottle)
- Django's templating (django)

Fortunately, there are some workarounds

Slide notes:
Not for bottle's simpletemplate engine though - you'll have to use an alternative templating engine

jinja2

Use **{% raw %} (http://jinja.pocoo.org/docs/templates/#escaping)** blocks

```
{% raw %}
    <p>{{ myVar }} is lovely.</p>
{% endraw %}
```

Django 1.5+

Use **{% verbatim %} (https://docs.djangoproject.com/en/dev/ref/templates/builtins/#std:templatetag-verbatim)** blocks

```
{% verbatim %}
    <p>{{ myVar }} is lovely.</p>
{% endverbatim %}
```

Django 1.4 and below

Not as straightforwards... use a custom template tag?

```
@register.filter
def curly(value):
    """Wraps string in {{ }}"""
    return '{{ ' + value + ' }}'


{{ 'myVar'|curly }}
```

Slide notes:
This is pretty nasty...

Alternatively, sidestep the problem altogether:

- Use partial templates that get included by the JS framework
- Your Python code never sees them
- Or, even more drastic: no Python templates at all

Partial JS templates

- Keep them with your other static files (like CSS and JS files)
- Most JS frameworks let you nest templates in quite a sophisticated manner
- Watch out for CORS issues on older flavours of IE - you may need to ensure you serve static from same domain as your site

---

Slide notes:
CORS = cross origin resource sharing. See http://caniuse.com/cors

Django + Angular partial template example

```
<form method="POST" action="{% url my_view %}">
    {% csrf_token %}
    <div
        ng-include
        src="staticURL + 'partials/partial.html'"
        ng-init="initData={state: '{% myVar %}'}"
    ></div>
</form>
```

Okay, so I'm now either serving Python templates that do very little, or I'm not serving them at all... what's supplying data to the JS framework?

RESTful API

Slide notes:

- *seperation of concerns*
- *easy to cache aggressively*
- *standardised pattern for third parties to interact with*
- *many js frameworks have built in ability to talk to REST services*

Popular API libraries

- **Tastypie (http://tastypieapi.org/)** (Django)
- **Django REST framework (http://django-rest-framework.org/)** (Django...)
- **Django-piston (https://bitbucket.org/jespern/django-piston/wiki/Home)** (Django...)
- **Flask-RESTful (http://flask-restful.readthedocs.org/en/latest/)** (Flask)
- **mimerender (https://github.com/martinblech/mimerender)** (webapp2, web.py, Flask, Bottle)
- **Eve (http://python-eve.org/)** (Flask + MongoDB only)

Slide notes:
Of the Django options, Django REST framework is possibly the sleekest.
Tastypie can be a little curmudegonly but is quite powerful

Roll your own

Some frameworks offer some api functionality out of the box:

- **web2py (http://web2py.com/books/default/chapter/29/10#HTML,-XML,-and-JSON)**
- **cherrypy (http://docs.cherrypy.org/dev/progguide/REST.html)**

Or write your own:

- build on top of light framework e.g. flash, bottle
- or use Django's **class-based views (http://ccbv.co.uk/)**

Slide notes:
Of the Django options, Django REST framework is possibly the sleekest.
Tastypie can be a little curmudegonly but is quite powerful

Some other things to consider

Watch out for “features” of API frameworks burning you

Slide notes:
Check everything is off by default & turn on the things you need (XML and YAML are kinda insecure by default).

You need to make sure you are sending **CSRF protection tokens (https://docs.djangoproject.com/en/1.5/ref/contrib/csrf/#ajax)** along with any POSTs you are doing

---

Slide notes:
Need to check that the POST originated from the right place.Django allows you to set a X-CSRFToken header. make sure you clean incoming data propely - if using django can use django forms for this, still.

Some JS frameworks have opinions about trailing slashes on URLs

---

Slide notes:
Django has trailing slashes on by default, Angular always strips them. Have to config Django (and Tasyie) to not require them for Angular's RESTful service interation to work.

If you know have a bunch of application logic in your JS, that JS **needs** to be unit tested.

Slide notes:
*Look at **Jasmine (http://pivotal.github.io/jasmine/)** as a unit testing framework. This has an HTML based test runner which you can set up to be scraped by e.g. Selenium to integrate with the rest of your tests.*

Keep your coding conventions nicely segregated - this will help you mentally flip between languages

---

Slide notes:
E.g. use camelCase for js and underscores for Python. Use hinters and linters to maintain good code quality.

If you have small bits of data you want to expose to your JS code without necessitating an API call, use the data attribute on the HTML or body element.

Slide notes:
E.g. use camelCase for js and underscores for Python. Use hinters and linters to maintain good code quality.

Using data attributes, example with Django template

```
<html data-static="{% get_static_prefix %}">
```

Scrape this with some JS:

```
var staticURL = document.documentElement.getAttrib
```

Thanks for listening