chewse

Straightening Out AngularJS with Python

chewse

30-SECOND ANGULAR

- JavaScript MV* framework

chewse

30-SECOND ANGULAR

- JavaScript MV* framework
- Extends HTML vocabulary

chewse

30-SECOND ANGULAR

- JavaScript MV* framework
- Extends HTML vocabulary
- Getting kinda popular

30-SECOND PYTHON

- Pretty honking great language

chewse

30-SECOND PYTHON

- Pretty honking great language
- Django, Pyramid, Flask, etc.

chewse

30-SECOND PYTHON

- Pretty honking great language
- Django, Pyramid, Flask, etc.
- Server-side MV* frameworks

chewse

1. SERVE ANGULAR

WHAT WE SERVE

- Base HTML page

BASE HTML

```
<html ng-app="app">
<body ng-controller="HumanCtrl">

  <input type="text" ng-model="humanName" placeholder="Name">
  <p>Hi, {{ humanName }}!</p>

  <script src="http://foo.com/angular.js"></script>
  <script src="http://foo.com/app.js"></script>
</body>
</html>
```

BASE HTML

```
<html ng-app="app">
<body ng-controller="HumanCtrl">

  <input type="text" ng-model="humanName" placeholder="Name">
  <p>Hi, {{ humanName }}!</p>

  <script src="http://foo.com/angular.js"></script>
  <script src="http://foo.com/app.js"></script>
</body>
</html>
```

chewse

BASE HTML

```
<html ng-app="app">
<body ng-controller="HumanCtrl">

  <input type="text" ng-model="humanName" placeholder="Name">
  <p>Hi, {{ humanName }}!</p>

  <script src="http://foo.com/angular.js"></script>
  <script src="http://foo.com/app.js"></script>
</body>
</html>
```

chewse

BASE HTML

```
<html ng-app="app">
<body ng-controller="HumanCtrl">

  <input type="text" ng-model="humanName" placeholder="Name">
  <p>Hi, {{ humanName }}!</p>

  <script src="http://foo.com/angular.js"></script>
  <script src="http://foo.com/app.js"></script>
</body>
</html>
```

chewse

BASE HTML

```
<html ng-app="app">
<body ng-controller="HumanCtrl">

  <input type="text" ng-model="humanName" placeholder="Name">
  <p>Hi, {{ humanName }}!</p>

  <script src="http://foo.com/angular.js"></script>
  <script src="http://foo.com/app.js"></script>
</body>
</html>
```

BASE HTML

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView

urlpatterns = patterns('',
    url(r'^.*$', TemplateView.as_view(template_name='app.html')),
)
```

BASE HTML

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView

urlpatterns = patterns('',
    url(r'^.*$', TemplateView.as_view(template_name='app.html')),
)
```

WHAT WE SERVE

- Base HTML page
- Template partials

TEMPLATE PARTIALS

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView

urlpatterns = patterns('',
    url(r'^rendered-partials/(?P<template_name>.*)$',
        'render_partial'),
    url(r'^.*$', TemplateView.as_view(template_name='app.html')),
)
```

TEMPLATE PARTIALS

```
from django.shortcuts import render

def render_partial(request, template_name=None):
    template_name = 'partials/{}'.format(template_name)
    context = {
        'you-are-a-unique-snowflake': True,
    }
    return render(request, template_name, context)
```

TEMPLATE PARTIALS

```
from django.shortcuts import render

def render_partial(request, template_name=None):
    template_name = 'partials/{}'.format(template_name)
    context = {
        'you-are-a-unique-snowflake': True,
    }
    return render(request, template_name, context)
```

TEMPLATE PARTIALS

```
from django.shortcuts import render

def render_partial(request, template_name=None):
    template_name = 'partials/{}'.format(template_name)
    context = {
        'you-are-a-unique-snowflake': True,
    }
    return render(request, template_name, context)
```

chewse

TEMPLATE PARTIALS

```
from django.shortcuts import render

def render_partial(request, template_name=None):
    template_name = 'partials/{}'.format(template_name)
    context = {
        'you-are-a-unique-snowflake': True,
    }
    return render(request, template_name, context)
```

chewse

WHAT WE SERVE

- Base HTML page
- Template partials
- Application API

2. APPLICATION API

OUR API

- Clean, RESTful JSON API

CLEAN API

```
from rest_framework import viewsets
from human.models import Human

class HumanViewSet(viewsets.ModelViewSet):
    model = Human
```

CLEAN API

```
from rest_framework import viewsets
from human.models import Human

class HumanViewSet(viewsets.ModelViewSet):
    model = Human
```

chewse

CLEAN API

```
from rest_framework import viewsets
from human.models import Human

class HumanViewSet(viewsets.ModelViewSet):
    model = Human
```

CLEAN API

```
from django.conf.urls import patterns, include, url
from rest_framework import routers

router = routers.DefaultRouter()
router.register(r'humans', HumanViewSet)
urlpatterns = patterns('',
    url(r'^', include(router.urls)),
)
```

chewse

CLEAN API

```
from django.conf.urls import patterns, include, url
from rest_framework import routers

router = routers.DefaultRouter()
router.register(r'humans', HumanViewSet)
urlpatterns = patterns('',
    url(r'^', include(router.urls)),
)
```

chewse

CLEAN API

```
from django.conf.urls import patterns, include, url
from rest_framework import routers

router = routers.DefaultRouter()
router.register(r'humans', HumanViewSet)
urlpatterns = patterns('',
    url(r'^', include(router.urls)),
)
```

OUR API

- Clean, RESTful JSON API
- And the other parts...

CLEAN...ISH API

CLEAN...ISH API

- Oh you wanted auth?

CLEAN...ISH AUTH

```
class UserViewSet(viewsets.ModelViewSet):
    @action()
    def login(self, request):
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(username=username, password=password)
        if user is not None and user.is_active:
            login(request, user)
            serializer = self.serializer_class(user)
            return Response(serializer.data)
        return Response('Fail', status=status.HTTP_403_FORBIDDEN)
```

chewse

CLEAN...ISH AUTH

```
class UserViewSet(viewsets.ModelViewSet):
    @action()
    def login(self, request):
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(username=username, password=password)
        if user is not None and user.is_active:
            login(request, user)
            serializer = self.serializer_class(user)
            return Response(serializer.data)
        return Response('Fail', status=status.HTTP_403_FORBIDDEN)
```

chewse

CLEAN...ISH AUTH

```
class UserViewSet(viewsets.ModelViewSet):
    @action()
    def login(self, request):
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(username=username, password=password)
        if user is not None and user.is_active:
            login(request, user)
            serializer = self.serializer_class(user)
            return Response(serializer.data)
        return Response('Fail', status=status.HTTP_403_FORBIDDEN)
```

chewse

CLEAN...ISH AUTH

```
class UserViewSet(viewsets.ModelViewSet):
    @action()
    def login(self, request):
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(username=username, password=password)
        if user is not None and user.is_active:
            login(request, user)
            serializer = self.serializer_class(user)
            return Response(serializer.data)
        return Response('Fail', status=status.HTTP_403_FORBIDDEN)
```

CLEAN...ISH API

- Oh you wanted auth?
- Oh you wanted permissions?

CLEAN...ISH PERMISSIONS

```
class AccountPermission(permissions.BasePermission):
    def has_permission(self, request, view):
        # Allow all access to admin users
        if request.auth and request.auth.is_admin:
            return True

        # Allow global read/write permission where option is set
        if getattr(view, 'global_full_permission', False):
            return True

        # Allow global read permission where option is set
        read = getattr(view, 'global_read_permission', False)
        if read and request.method in permissions.SAFE_METHODS:
            return True
```

CLEAN...ISH PERMISSIONS

```
class AccountPermission(permissions.BasePermission):
    def has_permission(self, request, view):
        # Allow all access to admin users
        if request.auth and request.auth.is_admin:
            return True

        # Allow global read/write permission where option is set
        if getattr(view, 'global_full_permission', False):
            return True

        # Allow global read permission where option is set
        read = getattr(view, 'global_read_permission', False)
        if read and request.method in permissions.SAFE_METHODS:
            return True
```

chewse

CLEAN...ISH PERMISSIONS

```
class AccountPermission(permissions.BasePermission):
    def has_permission(self, request, view):
        # Allow all access to admin users
        if request.auth and request.auth.is_admin:
            return True

        # Allow global read/write permission where option is set
        if getattr(view, 'global_full_permission', False):
            return True

        # Allow global read permission where option is set
        read = getattr(view, 'global_read_permission', False)
        if read and request.method in permissions.SAFE_METHODS:
            return True
```

CLEAN...ISH PERMISSIONS

```
class AccountPermission(permissions.BasePermission):
    def has_permission(self, request, view):
        # Allow all access to admin users
        if request.auth and request.auth.is_admin:
            return True

        # Allow global read/write permission where option is set
        if getattr(view, 'global_full_permission', False):
            return True

        # Allow global read permission where option is set
        read = getattr(view, 'global_read_permission', False)
        if read and request.method in permissions.SAFE_METHODS:
            return True
```

CLEAN...ISH PERMISSIONS

```
class AccountPermission(permissions.BasePermission):
    def has_permission(self, request, view):
        # Allow all access to admin users
        if request.auth and request.auth.is_admin:
            return True

        # Allow global read/write permission where option is set
        if getattr(view, 'global_full_permission', False):
            return True

        # Allow global read permission where option is set
        read = getattr(view, 'global_read_permission', False)
        if read and request.method in permissions.SAFE_METHODS:
            return True
```

chewse

CLEAN...ISH API

- Oh you wanted auth?
- Oh you wanted permissions?
- Oh you wanted filtering?

CLEAN...ISH FILTERING

```
class HumanFilterSet(django_filters.FilterSet):
    created_lte = IsoDateTimeFilter(
        name='created',
        lookup_type='lte',
        input_formats=IsoDateTimeFilter.ISO_INPUT_FORMATS
    )

    class Meta:
        model = Human
        fields = ['created_lte',]
```

chewse

CLEAN...ISH FILTERING

```
class HumanFilterSet(django_filters.FilterSet):
    created_lte = IsoDateTimeFilter(
        name='created',
        lookup_type='lte',
        input_formats=IsoDateTimeFilter.ISO_INPUT_FORMATS
    )

    class Meta:
        model = Human
        fields = ['created_lte',]
```

chewse

CLEAN...ISH FILTERING

```
class HumanFilterSet(django_filters.FilterSet):
    created_lte = IsoDateTimeFilter(
        name='created',
        lookup_type='lte',
        input_formats=IsoDateTimeFilter.ISO_INPUT_FORMATS
    )

    class Meta:
        model = Human
        fields = ['created_lte',]
```

chewse

CLEAN...ISH FILTERING

```
class HumanFilterSet(django_filters.FilterSet):
    created_lte = IsoDateTimeFilter(
        name='created',
        lookup_type='gte',
        input_formats=IsoDateTimeFilter.ISO_INPUT_FORMATS
    )

    class Meta:
        model = Human
        fields = ['created_lte',]
```

3. CURVEBALLS

MMVV**?

- Frontend MV* and backend MV*

MMVV**?

- Frontend MV* and backend MV*
- Backend models mirrored in frontend

MMVV**?

- Frontend MV* and backend MV*
- Backend models mirrored in frontend
- Controller and view code in frontend

chewse

MMVV**?

- Frontend MV* and backend MV*
- Backend models mirrored in frontend
- Controller and view code in frontend
- Backend background processing

INTERPOLATION FIGHT!

- “Wait — you were gonna use {{ var }}?”

INTERPOLATION FIGHT!

- “Wait — you were gonna use {{ var }}?”
- “I thought I was gonna use {{ var }}”

INTERPOLATION FIGHT!

- “Wait — you were gonna use {{ var }}?”
- “I thought I was gonna use {{ var }}”
- Said Angular, Django and Jinja all at once

INTERPOLATION FIGHT!

```
module.config(function ($interpolateProvider) {
  $interpolateProvider
    .startSymbol('[[')
    .endSymbol(']]');
});
```

INTERPOLATION FIGHT!

```
module.config(function ($interpolateProvider) {
  $interpolateProvider
    .startSymbol('[[')
    .endSymbol(']]');
});
```

OTHER GOTCHAS

- CSRF wiring

chewse

OTHER GOTCHAS

- CSRF wiring
- API pagination

chewse

4. ENJOY!

Jeff Schenck CTO & Co-Founder

- stinky cheese
- jeffschenck
- www.chewse.com