
django
MEETS
BLOCKCHAIN
WORKSHOP


rev1 ventures


talio

SafeChain

Format Tonight:

Goals for this Workshop

Gentle introduction to Python and Django REST Framework

Lab Exercise Part 1

How SafeChain uses Blockchain

Gentle introduction to Ethereum blockchain and Web3.py

Lab Exercise Part 2

Finish the prework ?

[Now’s a good time to run through those steps in the background if not]

Goals for Tonight

1. Build a Simple API
2. Talk to Ethereum mainnet
3. Collect your Columbus token
4. ???
5. Profit

Let’s talk Python

#1 -- Syntactic White Space

Where you're used to seeing:

```
function foo(a) {
  console.log("Hello");
}

if (a == b) {
  console.log("True");
}
```

This is what to expect in Python:

```
def foo(a):
    print("Hello")


if a == b:
    print("True")
```

No curly brackets to set scope nor semicolons to end lines

But… colons and indentation matter a lot

#2 -- Strongly-, Implicitly- and Dynamically-typed

No declaration:

Java:

```
String foo = "Bar";
```

Python:

```
foo = "Bar"
```

But types still matter:

```
b = 2 + "2"
```

^^ will throw a TypeError!

Variables can be reassigned to new types:

```
a = "the answer to life the
universe and everything"
a = 42
```

#3 -- Always by reference, all the time

No pointers!

But be careful…

Passing by reference can also make it easy to unintentionally modify an object passed from elsewhere.

#4 -- It’s [usually] interpreted

Pros:

- No separate compilation step
- Easy to interactively debug
- Fairly portable

Cons:

- Can be slower than true compiled languages

On to Django [REST Framework]

Django, out of the box

URLs/Routes

Models

Allow you to work with your database in an object-oriented fashion without touching SQL.

Views

Define how a given route behaves.

Templates

Define how a given route appears to the user.

- Closely resembles the MVC pattern that become popular in the mid/late 2000s.
- Great for quickly building dynamic, server-side rendered applications.
- Historically, popular for building news and other content sites.
- Can easily use just about any Python package within your Django app
- Good bones but not quite ideal for building modern client-rendered apps or RESTful APIs

Django REST Framework

Replaces Views with ViewSets

Standardized behavior to match RESTful HTTP verbs -- GET, PUT, POST, DELETE, etc.

Easily extend behavior with nested list and detail routes.

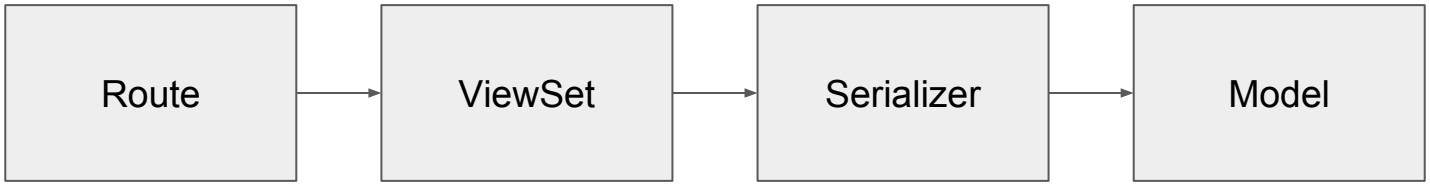
Replaces Templates with Serializers

Automatically accepts or generates standards-compliant JSON/XML/other output based on your model fields.

Ability to easily include/exclude fields.

In short, makes it just as easy to build APIs as Django makes it to build server-side rendered applications.

What’s in a route?



What's in a route?

Route

```
router.register(r'wallets', core_views.WalletViewSet)
```

Serializer

```
class WalletSerializer(serializers.ModelSerializer):

    class Meta:

        model = Wallet

        fields = ('id', 'label', 'address')
```

ViewSet

```
class WalletViewSet(viewsets.ModelViewSet):

    queryset = Wallet.objects.all().order_by('id')

    serializer_class = WalletSerializer
```

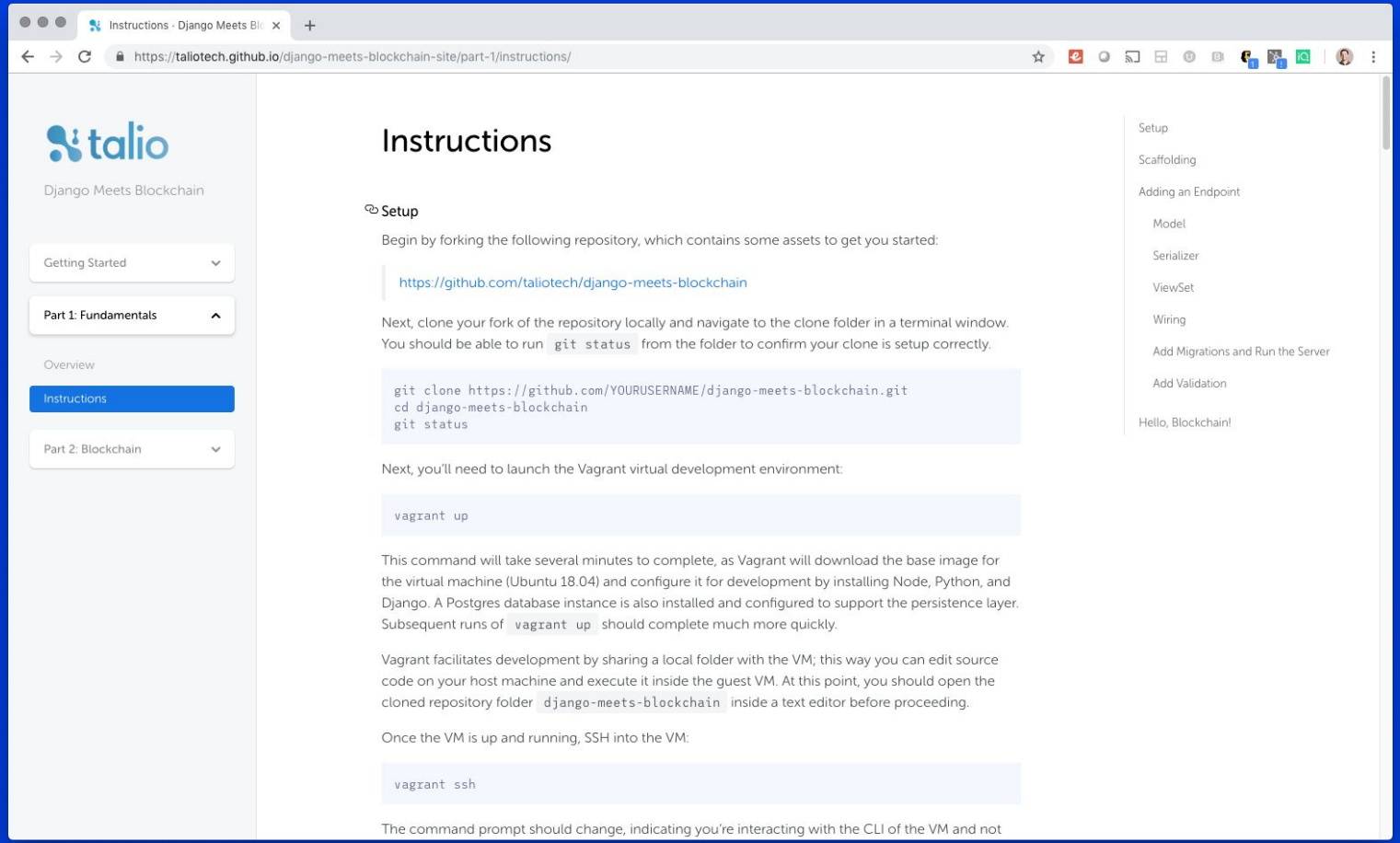
Model

```
class Wallet(models.Model):

    label = models.CharField(max_length=255)

    address = models.CharField(max_length=255)
```

Questions?


Instructions · Django Meets Bl
https://taliotech.github.io/django-meets-blockchain-site/part-1/instructions/
talio
Django Meets Blockchain
Getting Started
Part 1: Fundamentals
Overview
Instructions
Part 2: Blockchain
Instructions
Setup
Begin by forking the following repository, which contains some assets to get you started:
https://github.com/taliotech/django-meets-blockchain
Next, clone your fork of the repository locally and navigate to the clone folder in a terminal window. You should be able to run git status from the folder to confirm your clone is setup correctly.
git clone https://github.com/YOURUSERNAME/django-meets-blockchain.git
cd django-meets-blockchain
git status
Next, you'll need to launch the Vagrant virtual development environment:
vagrant up
This command will take several minutes to complete, as Vagrant will download the base image for the virtual machine (Ubuntu 18.04) and configure it for development by installing Node, Python, and Django. A Postgres database instance is also installed and configured to support the persistence layer. Subsequent runs of vagrant up should complete much more quickly.
Vagrant facilitates development by sharing a local folder with the VM; this way you can edit source code on your host machine and execute it inside the guest VM. At this point, you should open the cloned repository folder django-meets-blockchain inside a text editor before proceeding.
Once the VM is up and running, SSH into the VM:
vagrant ssh
The command prompt should change, indicating you're interacting with the CLI of the VM and not
Setup
Scaffolding
Adding an Endpoint
Model
Serializer
ViewSet
Wiring
Add Migrations and Run the Server
Add Validation
Hello, Blockchain!