THE DEVELOPER'S CONFERENCE

Pare de se perder! Crie rapidamente sua blockchain com o Convector Suite

Jefferson Bicca Charczuk
Software developer

Dúvidas para quem começa

- Como monto uma rede blockchain para desenvolvimento?
- Como estruturo o chaincode?
- Como testo o chaincode?

Precisa entender o básico

- Organization
- Channel
- Peers, CAs, Orderers
- Policies
- Collections config
- Install chaincode
- Instantiate chaincode

Não é sugerida uma estrutura básica

- Mistura entre modelo e controle
- Funciona bem para chaincodes pequenos e com complexidade "gerenciável":
  - Mas e se o chaincode crescer?
  - Mas e se o negócio for complexo?

Precisa subir uma rede

- Ciclo de instalação e instanciação de chaincode na rede
  - Isso leva tempo, consequentemente
- Precisa ser feito um *invoke* na rede. Se quiser automatizar:
  - Criar um script com os *invokes*
  - Criar API e automatizar as requisições por ela
- Testes são no nível de serviço

Convector Suite

THE DEVELOPER'S CONFERENCE

Convector

JavaScript fullstack smart contract systems

The way to develop world-class smart contract systems for Hyperledger Fabric.

Hurley

Streamline your development process

Quickly setup your Hyperledger Fabric development environment.

Facilita desenvolvimento chaincode

- Facilita na criação, desenvolvimento e testes
  - CLI e Runners
  - Testes de unidade com o Mocha
  - Validação de schema
- Sugere padronização de estruturação
  - TypeScript
  - Model/Controller

CLI

THE DEVELOPER'S CONFERENCE

```
npm i -g @worldsibu/convector-cli

# Create a new project with a default chaincode package
conv new tdcblockchain -c conference

cd tdcblockchain
npm i

# Bring up a development blockchain in your computer
npm run env:restart

# Install the chaincode to the blockchain
npm run cc:start -- conference

# Apply all the changes you want to the project.
# When you need to upgrade your chaincode in the blockchain, you can simply do
npm run cc:upgrade -- conference 1.2
```

Testes de unidade com o Mocha

THE DEVELOPER'S CONFERENCE

```
it('should create Participant in private collection speakers', async() => {

   const participant = new Participant({
      id: uuid(),
      name: 'ALICE',
      tracks: [{ "name": "blockchain", "status": "A" }]
   });

   const txId = await
participantCtrl.$withUser('Test').registerAsSpeaker(participant);
   expect(txId).to.exist;

   const participantBlockchain = await
participantCtrl.$withUser('Test').findSpeaker(participant.id);
   expect(new Participant(participantBlockchain).id).to.exist;


});
```

Validação de schema

THE DEVELOPER'S CONFERENCE

```
@Required()
@Validate(yup.string())
public name: string;

@Required()
@Validate(yup.array(Track.schema()))
public tracks: Array<FlatConvectorModel<Track>>;
```

Models

THE DEVELOPER'S CONFERENCE

```
export class Participant extends ConvectorModel<Participant> {

    @ReadOnly()
    @Required()
    public readonly type = 'conferences.Participant';
    // métodos herdados da superclasse
    static schema<T extends ConvectorModel<any>>
    static getOne<T extends ConvectorModel<any>>
    static query<T>
    static getAll<T extends ConvectorModel<any>>
    update(content: any)
    fetch(storageOptions?: any)
    history()
    save(storageOptions?: any)
    clone()
    toJSON(skipEmpty?: boolean)
    delete(storageOptions?: any)
}
```

Controllers

THE DEVELOPER'S CONFERENCE

```
@Controller('participant')
export class ParticipantController extends ConvectorController<ChaincodeTx> {

 @Invokable()
 public async registerAsSpeaker(@Param(Participant) participant: Participant) {
   await participant.save({ privateCollection : collection });

   return { txID: await this.getTxId() };
 }

}
```

Controllers - Native API

THE DEVELOPER'S CONFERENCE

```
@Controller('participant')
export class ParticipantController extends ConvectorController<ChaincodeTx> {

  @Invokable()
  public async findSpeakers() {
      let queryString = { selector: { '_id': { '$gt': null } } };
      let speakers = await
this.tx.stub.getStub().getPrivateDataQueryResult('collectionName',
JSON.stringify(queryString))
      speakers = (<any> speakers).iterator ? (<any> speakers).iterator : speakers; //
hack for fabric 1.4

      return await Transform.iteratorToList(speakers);
  }

}
```

Criar ambiente para desenvolvimento

- Gera uma rede com a quantidade de organizações, canais, usuários e políticas de acordo com parâmetros passados. Isto inclui:
  - Material criptográfico
  - Criação e *enroll* de usuários admin e user1..*
  - Políticas de acordo com arquivos org*.config.json
  - Arquivo docker-compose.yml

Hurley

THE DEVELOPER'S CONFERENCE

```
npm i -g @worldsibu/hurley

# Inicializa uma rede com 3 organizações (npm run env:restart)
hurl new --organizations 3

# Instala e instancia o chaincode (npm run cc:start -- conferences)
hurl install conferences node -P ./chaincode-conferences --collections-config
./collections_config.json;

# Invocando transação na blockchain
hurl invoke conferences participant_registerAsSpeaker '{"id":1,
"name":"BOB","tracks":[{"id":1,"status":"A","name":"blockchain"]}' "speakers" -u
user1 -o org1
```

Gera a API a partir do chaincode

Convector Rest Server

THE DEVELOPER'S CONFERENCE

```
npm install -g @worldsibu/conv-rest-api
npm install -g lerna
conv-rest-api generate api -c conference -f conference.config.json
lerna bootstrap
export CHANNEL=chconf && export ORG=blockchain && export
COUCHDBVIEW=chconf_conference
lerna run start --scope server --stream
```

Hands On

Mais informações

THE DEVELOPER'S CONFERENCE

https://github.com/jeffbicca/tdcblockchain

https://github.com/hyperledger-labs/convector

Obrigado!

Jefferson Bicca Charczuk
Software developer

jefferson.rs@gmail.com
https://www.medium.com/@jeffbicca
@jeffbicca

THE DEVELOPER'S CONFERENCE

THE DEVELOPER'S CONFERENCE

Jefferson Bicca Charczuk

Desenvolvedor desde 2004
Ciência da Computação PUCRS, 2007
Trabalhando no Serpro desde 2010

#java #javascript #typescript #microservices #tdd #ddd #blockchain