**詳細設計書**

**ブロックチェーンを用いた**

**酒類在庫共有システム**

**1198 桑田惇平**

**0.1版**

|  |  |  |
| --- | --- | --- |
| 版数 | 年月日 | 内容 |
| 0.1 | 2020/12/01 | 初版 |

# クラス一覧

|  |  |  |
| --- | --- | --- |
| 項番 | ファイル名 | 説明 |
| 1 | Liquor.sol | 酒類情報をもったトークンを表すコントラクト |
| 2 | liquorDao.py | データベース、およびブロックチェーンと酒類情報の  やり取りを行うためのクラス |
| 3 | app.py | 画面の表示、および入力情報の受け取りを  行うためのクラス |
| 4 | default\_controller.py | REST APIの処理を規定するクラス |

# クラス詳細

## Liquor.sol

### クラス概要

・酒類情報をもったトークンを表すコントラクト。

　ERC721インターフェースを継承する

【フィールド変数】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 項番 | 属性 | 変数名 | I/O | 内容 |
| 1 | struct | Liquor | I | 商品情報(注) |
| 2 | Liquor[] private | liquorCollection | I | 各商品を格納する配列 |
| 3 | address private | \_contractCreator | I | コントラクト作成者のアドレス |

（注）Liquor内には、以下の変数を宣言する。

1. uint tokenId(トークンID)

2. string memory liquorName (商品名)

3. struct LiquorInfo (商品情報)

3.1 string sellerName (取扱人名)

3.2 bool isReservable(予約可能かどうか)

3.3 string arrivalDay (入荷日)

3.4 int reserveScore (取り置き実績)

### メソッド定義

※”openzeppelin-solidity/contracts/token/ERC721/ERC721Full.sol” をインポートする

#### constructor() ERC721Full (public)

・継承元のERC721Fullのコンストラクタ。

【引数】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 項番 | 属性 | 引数名 | I/O | 内容 |
| 1 | string | “Liquor” | I | トークンの名前 |
| 2 | symbol | “LIQ” | I | トークンの単位 |

【返り値】

返り値なし。

【処理内容】

(1) \_contractCreator = msg.senderを実行する

#### B. fetchLiquorData (public view)

・各トークンに格納された酒類情報を参照する

【引数】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 項番 | 属性 | 引数名 | I/O | 内容 |
| 1 | uint256 | \_tokenId | I | トークンID |

【返り値】

|  |  |  |  |
| --- | --- | --- | --- |
| 項番 | 属性 | I/O | 内容 |
| 1 | Liquor | O | 酒類情報 |

【処理内容】

(1) return liquorCollection[\_tokenId]を実行する

#### C. getAllData (public view)

・ブロックチェーン上のすべてのデータを読み出す

【引数】

引数なし。

【返り値】

|  |  |  |  |
| --- | --- | --- | --- |
| 項番 | 属性 | I/O | 内容 |
| 1 | Liquor[] | O | BCから取得したデータ |

【処理内容】

(1) return liqourCollection を実行する

#### D. changeBlock (external)

・在庫のなくなった商品を参照しないようブロックを追加する

【引数】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 項番 | 属性 | 引数名 | I/O | 内容 |
| 1 | uint256 | \_tokenId | I | トークンID |

【返り値】

|  |  |  |  |
| --- | --- | --- | --- |
| 項番 | 属性 | I/O | 内容 |
| 1 | int | O | 新しいトークンID |

【処理内容】

(1) 変数newTokenId を宣言し、 liquorCollection.length + 1を代入する

(2) super.\_mint(\_contractCreator, newTokenId)を実行する

(3) liquorCollection.push(Liquor(newTokenId,

liquorCollection[\_tokenId].liquorName,

LiquorInfo(

liquorCollection[\_tokenId].liquorInfo.sellerName,

false,

liquorCollection[\_tokenId].liquorInfo.arrivalDay,

liquorCollection[\_tokenId].liquorInfo.reserveScore))))

　　を実行する

(4) return newTokenId を実行する

## liquorDao.py

### クラス概要 ・データベース、およびブロックチェーンと酒類情報のやり取りを

行う機能を担うクラス。flackフレームワーク内に作成する

【フィールド変数】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 項番 | 属性 | 変数名 | I/O | 内容 |
| 1 | string | transactionId | I | トランザクションID |

### メソッド定義

※１ 冒頭で ”from web3 import web3”,

および “import sqlite3”を記述する。

※２ 本プログラム実装前にLiquor.solをブロックチェーン上にデプロイし、

　　JSON形式のABIとコントラクトアドレスとを取得しておく

#### def fetchDataFromBC

・ブロックチェーンから情報を取得する

【引数】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 項番 | 属性 | 引数名 | I/O | 内容 |
| 1 | int | tokenId | I | トークンID |

【返り値】

|  |  |  |  |
| --- | --- | --- | --- |
| 項番 | 属性 | I/O | 内容 |
| 1 | string | O | BCから取得したデータ |

【処理内容】

1. 変数web3pyを宣言し、new Web3(web3.currentProvider)を代入する
2. 変数abiを宣言し、※２で得たABIを代入する
3. 変数contractAddressを宣言し、※２で得たコントラクトアドレスを代入する
4. 変数liquorsを宣言し、new web3py.eth.Contract(abi, contractAddress)を

代入する

1. return liquors.methods.fetchLiquorData(tokenId).call() を実行する

#### def fetchDataFromDB

・データベースから情報を取得する

【引数】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 項番 | 属性 | 引数名 | I/O | 内容 |
| 1 | string | liquorName | I | 商品名 |

【返り値】

|  |  |  |  |
| --- | --- | --- | --- |
| 項番 | 属性 | I/O | 内容 |
| 1 | int | O | トークンID |

【処理内容】

1. session.query(liquor\_table.TRANSCTION\_ID).filter(liquor\_table.LIQUOR\_

NAME == liquorName)を実行し、その返り値をreturnする

#### C. def getAllDataFromBC

・ブロックチェーンからすべてのトークンを取得する

【引数】

引数なし。

【返り値】

|  |  |  |  |
| --- | --- | --- | --- |
| 項番 | 属性 | I/O | 内容 |
| 1 | string[] | O | 全商品データ |

【処理内容】

1. 変数web3pyを宣言し、new Web3(web3.currentProvider)を代入する
2. 変数abiを宣言し、※２で得たABIを代入する
3. 変数contractAddressを宣言し、※２で得たコントラクトアドレスを代入する
4. 変数liquorsを宣言し、new web3py.eth.Contract(abi, contractAddress)を

代入する

1. return liquors.methods.getAllData().call() を実行する

#### D. def updateStockOnDB

・データベース上の商品在庫数を更新する

【引数】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 項番 | 属性 | 引数名 | I/O | 内容 |
| 1 | string | liquorName | I | 商品名 |
| 2 | string | sellerName | I | 取扱人名 |

【返り値】

|  |  |  |  |
| --- | --- | --- | --- |
| 項番 | 属性 | I/O | 内容 |
| 1 | bool | O | 更新が成功したかどうか |

【処理内容】

(1) 変数stockQueryを宣言し、session.query(liquor\_table)を代入する

(2) 変数stock\_updateを宣言し、stockQuery.filter(liquor\_table.LIQUOR\_NAME

== liquorName, liquor\_table.SELLER\_NAME == sellerName) を代入する

(3) stock\_update. STOCK\_QUANTITY -= 1 を実行する

(4) 条件分岐：

A. stock\_update. STOCK\_QUANTITY < 0のとき

1. return false を実行する

B. stock\_update. STOCK\_QUANTITY == 0 のとき

1. updateTokenId(stock\_update.TOKEN\_ID, stock\_update)を実行する

2. return true を実行する

C. A, B 以外のとき

1. return true を実行する

(5)return false を実行する

#### E. def changeBlock

・在庫のなくなった商品を参照しないようブロックを追加する

【引数】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 項番 | 属性 | 引数名 | I/O | 内容 |
| 1 | int | tokenId | I | トークンID |

【返り値】

|  |  |  |  |
| --- | --- | --- | --- |
| 項番 | 属性 | I/O | 内容 |
| 1 | int | O | 新しいトークンID |

【処理内容】

1. 変数web3pyを宣言し、new Web3(web3.currentProvider)を代入する
2. 変数abiを宣言し、※２で得たABIを代入する
3. 変数contractAddressを宣言し、※２で得たコントラクトアドレスを代入する
4. 変数liquorsを宣言し、new web3py.eth.Contract(abi, contractAddress)を

代入する

1. return liquors.methods.changeBlock(tokenId).call() を実行する

#### F. def updateTokenId

・データベースのトークンIDを更新する

【引数】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 項番 | 属性 | 引数名 | I/O | 内容 |
| 1 | int | tokenId | I | トークンID |
| 2 | session | updateRow | I | IDを更新する行 |

【返り値】

返り値なし。

【処理内容】

(1) 変数newTokenIdを宣言し、changeBlock(tokenId)を代入する

(2) updateRow.TOKEN\_ID にnewTokenIdを代入する

## app.py

### クラス概要

・jinjaを利用した画面の表示、および入力情報の受け取り機能を担うクラス。

　flaskフレームワーク内に作成する

### メソッド定義

※１ 冒頭で“from flask\_sqlalchemy import SQLAlchemy”,

“from sqlalchemy import Integer, String” を記述する。

※２ メソッド外に以下の処理を記述する。

1. 変数appを宣言し、Flask(\_\_name\_\_)を代入する
2. app.config[‘SECRET\_KEY’] に ‘secret key’ を代入する
3. app.config[‘SQLALCHEMY\_DATABASE\_URI’] に

‘sqlite:///flask.sqlite’ を代入する

1. 変数databaseを宣言し、SQLAlchemy(app)を代入する
2. 変数\_\_tablename\_\_を宣言し、’liquor\_table’を代入する
3. 変数LIQUOR\_NAMEを宣言し、database.Column(String(32), primary\_key=True)を代入する
4. 変数SELLER\_NAMEを宣言し、database.Column(String(32), primary\_key=True)を代入する
5. 変数STOCK\_QUANTITYを宣言し、database.Column(Integer)を代入する
6. 変数TOKEN\_IDを宣言し、database.Column(Integer)を代入する
7. database.create\_all()を実行する

#### def index

・アプリケーションルート(index.html)にアクセスがあった時の処理を

規定する。

@app.route(“/”, methods=[‘get’])をメソッドに付与する

【引数】

引数なし。

【返り値】

|  |  |  |  |
| --- | --- | --- | --- |
| 項番 | 属性 | I/O | 内容 |
| 1 | render\_template | O | トップ画面 |

【処理内容】

1. return render\_template(‘index.html’)を実行する

## default\_controller.py

### クラス概要

・REST APIが返す内容を規定するクラス

### 2.4.2メソッド定義

※1 冒頭で “import ../liquorDao”を記述する

※2 同じく冒頭で “import json”を記述する

#### A. def records\_get()

・REST APIにGETリクエストがあった時の処理を規定する

【引数】

引数なし。

【返り値】

|  |  |  |  |
| --- | --- | --- | --- |
| 項番 | 属性 | I/O | 内容 |
| 1 | string | O | BCから取得したデータ |

【処理内容】

(1) 変数allDataを宣言し、liquorDao.getAllDataFromBC() を代入する

(2) return json.dumps(allData) を実行する

#### B. def records\_post()

・REST APIにPOSTリクエストがあった時の処理を規定する

【引数】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 項番 | 属性 | 引数名 | I/O | 内容 |
| 1 | string | liquorName | I | 商品名 |
| 2 | string | sellerName | I | 取扱人名 |

【返り値】

|  |  |  |  |
| --- | --- | --- | --- |
| 項番 | 属性 | I/O | 内容 |
| 1 | bool | O | 更新が成功したかどうか |

【処理内容】

(1)liquorDao.updateStockOnDB(liquorName, sellerName)を実行する

(2) return Trueを実行する

#### C. def records\_put()

・REST APIにPUTリクエストがあった時の処理を規定する

【引数】

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 項番 | 属性 | 引数名 | I/O | 内容 |
| 1 | string | searchWord | I | 検索ワード |

【返り値】

|  |  |  |  |
| --- | --- | --- | --- |
| 項番 | 属性 | I/O | 内容 |
| 1 | string | O | BCから取得したデータ |

【処理内容】

(1) 変数tokenId を宣言し、liquorDao.fetchDataFromDB(searchWord)を

代入する

(2) 変数liquorDataを宣言し、liquorDao.fetchDataFromBC(tokenId) を代入する

(3) return json.dumps(liquorData) を実行する