Utilize REST API Server

In the previous tutorial, the chaincode functions are called through a NodeJS Client SDK that are utilized by NodeJS programs.

In order to easily connect the chaincode to other systems, a REST API can be utilized to call the different chaincode functions.

Quick Setup

1. Run the quick setup.

```
chaincode> ./quick-setup.sh blue-coin blue-coin 1.0
```

Register and Enroll the Users of Org1 and Org2

1. Enroll the bootstrap adminstrators of org1 and org2.

```
bc-client> node enrollAdminWithUserManager.js 1

bc-client> node enrollAdminWithUserManager.js 2
```

2. Regiser and enroll `user1` and `user2` of org1 and org2.

```
bc-client> node enrollUserWithUserManager.js 1 1

bc-client> node enrollUserWithUserManager.js 1 2

bc-client> node enrollUserWithUserManager.js 2 1

bc-client> node enrollUserWithUserManager.js 2 2
```

Examine the REST API Server Folder

1. List the files of the `rest-api-server/blue-coin` subfolder.

```
bc-rest> ls
```

The folder contains several files and subfolders. Some of these files and subfolders are listed below:

- `app.js`
- `route`
  - `BlueCoinRoute.js`
- `controller`
  - `BlueCoinController.js`

The file `app.js` is the main program that will start a REST API srever.

The program uses the following syntax.

Syntax:

```
bc-rest> node app.js <org index>
```

Example:

```
bc-rest> node app.js 1
```

The sample command above starts the REST API server for org1.

The program `BlueCoinRoute.js` defines the different endpoints for the blue coin REST API.

The details of the endpoints are shown below.

Endpoint	Method	Body	Example
`/generateInitialCoin/:userId/:mspId`	POST	none	`/generateInitialCoin/user1/Org1MSP`
`/getBalance/:userId/:mspId`	GET	none	`/getBalance/user1/Org1MSP`
`/transferCoin/:userId`	POST	`{ "srcMspId": <srcMspId>, "dstMspId": <dstMspId>, "amt": <amt> }`	`/transferCoin/user1 { "srcMspId": "Org1MSP", "dstMspId": "Org2MSP", "amt": 150 }`
`/getTransactionHistory/:userId/:mspId`	GET	none	`/getTransactionHistory/user1/Org1MSP`

The endpoints enumerated above are incorporated in `app.js` using the path `/blue-coin`. This means that all the endpoints starts with the said path (e.g., `/blue-coin/generateInitialCoin/:userId/:mspId`)

2. Start the REST API server of org1.

```
bc-rest> node app.js 1
```

A message states that the server is running on port `8081`.

This means that endpoints can be accessed locally through `http://localhost:8081`.

For example, to generate initial coins for org1 you can use the following REST API call: `http://localhost:8081/blue-coin/generateInitialCoin/user1/Org1MSP`

In this sample REST API call, `user1` of org1 is used to call the `generateInitialCoin` through the REST API call.

3. Open a second blue coin REST API terminal.

4. Start the REST API server of org2.

```
bc-rest #2> node app.js 2
```

A message states that the server is running on port `8082`.

This means that endpoints can be accessed locally through `http://localhost:8082`.

Make REST API Calls using Postman

1. Open Postman.
2. Choose the method POST.
3. Type the URL `http://localhost:8081/blue-coin/generateInitialCoin/user1/Org1MSP`.
4. Click `Send`.

   Verify that 500 blue coins are generated for org1.
5. Repeat the steps above but use the URL `http://localhost:8082/blue-coin/generateInitialCoin/user1/Org2MSP`.

   Verify that 500 blue coins are generated for org2.
6. Using postman, transfer 50 coins from org1 to org2.
7. Using postman, get the balance of org1.
8. Using postman, get the balance of org2.
9. Using postman, get the transaction history of the blue coins of org1.
10. Using postman, get the transaction history of the blue coins of org2.

Update the REST API Server.

1. Update the REST API server to include an endpoint for the chaincode function `getAboveAll`.