Happy friend v1 Specification

Author(s): Brian Yang

1. Hfv1 Summary

This is really a great examination which is very comprehensive, thank you so much!

According to the requirement, the request and response accept JSON format data, so I consider using restful web service. For lightweight and quick building purpose, I choose to use Jersey+ jetty. But it doesn’t mean I would recommend this in actual project. It based on requirement.

1. Interface
   1. Create : create a friend connection between two email addresses

* Detail

|  |  |
| --- | --- |
| Request method type | POST |
| URL | /api/user/v1/create |
| Sample URL | <http://192.168.1.108:8888/api/user/v1/create> |
| Request Type | application/json |
| Request Description | {'friends': ['xx@xx.com', 'xx@xx.com'] }  friends: email addresses |
| Response Type | application/json |
| Response Description | Success: {"success": true }  Failure:  {"friends":["xx@xx.com","xx@xx.com"],  "ErrMessage":"XXXX", "ErrorCode":"XXXX" }  ErrMessage: error description  ErrorCode: customized business error code |

* Error info which may be returned

|  |  |
| --- | --- |
| ErrorCode | ErrMessage |
| 1001 | Cannot find the user |
| 1002 | One user is in the block list of the other |
| 5001 | Bad request |
| 5002 | There is an error in the server |

* Sample

|  |  |
| --- | --- |
| Request | Response |
| {  friends:  [  'andy@example.com',  'john@example.com'  ]  } | {  "success": true  } |
| {  friends:  [  'andy@example.com',  'john@example.com'  ]  } | {  friends:[  "andy@example.com"  "john@example.com"]  ErrMessage: "One user is in the block list of the other"  ErrorCode: "1002"  } |

* 1. Get Friends list : retrieve the friends list for an email address
* Detail

|  |  |
| --- | --- |
| Request method type | POST |
| URL | /api/user/v1/getFriends |
| Sample URL | <http://192.168.1.108:8888/api/user/v1/getFriends> |
| Request Type | application/json |
| Request Description | {'email': 'xx@xx.com' }  email: email address |
| Response Type | application/json |
| Response Description | Success:  {"success": true, "friends" : [xx@xx.com'], "count" : 1}  Failure:  {"email":"xx@xx.com",  "ErrMessage":"XXXX", "ErrorCode":"XXXX" }  ErrMessage: error description  ErrorCode: customized business error code |

* Error info which may be returned

|  |  |
| --- | --- |
| ErrorCode | ErrMessage |
| 1001 | Cannot find the user |
| 5001 | Bad request |
| 5002 | There is an error in the server |

* Sample

|  |  |
| --- | --- |
| Request | Response |
| {  email:'andy@example.com'  } | {  success: true  friends: [  "john@example.com"  "tony@example.com"  ]  count: 2  } |
| {  email:'andy111@example.com'  } | {  email: "andy111@example.com"  ErrMessage: "Cannot find the user"  ErrorCode: "1001"  } |

* 1. Get Common Friends list :

Retrieve the common friends list between two email addresses

* Detail

|  |  |
| --- | --- |
| Request method type | POST |
| URL | /api/user/v1/getCommonFriends |
| Sample URL | <http://192.168.1.108:8888/api/user/v1/getCommonFriends> |
| Request Type | application/json |
| Request Description | {'friends': ['xx@xx.com', 'xx@xx.com'] }  friends: email addresses |
| Response Type | application/json |
| Response Description | Success:  {"success": true, "friends" : [xx@xx.com'], "count" : 1}  Failure:  {"friends":"xx@xx.com"],  "ErrMessage":"XXXX", "ErrorCode":"XXXX" }  ErrMessage: Error description |

* Error info which may be returned

|  |  |
| --- | --- |
| ErrorCode | ErrMessage |
| 1001 | Cannot find the user |
| 5001 | Bad request |
| 5002 | There is an error in the server |

* Sample

|  |  |
| --- | --- |
| Request | Response |
| {  friends:  ['andy@example.com',  'john@example.com'  ]  } | {  success: true  friends: [  "tony@example.com" ]  count: 1  } |
| {  friends:  ['andy12@example.com',  'john@example.com'  ]  } | {  friends: [  "andy12@example.com"  "john@example.com"]  ErrMessage: "Cannot find the user"  ErrorCode: "1001"  } |

* 1. Subscribe: provide a way to subscribe to updates from an email address
* Detail

|  |  |
| --- | --- |
| Request method type | POST |
| URL | /api/user/v1/subscribe |
| Sample URL | <http://192.168.1.108:8888/api/user/v1/subscribe> |
| Request Type | application/json |
| Request Description | {'requestor':'xx@xx.com',  'target': 'xx@xx.com' }  requestor: apply for subscribing  target: the subject which will notify if has update |
| Response Type | application/json |
| Response Description | Success: {"success": true }  Failure:  { requestor: "xx@xx.com"  target: "xx@xx.com"  "ErrMessage":"XXXX", "ErrorCode":"XXXX" }  ErrMessage: Error description  ErrorCode: customized business error code |

* Error info which may be returned

|  |  |
| --- | --- |
| ErrorCode | ErrMessage |
| 1001 | Cannot find the user |
| 5001 | Bad request |
| 5002 | There is an error in the server |

* Sample

|  |  |
| --- | --- |
| Request | Response |
| {  'requestor': 'lisa@example.com',  'target': 'john@example.com'  } | {  success: true  } |
| {  'requestor': 'lisa12@example.com',  'target': 'john@example.com'  } | {  requestor: "lisa12@example.com"  target: "john@example.com"  ErrMessage: "Cannot find the user"  ErrorCode: "1001"  } |

* 1. Block: provide a way to block updates from an email address
* Detail

|  |  |
| --- | --- |
| Request method type | POST |
| URL | /api/user/v1/block |
| Sample URL | http://192.168.1.108:8888/api/user/v1/block |
| Request Type | application/json |
| Request Description | {'requestor':'xx@xx.com',  'target': 'xx@xx.com' }  requestor: apply for blocking  target: be blocked by requestor |
| Response Type | application/json |
| Response Description | Success: {"success": true }  Failure:  { requestor: "xx@xx.com"  target: "xx@xx.com"  "ErrMessage":"XXXX", "ErrorCode":"XXXX" }  ErrMessage: error description  ErrorCode: customized business error code |

* Error info which may be returned

|  |  |
| --- | --- |
| ErrorCode | ErrMessage |
| 1001 | Cannot find the user |
| 5001 | Bad request |
| 5002 | There is an error in the server |

* Sample

|  |  |
| --- | --- |
| Request | Response |
| {  'requestor': 'andy@example.com',  'target': 'john@example.com'  } | {  success: true  } |
| {  'requestor': 'andy12@example.com',  'target': 'john@example.com'  } | {  requestor: "andy12@example.com"  target: "john@example.com"  ErrMessage: "Cannot find the user"  ErrorCode: "1001"  } |

* 1. Get Recipients: provide a way to get all email addresses can be receive update from an email address
* Detail

|  |  |
| --- | --- |
| Request method type | POST |
| URL | /api/user/v1/getRecipients |
| Sample URL | http://192.168.1.108:8888/api/user/v1/getRecipients |
| Request Type | application/json |
| Request Description | {'sender':'xx@xx.com',  'text': 'xxxx xx' }  sender: the subject which will notify if has update  text: the message which will send to all the recipients |
| Response Type | application/json |
| Response Description | Success:  {success: true, recipients: ["xx@xx.com", "xx@x.com"] }  Failure:  { 'sender':'xx@xx.com', 'text': 'xxxx xx',  "ErrMessage":"XXXX", "ErrorCode":"XXXX" }  ErrMessage: error description  ErrorCode: customized business error code |

* Error info which may be returned

|  |  |
| --- | --- |
| ErrorCode | ErrMessage |
| 1001 | Cannot find the user |
| 5001 | Bad request |
| 5002 | There is an error in the server |

* Sample

|  |  |
| --- | --- |
| Request | Response |
| {sender:  'andy@example.com',  'text':  'Hello World! [kate@example.com](mailto:kate@example.com)'  } | {  success: true  recipients: [  "john@example.com",  "kate@example.com",  "tony@example.com"]  } |
| {sender:  'andy12@example.com',  'text':  'Hello World! [kate@example.com](mailto:kate@example.com)'  } | {  sender: "andy12@example.com",  text: "Hello World! kate@example.com",  ErrMessage: "Cannot find the user",  ErrorCode: "1001"  } |

1. Supporting References
   1. Installation Notes

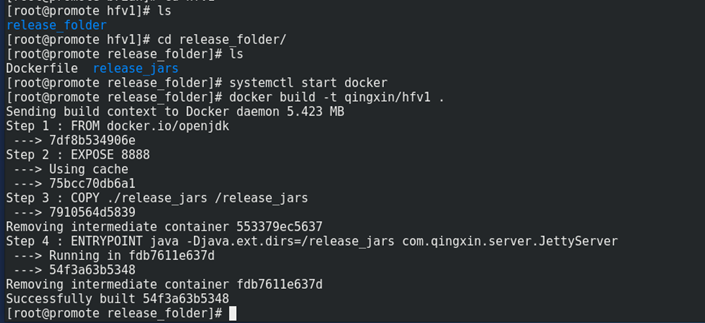
Assumption: docker has been started

1.Download the “release\_folder” folder from

<https://github.com/qx830112/brianhf/tree/master/hfv1>

2. Copy to your Linux folder, eg: ./brian/hfv1/release\_folder

Run command to install: docker build -t qingxin/hfv1 .

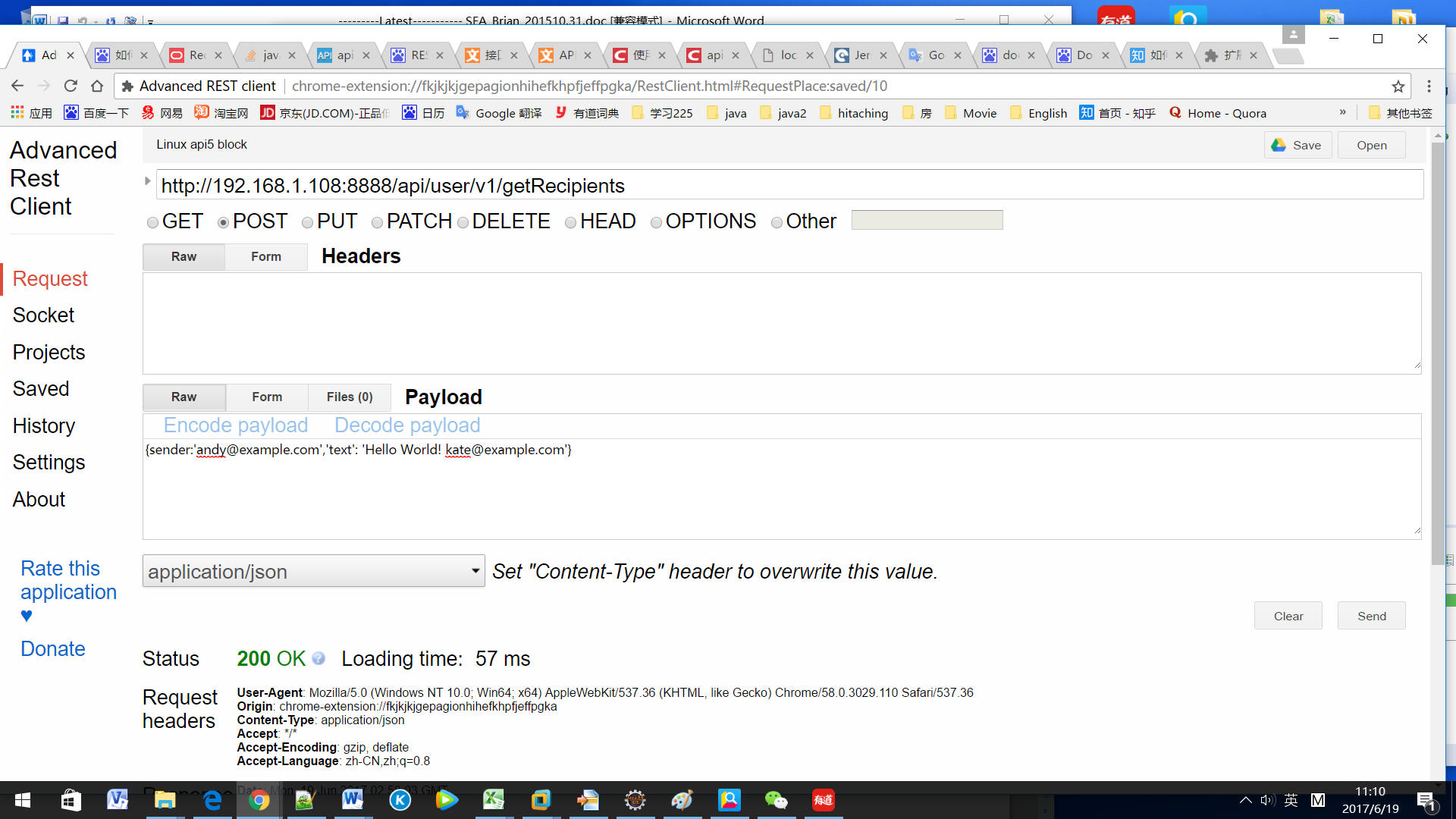


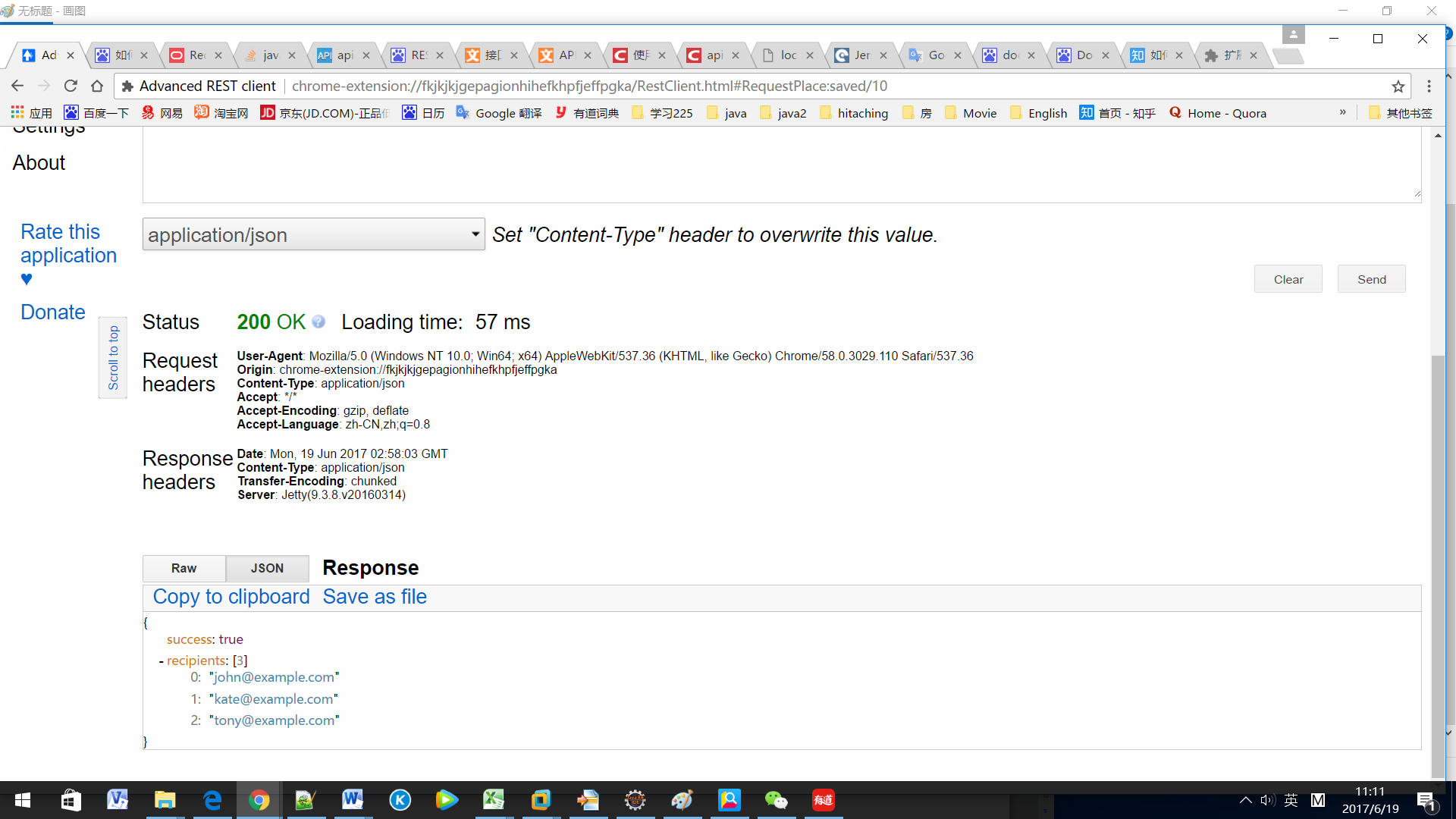
Run command to first start server:

docker run -t -i -p 8888:8888 --name hfv1 qingxin/hfv1

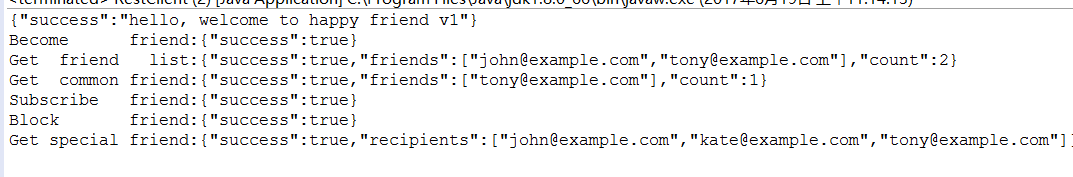
* 1. Testing
* Use Soupui or browser extensions to test the application in windows.

Eg. Advanced REST client





* Run main method in the class com.qingxin.client.RestClient to test.



* Provide JUnit test class: UserControllerTest

