Project Proposal for the
Google Summer of Code Program

FHIR OAuth Smart Apps Integration and OAuth Module Enhancements

from Sanatt Abrol

Contact Details Address : HN 53, Resham Ghar Colony, Jammu, India
Phone : +91-9419209883
E-Mail : sanatt.abrol.in@gmail.com
OpenMRS Talk : https://talk.openmrs.org/users/mavrk/summary
IRC : mavrk
Github : github.com/mavrk

Mentor : Mayank Sharma

Contents

1. Project Proposal

1.1 General Aspects

In the current state, OAuth2 Module only runs on OpenMRS 1.11.4 (max) due to restrictions of underlying API's that are incompatible with higher OpenMRS versions. The main task of this project is to make sure that OAuth2 Module is compatible with the latest 2.x release and demonstrate it using FHIR Module. Another major goal is to make FHIR module work with SMART applications which needs OAuth2 Authorization code grant type based authentication.

1.2 Goals

1.2.1 REST Endpoints

MANAGING CLIENT DEVELOPERS AND APPLICATIONS

As of now, the Client Developer registration is done using the LegacyUI where an Admin creates a Client Developer and the required credentials are generated and passed on to the Client Developer. The Client Developer can then register applications and these applications interact with the Web Services (FHIR, REST Services, etc.) using OAuth2 Module. However, Android/ iOS Applications, OWA's etc should be able to register and interact with OAuth2 Module in a pure RESTful manner. Therefore, the following REST endpoints need to be introduced:

- Registering a new Client Developer
  Consider the case of Bob. Bob is a registered user on the OpenMRS instance and wants to become a client developer so that he can create applications (clients) for that OpenMRS Instance. To do so, Bob needs to POST call to /ws/oauth2/clientdev

  POST http://localhost:8080/openmrs/ws/oauth2/clientdev?username=bob&password=bob
  If Bob is not a client developer, the module will register him as a client developer and send him the following response:

  ```
  200 OK
  {
        "status": client developer
        "description": you are now a client developer
  }
  ```

- Registering a new Client (Application)
  Bob is already a Client Developer by now and wants to register a new Client (application). To do so, Bob needs to make a POST call to /ws/oauth2/client. Bob needs to provide Application Name, Application Description, Website, Redirection URI, Client Type, Scope, Authorization Grant Type

  POST http://localhost:8080/openmrs/ws/oauth2/client?username=bob&password=bob&app-name=name&app-desc=desc&website=www.website.com&reduri=www.reduri.com&client-type=1&scope=r&grant-type=password

  If Bob's credentials match, he will be able to register a new application. Response generated:

```
200 OK
{
	“status”: “client”,
	“client-id”: “some_id”,
	“client-secret”: “some_secret”,
}
```

However, if Bob’s credentials don’t match or he is not a client developer, he’ll receive the following response:

```
401 Unauthorized
{
	“error”: “unauthorized”,
	“error-description”: “Credentials do not match. Unable to authorize”

}
```

- View Clients (Applications)
  Bob being a client developer must be able to view his clients (applications) in a pure RESTful manner. To do so, he needs to make a GET request to /ws/oauth2/client or /ws/oauth2/client/{client-id} and pass on his credentials.

  GET http://localhost:8080/openmrs/ws/oauth2/client?username=bob&password=bob
  This will generate a collection of all clients managed by bob. Sample response:

```
200 OK
{
	“client-id”: “id-1”,
	“client-secret”: “secret 1”,
	“name”: “name 1”,
	“description”: “some description”,
	“website”: “www.somewebsite.com
	“redirection-uri”: “http://localhost:8081/”,
	“client-type”: “WEB_APPLICATION”,
	“scope”: “read, write”,
	“grant-type”: “password”
},
{
	“client-id”: “id-2”,
	“client-secret”: “secret 2”,
	“name”: “name 2”,
	“description”: “some description”,
	“website”: “www.somewebsite.com
	“redirection-uri”: “http://localhost:8081/”,
	“client-type”: “WEB_APPLICATION”,
	“scope”: “read”,
	“grant-type”: “password”
}
```

  However, if the credentials don’t match or Bob is not a client developer, expected response will be:

```
401 Unauthorized
{
	“error”: “unauthorized”,
	“error-description”: “Credentials do not match. Unable to authorize”

}
```

Bob can also view information of a specific client by making a GET request to /ws/oauth2/client/{client-id}

For instance, a GET call to /ws/oauth2/client/id-1 will give a response like

```
200 OK
{
	“client-id”: “id-1”,
	“client-secret”: “secret 1”,
	“name”: “name 1”,
	“description”: “some description”,
	“website”: “www.somewebsite.com
	“redirection-uri”: “http://localhost:8081/”,
	“client-type”: “WEB_APPLICATION”,
	“scope”: “read, write”,
	“grant-type”: “password”
}
```

- Update Clients (Applications)
  If Bob wants to update one of his applications he needs to make a PUT call to /ws/oauth2/client/
  The call must contain fields which need to be updated along with bob’s credentials.

  PUT http://localhost:8080/openmrs/ws/oauth2/client?client-id=id1&client-secret=someSecret&website=www.abcd.com

  If Bob is a client developer and a client with client-id “id1” exists, Bob will be able to update the website to www.abcd.com. All other information about client will remain same and Bob will receive a response code like

```
200 OK
{
	“status”: “client updation”,
	“description”: “client website updated successfully”
}
```

  However, if no client with client-id “id1” exists, Bob will receive the following response code:

```
200 OK
{
	“status”: “client updation”,
	“description”: “no client found”
}
```

  Any malformed request will give a 400 Bad Request.

  If the client secret doesn’t match, Bob will receive the following response:

```
403 Forbidden

{
	“error”: “unauthorized”,
	“error-description”: “client secret failed to match”

}
```

- Delete Clients (Applications)
  If Bob wants to delete a client, he needs to make a DELETE call to /ws/oauth2/client. Bob needs to send client-id and client-secret in the request.

  DELETE http://localhost:8080/openmrs/ws/oauth2/client?client-id=id1&client-secret=somesecret&username=bob&password=bob

  If a client with client-id “id1” is found and the client secret matches and the module is able to verify the identity by matching Bob’s password, the following response will be received:

  ```
  200 OK
  {
        “status”: “client deletion”,
        “description”: “client deleted successfully”
  }
  ```

  However, if no client is found with client-id “id1”, Bob will receive

  ```
  200 OK
  {
        “status”: “client deletion”,
        “description”: “client not found”
  }
  ```

  If client-secret fails to match, Bob will receive

  ```
  403 Forbidden
  {
        “error”: “unauthorized”,
        “error-description”: “client secret failed to match”

  }
  ```

- Issue of new credentials

  If Bob wants new credentials for his client applications, he will use this REST endpoint. To get new credentials he simply needs to make a PUT Call like

  PUT http://localhost:8080/openmrs/ws/oauth2/client?client-id=id1&client-secret=someSecret

  Since this PUT call is made without supplying any fields to change, the OAuth Module will consider this call as an “issue new credentials call”. All data objects of Client will remain same however the client-secret will be re-issued.

  Expected Response:

  ```
  200 OK
  {
        “status”: “new credentials”,
        “client-id”: “id-1”,
        “client-secret”: “newClientSecret”
  }
  ```

1.2.2 Upgrade Dependencies

This involves moving the current state of OAuth Module to the latest OpenMRS 2.x release. The plan is to move everything to Spring 4.x release taking help from http://docs.spring.io/springsecurity/site/migrate/current/3-to-4/html5/migrate-3-to-4-xml.html

At the time of writing this proposal, I am mid-way getting myself familiar with Spring 4.x and Spring OAuth2 Project.

1.2.3 Annotation Model

Annotation based configuration is more common in the Spring 4.x release. This part involves identifying places where annotations can be use.

There are a few places I have found, where we can use Annotations when migrating to Spring 4.x

1. Resource Server

Resource Server hosts the resources [our REST API] the client is interested in. `@EnableResourceServer` annotation, applied on OAuth2 Resource Servers, enables a Spring Security filter that authenticates requests using an incoming OAuth2 token. We can create a class which extends `ResourceServerConfigurerAdapter` class in the Spring Security and then use @EnableResourceServer annotation.

2. Authorization Server

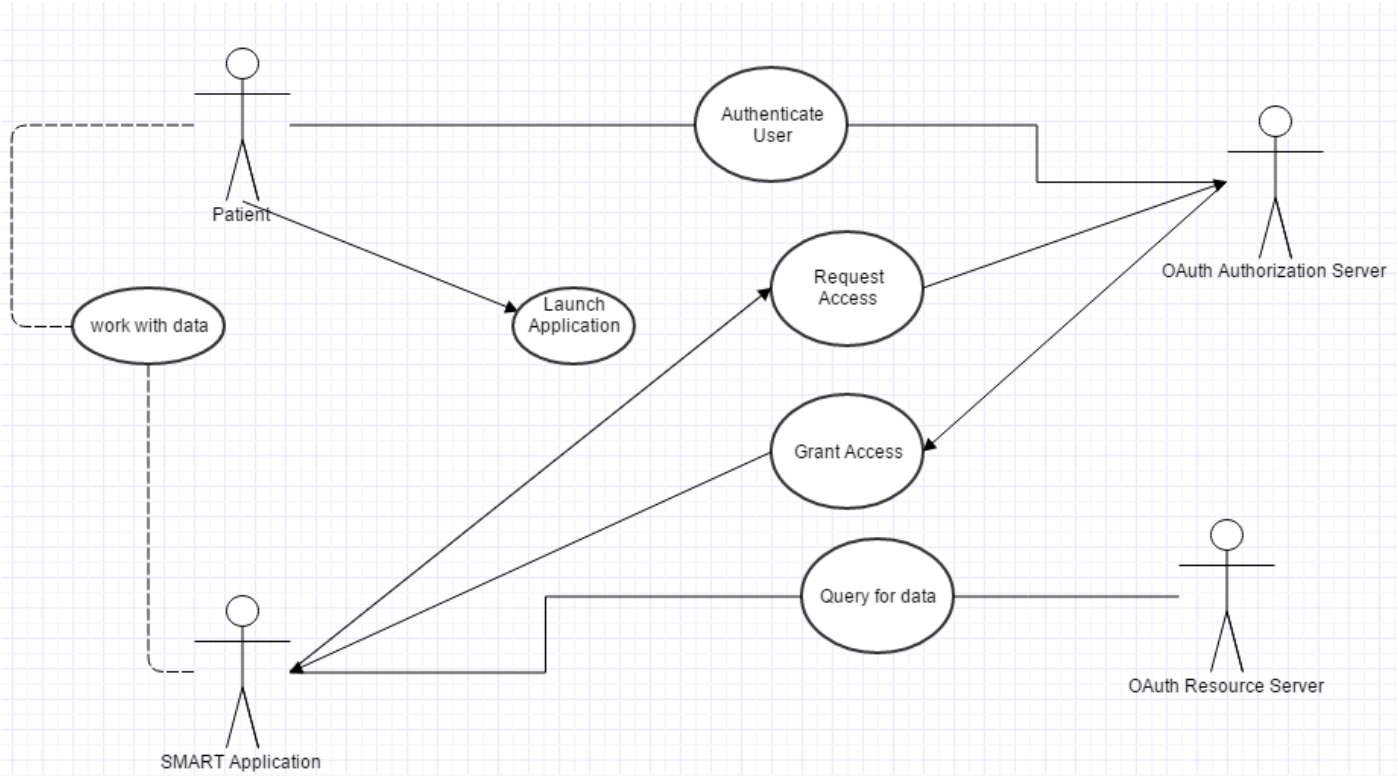
We can create an Authorization server which will be responsible for verifying credentials and if credentials are OK, providing the tokens [refresh-token as well as access-token]. The token store is used to store the token. We can use an in-memory token store.
`@EnableAuthorizationServer` enables an Authorization Server (i.e. an AuthorizationEndpoint and a TokenEndpoint) in the current application context. We can make a class which extends `AuthorizationServerConfigurerAdapter` from Spring Security framework which provides all the necessary methods to configure an Authorization server.

1.2.4 Use-Case Implementation

The main motive of creating an OAuth Module was to encourage module developers and application developers to use security benefits of OAuth2. This project aims to implement a use-case where client developers can see the power of OAuth Security and at the same time encourage them to utilise the benefits of OAuth2.

For this, a simple SMART Application interacting with FHIR module will serve our purpose.

I am thinking of making a simple JavaScript application using SMART on FHIR. This application will demonstrate a use-case where a patient will communicate with OAuth2 Module using the SMART application based on authentication guide given by www.smarthealthit.org



UML Diagram for SMART application

1.3 Deliverables

1.3.1 OAuth2 Module

This is the important and most time taking part. It will involve making the current module compatible with the latest 2.x release and addition of new endpoints for Client Registration and Management.

1.3.2 SMART Application

This will include making a JavaScript based SMART Application which will take advantage of our module and access FHIR Resources. It will serve as a use-case implementation of our module.

1.3.3 Bonus Tasks

If time permits I would love to complete the following bonus tasks

- An OpenMRS OWA demonstrating the implicit grant type flow
- Android Client demonstrating as Password protocol flow

1.4 Project Timeline

April 3 – April 20	Getting familiar with Spring 4.x and identifying more places to use Annotations. Also setting up my coding environment as per mentor's suggestions. Backtracking API calls from JSP files and learning any new frameworks or techniques that are required.
April 20 – May 15	I have my end semester examinations. However, I'll remain in touch and discuss what to code from May 15.
May 15 – May 31	Migration to OpenMRS 2.x release. By this time, I would be familiar with Spring 4.x
May 31 – June 7	Making sure all existing functionality works on the latest release.
June 7 – June 15	I have planned a trip, which is not confirmed yet. If I don't take this vacation, I'll resume with the pending work.
June 15 – June 30	Add new REST Endpoints for Client Registration and management and make sure everything works before the first evaluations.
July 1 – July 20	Build JavaScript based SMART application.
July 20 – July 30	Making sure that SMART application works and our use-case is perfectly demonstrated. Also, I'll try to complete the two major goals (OAuth Module and SMART Application) before the Second evaluations.
July 30 – August 25	Work on bonus deliverables.
August 25 – August 29	Finalising everything and submitting code.

From May 15 – July 5 I have my summer holidays so I'll be able to spend 6-8 hours daily. However, after July 5 I'll be able to spend no more than 4 hours daily.

Personal Information

Who am I? What am I studying?

I am a Computer Science student in my sophomore year from Birla Institute Of Technology, Mesra. I have been coding on Java and related frameworks for around 2 years. My work includes free software, freelancing and hackathon projects. Apart from coding, I love football, swimming and gaming. My other hobbies include music, dance and theatre.

Why am I right for this Project?

I am familiar with the technologies involved and have been contributing to this organization. So, I am familiar with the development process and ways and conventions of this community. Apart from this, I am a quick learner and I believe I have what it takes to finish this project and prepare a module which will offer many benefits to the OpenMRS Community.

Software Development Experience

My software development has revolved mostly around JAVA and related frameworks. Some projects that I've worked on include:

Pharmacy Management Software
https://drive.google.com/open?id=0BwLbWpBHCCwJZ1FQTGd2YllSdmI0bldxSTVaWElvSlpCbTRj

LAN based game tracking application and chat https://github.com/GameHubBITMesra/GameHub

Cyberoam Authentication and Wifi Module https://github.com/mavrk/bitm-cyberoam-client

Rainbow Six Players Auction Host https://github.com/mavrk/PlayerAuction

My other contributions and developments can be found at https://github.com/mavrk

Some technologies which I have used: Java, C++, MySQL, SQLite, Hibernate, Windows PowerShell, Swing, JavaFX, Java Sockets Programming

Link to my talk profile: https://talk.openmrs.org/users/mavrk/summary

Involvements and contributions

I am an active member for the last 2 months and I have contributed on the following issues:

TRUNK- 4947	https://github.com/openmrs/openmrs-core/pull/1993
TRUNK- 5036	https://github.com/openmrs/openmrs-core/pull/2007
TRUNK- 5038	https://github.com/openmrs/openmrs-core/pull/2003
TRUNK-5039	https://github.com/openmrs/openmrs-core/pull/2000
TRUNK- 5044	https://github.com/openmrs/openmrs-core/pull/2002
TRUNK- 5066	https://github.com/openmrs/openmrs-core/pull/2044

Out of these issues, TRUNK-4947 & TRUNK-5066 were very helpful in building concepts and basic knowledge of OpenMRS code base.

Issues I am working on right now:

TRUNK-5042

I have also gained /dev/1 status and got my Smart Developer badge.

My other interactions with community include reviewing pull requests, guiding new members on IRC and Talk.

Preferred Method Of Contact:

OpenMRS Talk: @mavrk

Email: sanatt.abrol.in@gmail.com

Skype: sanattabrol@outlook.com

Any other commitments:

I have a holiday scheduled and I have accommodated space for the same in my proposed timeline. Apart from this, I have no other major commitments from May 15 – 5 July. During this period, I can spend 6-8 hours daily on the project. I have my college starting from 5 July, so 5 July onwards, I can spend 3-5 hours daily on weekdays and 6-8 hours on weekends.

Why this project?

It was the first project I looked when I was finding my way around GSoC 2015 projects. The most interesting thing about this project is the fact that it involves working with different technologies and frameworks. Also, the OpenMRS community has still not utilised OAuth benefits properly and this project aims to change that.