
Seam

SeamSocial

Who**am**I

Marek Schmidt <**maschmid**@redhat.com>

not even on **Facebook**

Short introduction to...

Seam3

SeamPersistence

SeamMail

SeamSocial

SeamRest

SeamJMS

SeamSecurity

SeamRemoting

SeamJCR

SeamFaces

SeamInternational

SeamSpring

Solder

CDI

Servlet

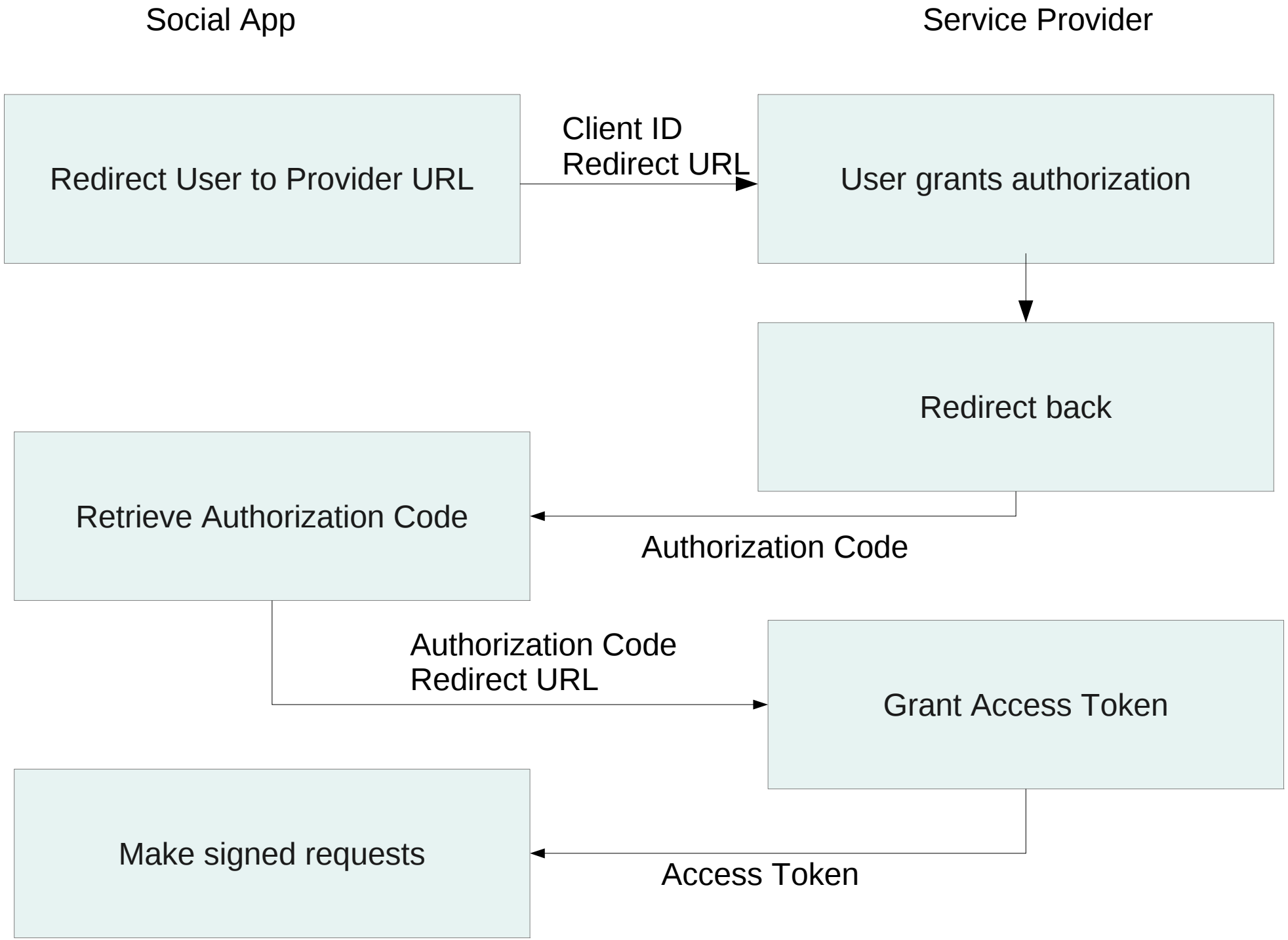
Java EE 6

About **Seam**Social

- “Interact from a Java EE 6 application with social services like Twitter, Facebook, linkedIn, etc...”
- Community project, lead by Antoine Sabot-Durand
- Not finished
- github.com/seam/social

What do
Facebook, **Twit**ter and Linked**In**
have in common?

OAuth


Social App
Service Provider
Redirect User to Provider URL
Client ID
Redirect URL
User grants authorization
Redirect back
Retrieve Authorization Code
Authorization Code
Authorization Code
Redirect URL
Grant Access Token
Make signed requests
Access Token

SeamSocial

Features

- A generic portable REST client API
- A generic API to deal with OAuth 1.0a and 2.0 services
- A generic API to work with JSON serialization and de-serialization
- A generic identification API to retrieve basic user information from a Social Service
- Specific APIs for interacting with Twitter, Facebook and LinkedIn Services
- A multiservices manager API allowing to deal with multiple OAuth applications and sessions in the same application
- An easy way to extend it by creating a new module for a new services

Maven Artifacts

```
<dependency>
  <groupId>org.jboss.seam.social</groupId>
  <artifactId>seam-social-api</artifactId>
  <version>3.1.0.Final</version>
  <scope>compile</scope>
</dependency>

<dependency>
  <groupId>org.jboss.seam.social</groupId>
  <artifactId>seam-social</artifactId>
  <version>3.1.0.Final</version>
  <scope>runtime</scope>
</dependency>

<dependency>
  <groupId>org.jboss.seam.social</groupId>
  <artifactId>seam-social-twitter</artifactId>
  <version>3.1.0.Final</version>
  <scope>compile</scope>
</dependency>
```

Configuration

- Solder XML Configuration (seam-beans.xml)
  - ```
    <o:TwitterServiceImpl>
      <s:modifies />
      <o:Twitter/>
      <o:OAuthApplication
        apiKey="3PUYDZInz50NpK"
        apiSecret="Ee3S1Iz7e3Vuyq8r"
        callback="http://www.foobarter.org/callback.jsf"/>
    </o:TwitterServiceImpl>
    ```
  - ```
    @Qualifier
    @ServiceRelated
    @Target({ TYPE, METHOD, PARAMETER, FIELD })
    @Retention(RUNTIME)
    @Documented
    public @interface Twitter {}
    ```

Injecting the Service Beans

```
@Named
@SessionScoped
public class MyBean implements Serializable {
    ...
    @Inject
    @Twitter
    OAuthService twitterService;
    ...
}
```

Redirect to Authorization URL

```
@Named
@SessionScoped
public class MyBean implements Serializable {
    ...
    @Inject
    @Twitter
    OAuthService twitterService;

    @Inject
    ExternalContext externalContext;

    public void connect() {
        externalContext.redirect(
            twitterService.getAuthorizationUrl());
    }

    ...
}
```

Set verifier

- callback.xhtml

```
<f:metadata>
  <f:viewParam
    name="#{myBean.twitterService.verifierParamName}"
    value="#{myBean.twitterService.verifier}"
    required="true"
    requiredMessage="Error with Twitter. Retry later"/>
    <f:event type="preRenderView"
listener="#{myBean.twitterService.initAccessToken}"/>
</f:metadata>
```

Set verifier

- callback.xhtml

```
<f:metadata>
  <f:viewParam
    name="#{myBean.twitterService.verifierParamName}"
    value="#{myBean.twitterService.verifier}"
    required="true"
    requiredMessage="Error with Twitter. Retry later"/>
    <s:viewAction
action="#{myBean.twitterService.initAccessToken}"/>
</f:metadata>

<navigation-rule>
  <from-view-id>/callback.xhtml</from-view-id>
  <navigation-case>
    <from-action>#{myBean.twitterService.initAccessToken}
    </from-action>
    <if>#{true}</if>
    <to-view-id>/wall.xhtml</to-view-id>
    <redirect/>
  </navigation-case>
</navigation-rule>
```

Using the API

- ```
  @Inject
  @Twitter
  OAuthService twitterService;

  public String getName() {
    if (twitterService.isConnected()) {
      return twitterService.getMyProfile().getFullName();
    }
    ...
  }
  ```
- ```
  @Inject
  @Twitter
  TwitterTimelineService timelineService;

  public void sayHello() {
    timelineService.updateStatus("Hello, world!");
  }
  ```

OAuthSession scope hack

- ```
  <o:OAuthGenericManager>
      <s:modifies/>
      <o:produceSession>
              <s:Produces/>
              <s:ApplicationScoped/>
      </o:produceSession>
  </o:OAuthGenericManager>
  ```
  ```

Twitter API

- OAuthService
  - getAuthorizationUrl, isConnected, initAccessToken, getAccessToken, ...
- TwitterBlockService
- TwitterDirectMessageService
- TwitterFriendService
- TwitterGeoService
- TwitterTimelineService
- TwitterUserService

UserProfile

- ```
  @Inject
  @Twitter
  OAuthService twitterService;

  ...

  UserProfile profile = twitterService.getMyProfile();
  ```

- abstract
- common OAuth properties
  - id
  - fullName
  - profileImageUrl
  ```

Missing the API

```
@Inject
@Twitter
OAuthService twitterService;

@Inject
JsonMapper jsonMapper;

public void isFollower(String source_id, String target_id) {
  RestResponse response =
    OAuthService.sendSignedRequest(RestVerb.GET,
“https://api.twitter.com/1/friendships/show.json?source_id =”
+ source_id + ”&target_id=” + target_id);

    String jsonBody = response.getBody();
    ...
}
```

Twitter JSON API

- https://api.twitter.com/1/friendships/show.json?source_id=123&target_id=456
-
```
{
    "relationship": {
        "target": {
            "id_str": "456",
            "following": true,
            "screen_name": "foo",
            "followed_by": true,
            "id": 456
        },
        "source": {
            "id_str": "123",
            "notifications_enabled": null,
            "can_dm": true,
            "following": true,
            "want_retweets": null,
            "screen_name": "bar",
            "marked_spam": null,
            "all_replies": null,
            "blocking": null,
            "followed_by": true,
            "id": 123
        }
    }
}
```

Using Jackson

- http://jackson.codehaus.org/
- Streaming API
- Tree Model
  - ```
    ObjectNode json = jsonMapper.requestObject(r, ObjectNode.class);
    return json.get("relationship").get("target")
        .get("followed_by").getBooleanValue();
    ```
- Data Binding
  - ```
    ...
    @JsonIgnoreProperties(ignoreUnknown = true)
    class RelationshipMember {
       ...
       @JsonCreator
       RelationshipMember(
            @JsonProperty(“following”) boolean following,
            @JsonProperty(“followed_by”) boolean followedBy) {
                ...
       }
    }
    ```

MultiServiceManager

- `@Inject`
  `MultiServiceManager manager;`

- `Set<String> manager.getListOfServices();`
- `String authUrl = manager.initNewSession(“Twitter”);`
- `OAuthService service = manager.getCurrentService();`
  `service.setVerifier(...);`
  `manager.connectCurrentService();`

- `Set<OAuthSession> getActiveSessions();`
  `setCurrentSession(OAuthSession);`
  `destroyCurrentSession();`

DemoTime I

The Web Client, Seam Social Example application

http://webclient-social.rhcloud.com

SeamSocial

Integration with Seam Security

Seam Security Authenticator

```
<h:form>
  <h:inputText value="#{credentials.username}"/>
  <h:inputSecret value="#{credentials.password}"/>
  <h:commandButton action="#{identity.login}" value="Login"/>
</h:form>
```

```
public class SimpleAuthenticator extends BaseAuthenticator {
  @Inject
  Credential credentials;

  public void authenticate() {
    if ("demo".equals(credentials.getUsername() && ...) {
      setStatus(AuthenticationStatus.SUCCESS);
      setUser(new SimpleUser("demo"));
    }
    else {
      setStatus(AuthenticationStatus.FAILURE);
    }
  }
}
```

OAuthAuthenticator

- seam-beans.xml

```
<security:IdentityImpl>
  <s:modifies/>
  <security:authenticatorName>oauthAuthenticator</security:authenticatorName>
</security:IdentityImpl>
```

-
```
<h:commandButton
    value="Login with Twitter"
    action="#{identity.login}">
  <f:setPropertyActionListener
    target="#{oauthAuthenticator.serviceName}"
    value="Twitter" />
</h:commandButton>
```

- callback.xhtml

```
<f:metadata>
  <f:viewParam name="#{oauthAuthenticator.verifierParamName}"
value="#{oauthAuthenticator.verifier}"
  required="true"
  requiredMessage="Please go back and Try Again !"/>
<s:viewAction action="#{oauthAuthenticator.connect}"/>
</f:metadata>
```

OAuthUser

- @Inject
  Identity identity;

  ...
  OAuthUser oauthUser = (OAuthUser)identity.getUser();

- properties
  - serviceName
  - oauthId
  - id = serviceName + “_” + oauthId
  - userProfile

Identity Management

- Seam Security, based on PicketLink IDM

```
@IdentityEntity(IDENTITY_OBJECT)
@Entity
public class IdentityObject {
   ...
   @IdentityProperty(PropertyType.NAME)
   private String name;

   @ManyToOne
   @IdentityProperty(PropertyType.TYPE)
    public IdentityObjectType getType() {
        return type;
    }
}
```

Demo II, FooBarTer

- Login with Twitter
- Post messages (optionally also to your twitter)
- Persist users and messages
- Follow other FooBarTer users
- Display users' Twitter timeline
- http://www.foobarter.org

DemoTime II

FooBarTer, a Seam Social with Security Sample Application

http://www.foobarter.org

RED HAT
DEVELOPER CONFERENCE 2012
February 17–18th

devconf.cz

FACULTY of INFORMATICS
MASARYK UNIVERSITY, BRNO

fedora redhat JBoss