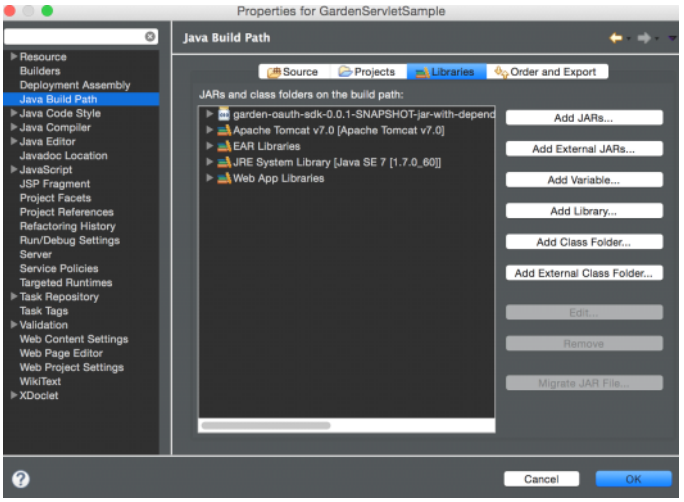
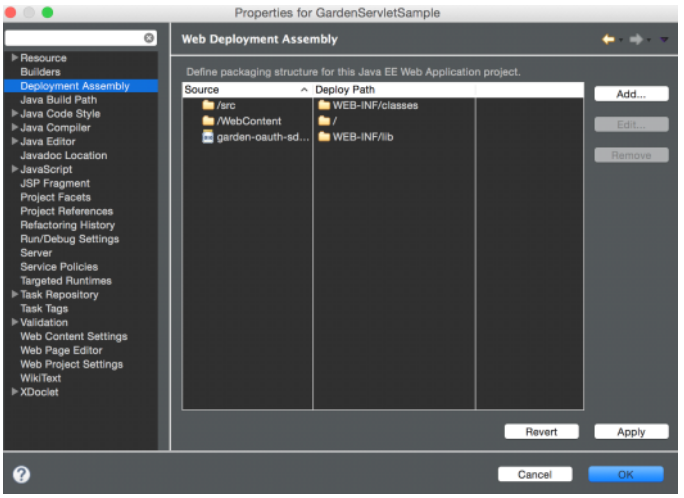
# Tutorial (Java)

2014년 9월 21일 일요일 오후 8:16

*** 샘플 서블릿 컨트롤러 (ServletSample.java, ServletCallbackSample.java)를 다운로드 받아 추가하면 더욱 쉽게 구현할 수 있다.**

1. **Garden Platform Developer Page에 가서 본인의 어플리케이션을 등록하여 Client ID와 Client Secret을 받는다.**
   * 이때 redirect URL은 Authentication 과정을 위한 URL로,
   실제로 로그인 후 redirect될 페이지가 아닌 임의의 콜백 URL로 지정한다. (실제 존재하는 페이지가 아니어도 됨.)
   Ex) http://localhost:8080/GardenServletSample/garden2callback
   앱 등록하는 스크린 샷 넣자.
2. **Garden-Oauth-SDK를 다운로드 받고, 본인의 IDE에 Import 한다.**
   이클립스의 경우,
   1) 해당 프로젝트 우클릭 -> Properties 선택 -> Java Build Path메뉴에서 Libraries 탭 선택 후
   Add External Jars 버튼 클릭 후 SDK 선택.
   2) 해당 프로젝트 우클릭 -> Properties 선택 -> Deployment Assembly 메뉴에서 Add 버튼 클릭 ->
   Java Build Path Entries 선택 후 SDK 선택.




3. **Garden 로그인을 위한 서블릿 클래스(컨트롤러) 1개를 만든 후, AbstractAuthorizationCodeServlet를 상속하여, Implemented된 메소드들을 상속받는다.**

```
package controller;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;

import swssm.garden.sdk.AbstractAuthorizationCodeServlet;
import swssm.garden.sdk.AuthorizationCodeFlow;

public class GardenTutorial extends AbstractAuthorizationCodeServlet{

   private static final long serialVersionUID = 1L;

   @Override
   protected String getRedirectUri(HttpServletRequest arg0)
         throws ServletException, IOException {
      // TODO Auto-generated method stub
      return null;
   }

   @Override
   protected AuthorizationCodeFlow initializeFlow() throws ServletException,
         IOException {
      // TODO Auto-generated method stub
      return null;
   }

}
```

4. **Garden Platform에 앱을 등록할 때 입력한 Redirect URI와, 등록 후 받은 Client Id, Client Secret, Scope를 String타입으로 선언 및 초기화한다. (Scope의 경우 현재 'read'권한만 제공한다.)**

```java
package controller;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;

import swssm.garden.sdk.AbstractAuthorizationCodeServlet;
import swssm.garden.sdk.AuthorizationCodeFlow;

public class GardenTutorial extends AbstractAuthorizationCodeServlet{
   public static final String Garden_Client_ID = "bBgoBDG0bxv8FEOazIjg_XXK7k!-udssvw55ug1O";
   public static final String Garden_Client_Secret = "I7k7aM;8o@yR?SQIuJ19gDvymdO8YPyL4ok!f2=H6q4;bYHztp=7PaGK5;!d:fLJ0ReG.p0@
   public static final String SCOPE = "read";
   public static final String Redirect_Uri = "/GardenServletSample/garden2callback";
   private static final long serialVersionUID = 1L;

   @Override
   protected String getRedirectUri(HttpServletRequest arg0)
         throws ServletException, IOException {
      // TODO Auto-generated method stub
      return null;
   }

   @Override
   protected AuthorizationCodeFlow initializeFlow() throws ServletException,
         IOException {
      // TODO Auto-generated method stub
      return null;
   }

}
```

5. 상속받은 getRedirectUri 메소드내에 Redirect_Uri를 리턴할 수 있도록 다음과 같은 코드를 추가한다.

```java
GenericUrl url = new GenericUrl(req.getRequestURL().toString());
url.setRawPath(Redirect_Uri); // Redirect_uri는 Garden Platform 앱에 등록한 RedirectURL의 URI이다.
return url.build();
```

6. 상속받은 initializeFlow() 메소드 내에 다음과 같은 코드를 추가한다.

```java
AuthorizationCodeFlow flow = new AuthorizationCodeFlow.Builder(
new BasicAuthentication(Garden_Client_ID, Garden_Client_Secret),Garden_Client_ID) // Garden_Client_ID, Garden_Client_Secret은 Garden Platform에서 서 발급 받은 값이다.
.setScopes(Arrays.asList(SCOPE)) // Scope는 현재 'read' 권한만 제공한다.
.setApprovalPrompt("auto")
.build();
return flow;
```

7. Oauth Callback을 위한 서블릿 클래스(컨트롤러)를 하나 더 만든 후, AbstractAuthorizationCodeServlet을 상속하여 Implemented된 메소드와 onError, on Success 메소드를 상속받는다.

```java
public class GardenTutorialCallback extends AbstractAuthorizationCodeCallbackServlet {

   @Override
   protected void onError(HttpServletRequest req, HttpServletResponse resp,
         AuthorizationCodeResponseUrl errorResponse)
         throws ServletException, IOException {
      // TODO Auto-generated method stub
      super.onError(req, resp, errorResponse);
   }

   @Override
   protected void onSuccess(HttpServletRequest req, HttpServletResponse resp,
         TokenResponse tokenResponse) throws ServletException, IOException {
      // TODO Auto-generated method stub
      super.onSuccess(req, resp, tokenResponse);
   }

   private static final long serialVersionUID = 1L;

   @Override
   protected String getRedirectUri(HttpServletRequest arg0)
         throws ServletException, IOException {
      // TODO Auto-generated method stub
      return null;
   }

   @Override
   protected AuthorizationCodeFlow initializeFlow() throws ServletException,
         IOException {
      // TODO Auto-generated method stub
      return null;
   }

}
```

8. 상속받은 onError 메소드 내에 redirectAfterErrorAuth 메소드를 넣어 에러 발생 시 Redirect될 주소를 지정한다.

```java
protected void onError(HttpServletRequest req, HttpServletResponse resp,
AuthorizationCodeResponseUrl errorResponse)
throws ServletException, IOException {
redirectAfterErrorAuth(req,resp, "http://localhost:8080/GardenServletSample/"); // 세 번째 파라미터에는 에러 발생 시 Redirect 될 주소를 넣는다/
}
```

9. 상속받은 onSuccess 메소드 내에도 redirectAfterSuccessAuth 메소드를 넣어 로그인 성공시 Redirect될 주소를 넣는다.

```java
protected void onSuccess(HttpServletRequest req, HttpServletResponse resp,
TokenResponse tokenResponse) throws ServletException, IOException {
redirectAfterSuccessAuth(req,resp, "http://localhost:8080/GardenServletSample/login_success"); // 세번 째 파라미터에는 로그인 성공 시 Redirect될 주소를 넣는다.
}
```

10. getRedirectUri 메소드와 initializeFlow 메소드 내에는 이전에 만들었던 서블릿 클래스 (GardenTutorial.java) 를 참조하여 다음과 같은 코드를 추가한다.

```java
protected String getRedirectUri(HttpServletRequest req)
throws ServletException, IOException {
GenericUrl url = new GenericUrl(req.getRequestURL().toString());
   url.setRawPath(GardenTutorial.Redirect_Uri);
   return url.build();
}
```

```java
protected AuthorizationCodeFlow initializeFlow() throws ServletException,
IOException {
AuthorizationCodeFlow flow = new AuthorizationCodeFlow.Builder(
      new BasicAuthentication(GardenTutorial.Garden_Client_ID,
GardenTutorial.Garden_Client_Secret),GardenTutorial.Garden_Client_ID)
        .setScopes(Arrays.asList(GardenTutorial.SCOPE))
        .setApprovalPrompt("auto")
            .build();
return flow;
}
```

11. **web.xml에 만든 서블릿 컨트롤러를 추가하고 , url-pattern을 다음 주석과 같이 지정한다.**

```
<servlet>
  <servlet-name>ServletSample</servlet-name>
  <servlet-class>controller.GardenTutorial</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>ServletSample</servlet-name>
  <url-pattern>/gardenLogin</url-pattern> <!-- '가든으로 로그인' 버튼을 눌렀을 때, Redirect되는 URI  -->
</servlet-mapping>

<servlet>
  <servlet-name>ServletCallbackSample</servlet-name>
  <servlet-class>controller.GardenTutorialCallback</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>ServletCallbackSample</servlet-name>
  <url-pattern>/garden2callback</url-pattern> <!-- Garden Platform에 앱 등록시 입력했던 Redirect되는 URI -->
</servlet-mapping>
```

12. **Garden Platform 전용 로그인 버튼을 만든 후 , 버튼을 누를 시 web.xml에서 지정한 url-pattern으로 GET Method로 Request 보내도록 한다. (로그인 버튼 이미지 다운로드 받을 수 있도록 할까요..?)**

**EX)**

```
<form action ="gardenLogin" method="get">
    <input type="submit" value="가든 로그인">
</form>
```

13. **만든 로그인 버튼을 눌렀을 때, Authorization 과정에 진행되고, Redirect될 페이지로 성공적으로 Redirect 된 경우 연동에 성공한 것이다. Redirect된 페이지 내 세션에서 다음과 같이 Access Token 정보를 확인할 수 있다. 사용자 정보를 가져오는 등 API를 호출하는 것은 API Guide 참조.**

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
HttpSession session = request.getSession();
String accessToken = (String)session.getAttribute("access_token");
System.out.println("success session tests : " + accessToken);
}
```