

# Adding Facebook Connect via Javascript SDK

Inspired by the [older Facebook-Connect-Example](#) I wrote a Login-panel for the new Facebook Javascript SDK.

For the REST-calls to the new Graph-API I use [RestFB](#)

I used mostly plain JavaScript, but you can put it easily into a wrapper for your convenience.

## FacebookSignInPanel.html

```
<html xmlns:wicket>
  <body>
    <wicket:head>
      <script type="text/javascript" wicket:id="loginCallback"></script>
      <script type="text/javascript" wicket:id="logoutCallback"></script>
    </wicket:head>
    <wicket:panel>
      <fb:login-button autologoutlink="true" perms="email" >
      </fb:login-button>
      <!-- Facebook-API -->
      <div id="fb-root">
      </div>
      <script wicket:id="FBapi"> <!-- calling Facebook API, init and check login -->
      </script>
    </wicket:panel>
  </body>
</html>
```

The panel is as minimalistic as possible. You can simply put it on the header of your design and use it as login-indicator and -button or extend the code to a more sophisticated login-page.

## FacebookSignInPanel.java

```
package x.y.z.authentication;

import java.util.Map;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.wicket.Page;
import org.apache.wicket.RequestCycle;
import org.apache.wicket.ajax.AbstractDefaultAjaxBehavior;
import org.apache.wicket.ajax.AjaxRequestTarget;
import org.apache.wicket.markup.html.JavaScriptPackageResource;
import org.apache.wicket.markup.html.basic.Label;
import org.apache.wicket.markup.html.panel.Panel;
import org.apache.wicket.model.AbstractReadOnlyModel;
import org.apache.wicket.protocol.http.WebRequestCycle;
import x.y.z.AuthWebSession;

/**
 * Minimalistic Panel for Facebook-Login-Management
 *
 * - as a single Button you can put into every Page
 * - as a basis for a full-blown Login-Page
 *
 * @author Markus Lachinger <markus.lachinger @ gmail com>
 */
public class FacebookSignInPanel extends Panel {
    /**
     * default SID
     */
    private static final long serialVersionUID = 1L;
    private static final Log log = LogFactory.getLog(FacebookSignInPanel.class);
```

```

protected String apiId = "yourAPIKey";
protected String secret = "yourSecret";

/** Wrapper to super() */
public FacebookSignInPanel(String id) {
    super(id);
    createPanel();
}

/**
 * This method will load the Facebook-API and manage login-states
 */
public void createPanel() {
    // Add and initialize Facebook-API
    add(JavascriptPackageResource
        .getHeaderContribution("http://connect.facebook.net/en_US/all.js"));
    StringBuffer sb = new StringBuffer();
    sb.append("FB.init({");
    sb.append("    appId: '"+ apiId +"',");
    sb.append("    status: true,");
    sb.append("    cookie: true,");
    sb.append("    xfbml: true");
    sb.append("});");

    // set Event-Handler for Login and Logout
    sb.append("FB.Event.subscribe('auth.login', function(respond){callWicketLogin(respond)});");
    sb.append("FB.Event.subscribe('auth.logout', function(respond){callWicketLogout(respond)});");

    // If the User is already logged in into Facebook, send this to Session
    if(!((AuthWebSession) getSession()).isSignedIn()) {
        sb.append("FB.getLoginStatus(function(response){");
        sb.append("    if (response.session) {");
        sb.append("        FB.api('/me', function(response){");
        sb.append("            if(response.session) {");
        sb.append("                callWicketLogin(response);");
        sb.append("            } else {");
        sb.append("                FB.login(function(res) {");
        sb.append("                    callWicketLogin(res); });");
        sb.append("            });");
        sb.append("        });");
        sb.append("    });");
        sb.append("    else {");
        sb.append("        alert(\"not logged in\");");
        sb.append("    }");
        sb.append("});");
    }

    Label FBapi = new Label("FBapi", sb.toString());
    FBapi.setEscapeModelStrings(false);
    add(FBapi);

    /**
     * Javascript-Wicket-Bridge that gets called after a successful login
     * contains Facebook-UserID, Facebook-SessionId for i.e. JSON and
     * AccessToken (OAuth2)
     */
    final AbstractDefaultAjaxBehavior loginBehavior = new AbstractDefaultAjaxBehavior() {
        protected void respond(final AjaxRequestTarget target) {
            handleLoginEventCallback(target.getPage());
        }
    };
    add(loginBehavior);

    /**
     * Javascript-Wicket-Bridge that gets called after a logout occurred
     */
    final AbstractDefaultAjaxBehavior logoutBehavior = new AbstractDefaultAjaxBehavior() {
        protected void respond(final AjaxRequestTarget target) {
            //log out the user but keep the session
            ((AuthWebSession) getSession()).signOut();
        }
    };
    add(logoutBehavior);
}

```

```

        }
    };
    add(loginCallback);

    Label loginCallback = new Label("loginCallback", new AbstractReadOnlyModel<String>() {
        @Override
        public String getObject() {
            CharSequence url = loginBehavior.getCallbackUrl();
            StringBuffer sb = new StringBuffer();
            sb.append("function callWicketLogin(response) { \n");
            sb.append("            if (response.session) {\n");
            //optional: check permissions here if all necessary are granted!
            //example: (2010/6/1 @ http://developers.facebook.com/docs/reference/javascript
/ FB.login)
//
//                                if (response.perms) {
//                                // user is logged in and granted some permissions.
//                                // perms is a comma separated list of granted permissions
//                                } else {
//                                // user is logged in, but did not grant any permissions
//                                }
            sb.append("                var wcall = wicketAjaxGet('');
            sb.append(url);
            sb.append("&fbid="+ response.session.uid +");
            //SessionId in case you want to use the old API
            sb.append("&fbssid="+ response.session.session_key +");
            //AccessToken for new OAuth2-based requests
            sb.append("&fbaccess_token="+ response.session.access_token");
            sb.append(", function() { }, function() { });\n");
            sb.append("            } else {\n");
            sb.append("                alert(\"Sorry, there was some kind of error
in the Facebook-login. Please try again.\");
            sb.append("            }\n");
            sb.append("}\n");
            return sb.toString();
        }
    });
    loginCallback.setEscapeModelStrings(false);
    loginCallback.setOutputMarkupId(true);
    add(loginCallback);

    Label logoutCallback = new Label("logoutCallback", new AbstractReadOnlyModel<String>() {
        @Override
        public String getObject() {
            CharSequence url = logoutBehavior.getCallbackUrl();
            StringBuffer sb = new StringBuffer();
            sb.append("function callWicketLogout(response) { \n");
            sb.append("alert(\"You have been logged out from facebook. We will log you out
too - see FAQ\");");
            sb.append("    var wcall = wicketAjaxGet('');
            sb.append(url);
            sb.append(", function() { }, function() { });");
            sb.append("    }");
            return sb.toString();
        }
    });
    logoutCallback.setEscapeModelStrings(false);
    logoutCallback.setOutputMarkupId(true);
    add(logoutCallback);
}

public void handleLoginEventCallback(Page p) {
    Map<String, String[]> map = ((WebRequestCycle) RequestCycle.get()).getRequest().
getParameterMap();

    String uid = map.get("fbid")[0];
    String accessToken = map.get("fbaccess_token")[0];

    if(!((AuthWebSession) getSession()).signInFacebook(accessToken, uid)) {
        error("Sorry, something went wrong with the Login - please try again.");
    }
}

```

```

    }

}

```

Here I add the Javascript-API from Facebook, listen to the login- and logout-triggers and check if the user is already logged in in Facebook but not at my page (so no login-trigger will fire).  
If the login is triggered, I extract the interesting data from the response (that is what the Facebook-javascript-request returned to me) and give it to the Session.

The details are inline-commented.

#### AuthWebSession.java

```

package x.y.z.authentication;

import java.util.List;
import java.util.Set;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.wicket.Request;
import org.apache.wicket.Session;
import org.apache.wicket.WicketRuntimeException;
import org.apache.wicket.authentication.AuthenticatedWebSession;
import org.apache.wicket.authorization.strategies.role.Roles;
import org.apache.wicket.injection.web.InjectorHolder;
import org.apache.wicket.spring.injection.annot.SpringBean;

import com.restfb.DefaultFacebookClient;
import com.restfb.FacebookClient;
import com.restfb.FacebookException;
import com.restfb.types.User;

/**
 *
 * @author Markus Lachinger <markus.lachinger AT gmail com>
 */
public class AuthWebSession extends AuthenticatedWebSession {

    /** UserID in database */
    private long userId = -1;

    /** facebook-account? */
    private boolean facebookAcc = false;

    /** Access-Token for Facebook-REST-API */
    private String fbToken = null;

    /** Fixed SerialVersionUID */
    private static final long serialVersionUID = 1L;

    /** Logger */
    private Log log = LogFactory.getLog(AuthWebSession.class);

    /**
     * @return Current authenticated web session
     */
    public static AuthenticatedWebSession get() {
        return (AuthenticatedWebSession) Session.get();
    }

    /**
     * Constructor sets Spring-Injector
     *
     * @param request
     *         Webrequest
     */
}

```

```

    */
    public AuthWebSession(Request request) {
        super(request);
        InjectorHolder.getInjector().inject(this); //Spring Annotations
    }

    /**
     * Wrapper method for signin in a User via his Facebook-account
     *
     * @param accesstoken
     * @param uid
     * @return
     */
    public boolean signInFacebook(final String accesstoken, final String uid) {
        boolean status = authenticateFacebook(accesstoken, uid);
        signIn(status);
        return status;
    }

    /**
     * The user logged in at facebook.
     *
     * @param accesstoken
     *             Access-Token for OAuth2-REST-API of Facebook
     * @param uid
     *             Unique User-ID in facebook
     * @return successfully changed Session
     */
    public boolean authenticateFacebook(String accesstoken, String uid) {
        long uidLong = Long.parseLong(uid);

        if(isSignedIn() && getUserId() != uidLong) {
            //wrong user in current session
            signOut();
            replaceSession();
        }

        getUserId() = uidLong;
        facebookAcc = true;
        fbToken = accesstoken;

        FacebookClient fbClient = new DefaultFacebookClient(accesstoken);
        try {
            User u = fbClient.fetchObject("me", User.class);
            //get some data of the User...
            //i.e. check if user is already in database, add/update entry
            // user-picture via String picLink = u.getLink()+"/picture"
            //u.getPic doesn't seem to work for me
        } catch (FacebookException e) {
            log.error("Facebook-REST-error for uid:"+ uid + " token:"+ accesstoken, e);
            return false;
        }
        return true;
    }

    /**
     * Authenticates this session using the given email-address and password
     *
     * @param email
     * @param password
     * @return True if the user was authenticated successfully
     */
    @Override
    public boolean authenticate(final String email, final String password) {
        //possibility to do the usual username/password-login
    }

    @Override
    public Roles getRoles() {
        //get the roles @see AuthenticatedWebSession
    }

```

```

/**
 * @return Id of the user logged in with this session
 */
public long getUserId() {
    if (isSignedIn()) {
        return userId;
    } else {
        return -1;
    }
}

public void signOut() {
    super.signOut();
    userId=-1;
}
}

```

My Session uses [AuthenticatedWebSession](#) (auth-roles) as base.

Here I handle the login or logout and take care about things like a different user-ID for already logged-in user (another user logged into Facebook while our page was not refreshed in between)

Just add it like this:

```

FacebookSignInPanel x = new FacebookSignInPanel("signInPanel");
add(x);

```

## Errors that can occur:

- Facebook API-Error 100:

```

API Error Code: 100
API Error Description: Invalid parameter
Error Message: next is not owned by the application.

```

Solution: Add a Post-Authorize Rederict URL in you Facebook-App  
see: [Scott Murphy's Blog](#)

I hope this is useful for you,  
Markus

Updates:

- 2010/6/10:

```

* Added createPanel() to the constructor.
* Added common Bug API Error 100

```