

# Custom Authenticator and Login Module in hybrid applications

This is a continuation of Custom Authenticator and Login Module (../).

## Creating client-side authentication components

The application consists of two main *div* elements:

The *AppDiv* element is used to display the application content.

The *AuthDiv* element is used for authentication forms.

When authentication is required, the application hides *AppDiv* and shows *AuthDiv*. When authentication is complete, it does the opposite.

### AppDiv

```

1  <div id="AppDiv"><br />
2    <input type="button" id="getSecretDataButton" value="Call protected adapter proc" onclick="getSecretData()" />
3    <input type="button" class="appButton" value="Logout" onclick="WL.Client.logout('CustomAuthenticatorRealm',{onSucce:
4    <div id="ResponseDiv"></div>
5  </div>

```

Buttons are used to call the *getSecretData* procedure and to log out.

### AuthDiv

```

1  <br />
2  <div id="AuthDiv" style="display: none">
3    <p id="AuthInfo"></p><br />
4    <div id="loginForm"><br />
5      <input type="text" id="AuthUsername" placeholder="Enter username" />
6      <br/>
7      <br/>
8      <input type="password" id="AuthPassword" placeholder="Enter password" />
9      <br/>
10     <input type="button" id="AuthSubmitButton" class="formButton" value="Login" />
11     <input type="button" id="AuthCancelButton" class="formButton" value="Cancel" />
12   </div>
13 </div>

```

*AuthDiv* is styled with *display:none* because it must not be displayed before the server requests the authentication.

## Challenge Handler

Use *WL.Client.createChallengeHandler* to create a challenge handler object. Supply a realm name as a parameter.

```

1  var customAuthenticatorRealmChallengeHandler = WL.Client.createChallengeHandler("CustomAuthenticatorRealm");

```

The *isCustomResponse* function of the challenge handler is called each time a response is received from the server.

It is used to detect whether the response contains data that is related to this challenge handler. It must return **true** or **false**.

```
1 customAuthenticatorRealmChallengeHandler.isCustomResponse = function(response) {}  
2
```

If *isCustomResponse* returns true, the framework calls the *handleChallenge* function. This function is used to perform required actions, such as hide application screen and show login screen.

```
1 customAuthenticatorRealmChallengeHandler.handleChallenge = function(response){}
```

In addition to the methods that the developer must implement, the challenge handler contains functionality that the developer might want to use:

- *submitLoginForm* to send collected credentials to a specific URL. The developer can also specify request parameters, headers, and callback.
- *submitSuccess* to notify the framework that the authentication finished successfully. The framework then automatically issues the original request that triggered the authentication.
- *submitFailure* to notify the framework that the authentication completed with a failure. The framework then disposes of the original request that triggered the authentication

Note: Attach each of these functions to its object. For example: *myChallengeHandler.submitSuccess()*

## isCustomResponse

If the challenge JSON block contains the *authStatus* property, return *true*, otherwise return *false*.

```
1 customAuthenticatorRealmChallengeHandler.isCustomResponse = function(response) {  
2     if (!response || !response.responseJSON) {  
3         return false;  
4     }  
5     if (response.responseJSON.authStatus)  
6         return true;  
7     else  
8         return false;  
9 };
```

## handleChallenge

If the *authStatus* property equals “required”, show the login form, clean up the password input field, and display the error message if applicable.

if *authStatus* equals “complete”, hide the login screen, return to the application, and notify the framework that authentication completed successfully.

```

1 customAuthenticatorRealmChallengeHandler.handleChallenge = function(response){
2     var authStatus = response.responseJSON.authStatus;
3     if (authStatus == "required"){
4         $('#AppDiv').hide();
5         $('#AuthDiv').show();
6         $('#AuthInfo').empty();
7         $('#AuthPassword').val("");
8         if (response.responseJSON.errorMessage){
9             $('#AuthInfo').html(response.responseJSON.errorMessage);
10        }
11    } else if (authStatus == "complete"){
12        $('#AppDiv').show();
13        $('#AuthDiv').hide();
14        customAuthenticatorRealmChallengeHandler.submitSuccess();
15    }
16 };

```

Clicking the **login** button triggers the function that collects the user name and password from HTML input fields and submits them to server. You can set request headers here and specify callback functions.

```

1 $('#AuthSubmitButton').bind('click', function () {
2     var reqURL = '/my_custom_auth_request_url';
3     var options = {};
4     options.parameters = {
5         username : $('#AuthUsername').val(),
6         password : $('#AuthPassword').val()
7     };
8     options.headers = {};
9     customAuthenticatorRealmChallengeHandler.submitLoginForm(reqURL, options, customAuthenticatorRealmChallengeH
10 });

```

Clicking the **cancel** button hides *AuthDiv*, shows *AppDiv* and notifies the framework that authentication failed.

```

1 $('#AuthCancelButton').bind('click', function () {
2     $('#AppDiv').show();
3     $('#AuthDiv').hide();
4     customAuthenticatorRealmChallengeHandler.submitFailure();
5 });

```

The **submitLoginFormCallback** function checks the response for the containing server challenge once again. If the challenge is found, the *handleChallenge* function is called again.

```

1 customAuthenticatorRealmChallengeHandler.submitLoginFormCallback = function(response) {
2     var isLoginFormResponse = customAuthenticatorRealmChallengeHandler.isCustomResponse(response);
3     if (isLoginFormResponse){
4         customAuthenticatorRealmChallengeHandler.handleChallenge(response);
5     }
6 };

```

## Sample application

Click to download  
(<http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/CustomLoginModuleHybridProject.zip>)  
the Studio project.

