

Adapter-based authentication in native iOS applications

This is a continuation of the Adapter-based authentication (../) tutorial.

Creating the client-side authentication components

Create a native iOS application and add the MobileFirst native APIs following the documentation.

Storyboard

In your storyboard, add a ViewController containing a login form.



ChallengeHandler

Create a MyChallengeHandler class as a subclass of ChallengeHandler

We will implement some of the ChallengeHandler methods to respond to the form-based challenge.

```

1  @interface MyChallengeHandler : ChallengeHandler
2  @property ViewController* vc;
3  //A convenient way of updating the View
4  -(id)initWithViewController: (ViewController*) vc;
5  @end

```

Before calling your protected adapter, make sure to register your challenge handler using WLCClient's registerChallengeHandler.

```

1  [[WLCClient sharedInstance] registerChallengeHandler:[MyChallengeHandler alloc] initWithViewController:self];

```

The isCustomResponse method of the challenge handler is invoked each time that 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 either true or false.

```

1  @implementation MyChallengeHandler
2  //...
3  -(BOOL) isCustomResponse:(WLResponse *)response {
4      NSLog(@"Inside isCustomResponse");
5      if(response && [response getResponseJson]){
6          if ([[response getResponseJson] objectForKey:@"authRequired"]) {
7              NSLog(@"Detected adapter auth - return true");
8              NSString* authRequired = (NSString*) [[response getResponseJson] objectForKey:@"authRequired"];
9              return [authRequired boolValue]; //return if auth is required
10         }
11     }
12     return false;
13 }
14 @end

```

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

```

1  @implementation MyChallengeHandler
2  //...
3  -(void) handleChallenge:(WLResponse *)response {
4      NSLog(@"Inside handleChallenge - need to show form on the screen");
5      LoginViewController* loginController = [self.vc.storyboard instantiateViewControllerWithIdentifier:@"LoginViewControll
6      loginController.challengeHandler = self;
7      [self.vc.navigationController pushViewController:loginController animated:YES];
8  }
9  @end

```

`onSuccess` and `onFailure` get triggers when the authentication ends.

You need to call `submitSuccess` to inform the framework that the authentication process is over, and allow the invocation's success handler to be called.

```

1  @implementation MyChallengeHandler
2  //...
3  -(void) onSuccess:(WLResponse *)response {
4      NSLog(@"inside challenge success");
5      [self.vc.navigationController popViewControllerAnimated:YES];
6      [self submitSuccess:response];
7  }
8  -(void) onFailure:(WLFailResponse *)response {
9      NSLog(@"inside challenge failure");
10     [self submitFailure:response];
11 }

```

In your `LoginViewController`, when the user clicks to submit his credentials, you need to call `submitAdapterAuthentication` to send the credentials to the `submitAuthentication` procedure you wrote previously.

```

1  @implementation LoginViewController
2  /**
3   - (IBAction)login:(id)sender {
4       WLProcedureInvocationData *myInvocationData = [[WLProcedureInvocationData alloc]
5   initWithAdapterName:@"NativeAdapterBasedAdapter"
6   procedureName:@"submitAuthentication"];
7   myInvocationData.parameters = @[self.username.text, self.password.text];
8   [self.challengeHandler submitAdapterAuthentication:myInvocationData options:nil];
9   }

```

Sample application

Click to download

(<http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v630/NativeAdapterBasedAuthProject.zip>)
the Studio project.

Click to download

(<http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v630/iOSNativeAdapterBasedAuthProject.zip>)
the Native project.

