Form-based authentication

Overview

In form-based authentication, the HTML code of a login form is returned in the server response when the application tries to access a protected resource.
Although form-based authentication is best suited for desktop and web environments, where you actually display and use the returned login form, you can also use this authentication mode in mobile applications.

To use form-based authentication, you must use a login module to validate the received credentials.
In this tutorial, you implement a simple form-based authentication mechanism that is based on a user name and a password.

Jump to:

- Creating the client-side authentication components

Configuring the authenticationConfig.xml

The default **authenticationConfig.xml** file already contains a sample realm that is configured to use a form-based authenticator.

```
<realm name="SampleAppRealm" loginModule="StrongDummy">
  <className>com.worklight.core.auth.ext.FormBasedAuthenticator</className
>
</realm>
```

Notice the **StrongDummy** login module that is used for this realm.

```
<loginModule name="StrongDummy">
  <className>com.worklight.core.auth.ext.NonValidatingLoginModule</className
>
</loginModule>
```

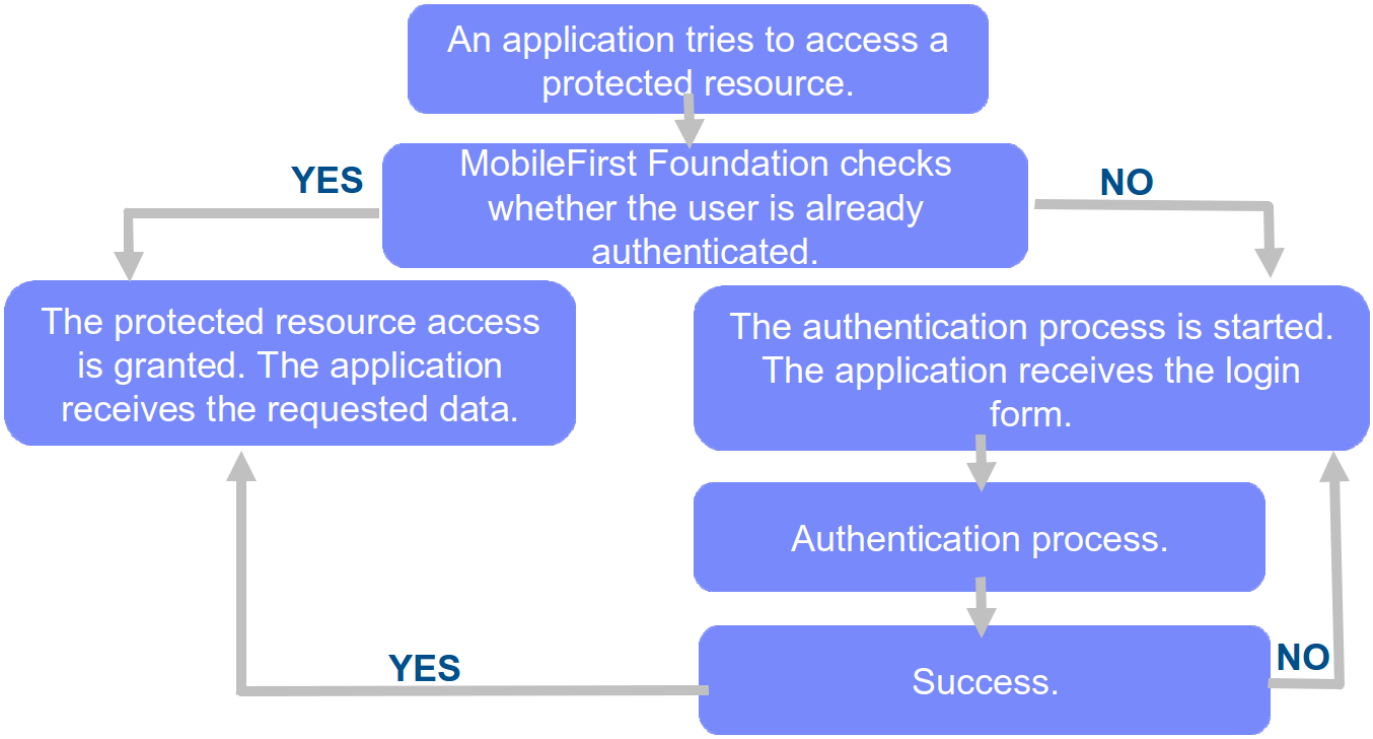
NonValidatingLoginModule means that the user credentials are not validated. In other words: any combination of user name and password is valid.

Define a security test that uses the SampleAppRealm. Remember the security test name, to use it in the subsequent steps.

```
<customSecurityTest name="DummyAdapter-securityTest">
  <test isInternalUserID="true" realm="SampleAppRealm" />
</customSecurityTest>
```

Creating the server-side authentication components

The following diagram illustrates the form-based authentication process.



Create an adapter and name it **DummyAdapter**.
Add a **getSecretData** procedure and protect it with the security test that you created in previous slides.

```
<procedure name="getSecretData" securityTest="DummyAdapter-securityTest"/>
```

In this module, the **getSecretData** procedure returns some hardcoded value:

```
function getSecretData(){
 return {
  secretData: '123456'
 };
}
```