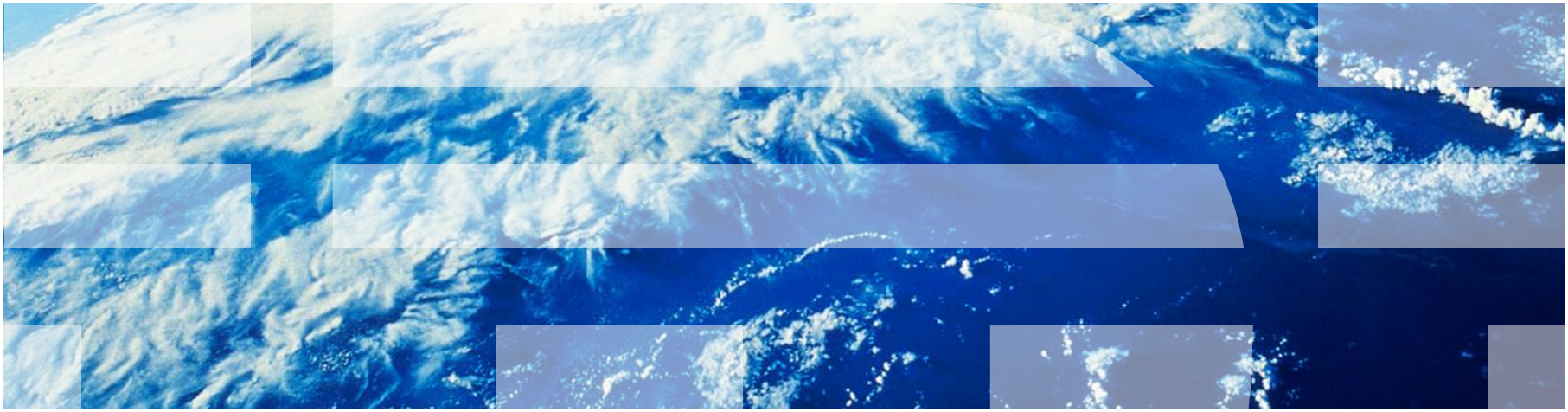
IBM Worklight Foundation V6.2.0 入門

ハイブリッド・アプリケーションでの LDAPLoginModule クラスを使用した LDAP サーバーによるユーザーの認証



2014 年 6 月 20 日

商標

- IBM、IBM ロゴ、ibm.com、WebSphere および Worklight は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。
- Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

- この資料は、事前に IBM の書面による許可を得ずにその一部または全部を複製することは禁じられています。

IBM について

- http://www.ibm.com/ibm/us/en/ を参照してください。

アジェンダ

- LdapLoginModule の概説
- authenticationConfig.xml ファイルの構成
- クライアント・サイドの認証コンポーネントの作成
- 結果の確認

LdapLoginModule の概説

- **LdapLoginModule** クラスを使用すると、OpenLDAP や Active Directory などの LDAP サーバーでユーザーを認証することができます。
- **LdapLoginModule** クラスは、**UserNamePasswordLoginModule** インターフェースを実装しています。したがって、**UsernamePasswordAuthenticator** インターフェースを実装するオーセンティケーターと一緒に使用する必要があります。例えば、**FormBasedAuthenticator** と一緒に使用します。
- **UsernamePasswordAuthenticator** インターフェースの実装方法について詳しくは、チュートリアルおよびサンプルの 9 番目のカテゴリーにあるモジュール『*カスタム・オーセンティケーターとログイン・モジュール (Custom Authenticator and Login Module)*』を参照してください。
- 以降のスライドでは、**LdapLoginModule** クラスを構成および使用して、多様な Worklight エンティティーを保護する方法について説明します。

アジェンダ

- LdapLoginModule の概説
- authenticationConfig.xml ファイルの構成
- クライアント・サイドの認証コンポーネントの作成
- 結果の確認

authenticationConfig.xml ファイルの構成 (1/10)

- 認証レルムを **authenticationConfig.xml** ファイルの `<realms>` セクションに追加し、**LDAPRealm** という名前を付けます。

```
<realms>
    <realm loginModule="LDAPLoginModule" name="LDAPRealm">
        <className>com.worklight.core.auth.ext.FormBasedAuthenticator</className>
        <onLoginUrl>/console</onLoginUrl>
    </realm>
</realms>
```

- **FormBasedAuthenticator** を `<className>` エレメント内で使用します。これによって必要な **UsernamePasswordAuthenticator** インターフェースが実装されるからです。
- このレルムではログイン・モジュールとして **LDAPLoginModule** を使用しますが、これは後で定義します。

authenticationConfig.xml ファイルの構成 (2/10)

- ログイン・モジュールを <loginModules> セクションに追加し、それに **LDAPLoginModule** という名前を付けます。

```
<loginModule name="LDAPLoginModule">
    <className>com.worklight.core.auth.ext.LdapLoginModule</className>
    <parameter name="ldapProviderUrl" value="ldap://10.0.1.2"/>
    <parameter name="ldapTimeoutMs" value="2000"/>
    <parameter name="ldapSecurityAuthentication" value="simple"/>
    <parameter name="validationType" value="searchPattern"/>
    <parameter name="ldapSecurityPrincipalPattern" value="{username}@myserver.com"/>
    <parameter name="ldapSearchFilterPattern" value="(&amp;(objectClass=user)(sAMAccountName={username})(memberof=
    <parameter name="ldapSearchBase" value="dc=myserver,dc=com"/>
</loginModule>
```

- **com.worklight.core.auth.ext.LdapLoginModule** を <className> エレメント内で使用します。

authenticationConfig.xml ファイルの構成 (3/10)

- ログイン・モジュールを <loginModules> セクションに追加し、それに **LDAPLoginModule** という名前を付けます。

```
<loginModule name="LDAPLoginModule">
    <className>com.worklight.core.auth.ext.LdapLoginModule</className>
    <parameter name="ldapProviderUrl" value="ldap://10.0.1.2"/>
    <parameter name="ldapTimeoutMs" value="2000"/>
    <parameter name="ldapSecurityAuthentication" value="simple"/>
    <parameter name="validationType" value="searchPattern"/>
    <parameter name="ldapSecurityPrincipalPattern" value="{username}@myserver.com"/>
    <parameter name="ldapSearchFilterPattern" value="(&amp;(objectClass=user)(sAMAccountName={username})(memberof=
    <parameter name="ldapSearchBase" value="dc=myserver,dc=com"/>
</loginModule>
```

- **ldapProviderUrl** は必須パラメーターです。これは LDAP サーバーの URL を定義します。

authenticationConfig.xml ファイルの構成 (4/10)

- ログイン・モジュールを <loginModules> セクションに追加し、それに **LDAPLoginModule** という名前を付けます。

```
<loginModule name="LDAPLoginModule">
    <className>com.worklight.core.auth.ext.LdapLoginModule</className>
    <parameter name="ldapProviderUrl" value="ldap://10.0.1.2"/>
    <parameter name="ldapTimeoutMs" value="2000"/>
    <parameter name="ldapSecurityAuthentication" value="simple"/>
    <parameter name="validationType" value="searchPattern"/>
    <parameter name="ldapSecurityPrincipalPattern" value="{username}@myserver.com"/>
    <parameter name="ldapSearchFilterPattern" value="(&amp;(objectClass=user)(sAMAccountName={username})(memberof=
    <parameter name="ldapSearchBase" value="dc=myserver,dc=com"/>
</loginModule>
```

- **ldapTimeoutMs** は必須パラメーターです。このパラメーターは、LDAP サーバーの要求のタイムアウトを (ミリ秒単位で) 定義します。

authenticationConfig.xml ファイルの構成 (5/10)

- ログイン・モジュールを <loginModules> セクションに追加し、それに **LDAPLoginModule** という名前を付けます。

```
<loginModule name="LDAPLoginModule">
    <className>com.worklight.core.auth.ext.LdapLoginModule</className>
    <parameter name="ldapProviderUrl" value="ldap://10.0.1.2"/>
    <parameter name="ldapTimeoutMs" value="2000"/>
    <parameter name="ldapSecurityAuthentication" value="simple"/>
    <parameter name="validationType" value="searchPattern"/>
    <parameter name="ldapSecurityPrincipalPattern" value="{username}@myserver.com"/>
    <parameter name="ldapSearchFilterPattern" value="(&amp;(objectClass=user)(sAMAccountName={username})(memberof=
    <parameter name="ldapSearchBase" value="dc=myserver,dc=com"/>
</loginModule>
```

- **ldapSecurityAuthentication** は必須パラメーターです。このパラメーターは、LDAP サーバーが要求する認証のタイプを定義します。通常の値は simple ですが、必要に応じて、より適切な値を LDAP 管理者に問い合わせてください。

authenticationConfig.xml ファイルの構成 (6/10)

- ログイン・モジュールを <loginModules> セクションに追加し、それに **LDAPLoginModule** という名前を付けます。

```
<loginModule name="LDAPLoginModule">
    <className>com.worklight.core.auth.ext.LdapLoginModule</className>
    <parameter name="ldapProviderUrl" value="ldap://10.0.1.2"/>
    <parameter name="ldapTimeoutMs" value="2000"/>
    <parameter name="ldapSecurityAuthentication" value="simple"/>
    <parameter name="validationType" value="searchPattern"/>
    <parameter name="ldapSecurityPrincipalPattern" value="{username}@myserver.com"/>
    <parameter name="ldapSearchFilterPattern" value="(&amp;(objectClass=user)(sAMAccountName={username})(memberof=
    <parameter name="ldapSearchBase" value="dc=myserver,dc=com"/>
</loginModule>
```

- **validationType** は必須パラメーターです。このパラメーターは、実行される検証のタイプを定義します。LdapLoginModule は、**exists**、**searchPattern**、および **custom** という 3 つのタイプの検証をサポートしています。

authenticationConfig.xml ファイルの構成 (7/10)

- **validationType** パラメーターは、以下の値のいずれか 1 つを受け取ります。
  - **exists**: ログイン・モジュールは、提供された資格情報を使用して LDAP バインディングを確立することを試行します。バインディングが正常に確立された時点で、資格情報の検証は成功したと見なされます。
  - **searchPattern**: ログイン・モジュールは、最初に **exists** 検証の実行を試行します。この検証が成功した後で、ログイン・モジュールは、**ldapSearchFilterPattern** パラメーターおよび **ldapSearchBase** パラメーターに従って、検索照会を LDAP サーバー・コンテキストに対して発行します。検索照会で 1 つ以上の項目が返されたら、資格情報の検証は成功と見なされます。
  - **custom**: この値は、カスタム検証ロジックを使用可能にするために使用します。ログイン・モジュールは、**exists** 検証の実行を試行します。この検証が成功した後で、ログイン・モジュールは、`public boolean doCustomValidation(LdapContext ldapCtx, String username)` メソッドを呼び出します。このメソッドは、カスタム Java™ クラスを Worklight プロジェクト内で作成し、`com.worklight.core.auth.ext.UserNamePasswordLoginModule` クラスを拡張することでオーバーライドできます。
    カスタム LDAP 検証タイプについて詳しくは、IBM® Worklight® Foundation のユーザー文書を参照してください。

authenticationConfig.xml ファイルの構成 (8/10)

- ログイン・モジュールを <loginModules> セクションに追加し、それに **LDAPLoginModule** という名前を付けます。

```
<loginModule name="LDAPLoginModule">
    <className>com.worklight.core.auth.ext.LdapLoginModule</className>
    <parameter name="ldapProviderUrl" value="ldap://10.0.1.2"/>
    <parameter name="ldapTimeoutMs" value="2000"/>
    <parameter name="ldapSecurityAuthentication" value="simple"/>
    <parameter name="validationType" value="searchPattern"/>
    <parameter name="ldapSecurityPrincipalPattern" value="{username}@myserver.com"/>
    <parameter name="ldapSearchFilterPattern" value="(&amp;(objectClass=user)(sAMAccountName={username})(memberof=
    <parameter name="ldapSearchBase" value="dc=myserver,dc=com"/>
</loginModule>
```

- **ldapSecurityPrincipalPattern** は必須パラメーターです。このパラメーターは、LDAP セキュリティー・プリンシパルを LDAP サーバーに送信する際のパターンを定義します。{username} プレースホルダーを使用すると、オーセンティケーターから受け取ったユーザー名を挿入できます。

authenticationConfig.xml ファイルの構成 (9/10)

- ログイン・モジュールを <loginModules> セクションに追加し、それに **LDAPLoginModule** という名前を付けます。

```
<loginModule name="LDAPLoginModule">
    <className>com.worklight.core.auth.ext.LdapLoginModule</className>
    <parameter name="ldapProviderUrl" value="ldap://10.0.1.2"/>
    <parameter name="ldapTimeoutMs" value="2000"/>
    <parameter name="ldapSecurityAuthentication" value="simple"/>
    <parameter name="validationType" value="searchPattern"/>
    <parameter name="ldapSecurityPrincipalPattern" value="{username}@myserver.com"/>
    <parameter name="ldapSearchFilterPattern" value="(&amp;(objectClass=user)(sAMAccountName={username}))(memberof=
    <parameter name="ldapSearchBase" value="dc=myserver,dc=com"/>
</loginModule>
```

- **ldapSearchFilterPattern** と **ldapSearchBase** はオプション・パラメーターです。これらのパラメーターは、**searchPattern** 検証タイプにのみ適用されます。

authenticationConfig.xml ファイルの構成 (10/10)

- セキュリティー・テストを **authenticationConfig.xml** ファイルの `<securityTests>` セクションに追加します。
- このセキュリティー・テストは、アダプター・プロシージャーを保護するために使用します。したがって、`<customSecurityTest>` エレメントを使用してください。

```
<customSecurityTest name="LDAPSecurityTest">
    <test isInternalUserID="true" realm="LDAPRealm"/>
</customSecurityTest>
```

- このセキュリティー・テスト名は、以降のスライドで使用するので、覚えておいてください。

アジェンダ

- LdapLoginModule の概説
- authenticationConfig.xml ファイルの構成
- クライアント・サイドの認証コンポーネントの作成
- 結果の確認

クライアント・サイドの認証コンポーネントの作成 (1/14)

- Worklight アプリケーションを作成します。
- このアプリケーションは、以下の 2 つの `<div>` エレメントで構成されます。
  - **`<div id="AppBody">`** エレメントは、アプリケーション・コンテンツの表示に使用されます。
  - **`<div id="AuthBody">`** エレメントは、認証フォームに使用します。
- 認証が要求されると、アプリケーションは `AppBody` エレメントを非表示にして `AuthBody` エレメントを表示します。
- 認証が完了すると、アプリケーションは表示と非表示を逆にします。

クライアント・サイドの認証コンポーネントの作成 (2/14)

- 最初に AppBody を作成します。
- これには基本構造と関数が含まれています。

```
<div id="AppDiv">
    <div class="header">
        <h1>LDAPApp</h1>
    </div>
    <div class="wrapper">
        <input type="button" value="Call protected adapter proc" onclick="getSecretData()" />
        <input type="button" value="Logout"
               onclick="WL.Client.logout('LDAPRealm',{onSuccess: WL.Client.reloadApp})" />
    </div>
    <div id="resultDiv"></div>
</div>
```

- ボタンは、**getSecretData** プロシージャーの呼び出しとログアウトに使用します。

クライアント・サイドの認証コンポーネントの作成 (3/14)

- AuthBody エレメントには、以下のサブエレメントが含まれます。

```
<div id="AuthDiv" style="display:none">
    <div id="loginForm">
        Username:<br/>
        <input type="text" id="usernameInputField" value=""/><br />
        Password:<br/>
        <input type="password" id="passwordInputField" value=""/><br/>
        <input type="button" id="loginButton" value="Login" />
        <input type="button" id="cancelButton" value="Cancel" />
    </div>
</div>
```

  - 「ユーザー名 (Username)」入力フィールドと「パスワード (Password)」入力フィールド
  - 「ログイン (Login)」ボタンと「キャンセル (Cancel)」ボタン
- AuthBody のスタイルは、`display:none` と指定されています。これは、サーバーによって認証が要求される前に表示されてはならないからです。

クライアント・サイドの認証コンポーネントの作成 (4/14)

- 最後に、チャレンジ・ハンドラーを作成します。
- 以下の API を使用してこのハンドラーを作成し、ハンドラーの機能を実装します。

```
var myChallengeHandler = WL.Client.createChallengeHandler("realm-name");

myChallengeHandler.isCustomResponse = function (response){
    return false;
};

myChallengeHandler.handleChallenge = function (response){
};
```

WL.Client.createChallengeHandler メソッドを使用してチャレンジ・ハンドラー・オブジェクトを作成します。レルム名をパラメーターとして指定します。

クライアント・サイドの認証コンポーネントの作成 (5/14)

- 最後に、チャレンジ・ハンドラーを作成します。
- 以下の API を使用してこのハンドラーを作成し、ハンドラーの機能を実装します。

```
var myChallengeHandler = WL.Client.createChallengeHandler("realm-name");

myChallengeHandler.isCustomResponse = function (response){
    return false;
};

myChallengeHandler.handleChallenge = function (response){
};
```

チャレンジ・ハンドラーの isCustomResponse 関数は、サーバーから応答を受け取るたびに呼び出されます。この関数は、このチャレンジ・ハンドラーに関連するデータが応答に含まれているかどうかを検出するために使用します。戻り値は **true** または **false** でなければなりません。

クライアント・サイドの認証コンポーネントの作成 (6/14)

- 最後に、チャレンジ・ハンドラーを作成します。
- 以下の API を使用してこのハンドラーを作成し、ハンドラーの機能を実装します。

```
var myChallengeHandler = WL.Client.createChallengeHandler("realm-name");

myChallengeHandler.isCustomResponse = function (response){
    return false;
};

myChallengeHandler.handleChallenge = function (response){
};
```

isCustomResponse メソッドが **true** を返した場合、フレームワークは **handleChallenge** 関数を呼び出します。この関数は、必要なアクション (例えば、アプリケーション画面の非表示、ログイン画面の表示など) を実行するために使用します。

クライアント・サイドの認証コンポーネントの作成 (7/14)

- チャレンジ・ハンドラーには、開発者が実装しなければならないメソッドに加え、開発者が必要に応じて使用できる機能も含まれています。
  - `submitLoginForm` 関数は、収集した資格情報を特定の URL に送信します。開発者は、要求パラメーター、ヘッダー、およびコールバックを指定することもできます。
  - `submitSuccess` 関数は、認証プロセスが正常に完了したことを Worklight フレームワークに通知します。Worklight フレームワークはその後で、認証をトリガーした元の要求を自動的に発行します。
  - `submitFailure` 関数は、認証プロセスが失敗に終わったことを Worklight フレームワークに通知します。Worklight フレームワークはその後で、認証をトリガーした元の要求を破棄します。

  注: これらの各関数は対応するオブジェクトに付加する必要があります。例えば、次のように使用します。
  `myChallengeHandler.submitSuccess()`
- これらの関数は、以降のスライドでチャレンジ・ハンドラーを実装する際に使用します。

クライアント・サイドの認証コンポーネントの作成 (8/14)

- チャレンジ・ハンドラーを作成します。

```
var LDAPRealmChallengeHandler = WL.Client.createChallengeHandler("LDAPRealm");

LDAPRealmChallengeHandler.isCustomResponse = function(response) {
    if (!response || !response.responseText) {
        return false;
    }
    var idx = response.responseText.indexOf("j_security_check");

    if (idx >= 0){
        return true;
    }
    return false;
};

LDAPRealmChallengeHandler.handleChallenge = f
        $('#AppDiv').hide();
        $('#AuthDiv').show();
        $('#passwordInputField').val('');
};
```

Worklight Server から返されるデフォルトのログイン・フォームには `j_security_check` ストリングが含まれています。チャレンジ・ハンドラーがこのストリングを応答内で検出すると、**true** を返します。

クライアント・サイドの認証コンポーネントの作成 (9/14)

- チャレンジ・ハンドラーを作成します。

```
var LDAPRealmChallengeHandler = WL.Client.createChallengeHandler("LDAPRealm");

LDAPRealmChallengeHandler.isCustomResponse = fu
    if (!response || !response.responseText) {
        return false;
    }
    var idx = response.responseText.indexOf("j

    if (idx >= 0){
        return true;
    }
    return false;
};

LDAPRealmChallengeHandler.handleChallenge = function(response){
        $('#AppDiv').hide();
        $('#AuthDiv').show();
        $('#passwordInputField').val('');
};
```

サーバーがログイン・フォームを送信した (つまりサーバーが認証を要求している) ことをクライアント・アプリケーションが検出すると、そのクライアント・アプリケーションはその後で AppBody を非表示にし、AuthBody を表示し、**passwordInputField** をクリーンアップします。

クライアント・サイドの認証コンポーネントの作成 (10/14)

- チャレンジ・ハンドラーを作成します。

```
$('#loginButton').bind('click', function () {
    var reqURL = '/j_security_check';
    var options = {};
    options.parameters = {
            j_username : $('#usernameInputField').val(),
            j_password : $('#passwordInputField').val()
    };
    options.headers = {};
    LDAPRealmChallengeHandler.submitLoginForm(reqURL, options,
            LDAPRealmChallengeHandler.submitLoginFormCallback);
});

$('#cancelButton').bind('click', function
    $('#AppDiv').show();
    $('#AuthDiv').hide();
    LDAPRealmChallengeHandler.submitFailur
});
```

「ログイン (login)」ボタンをクリックすると、HTML 入力フィールドからユーザー名とパスワードを収集してサーバーに送信する関数がトリガーされます。
ここで要求ヘッダーを設定し、コールバックを指定することが可能です。

クライアント・サイドの認証コンポーネントの作成 (11/14)

- チャレンジ・ハンドラーを作成します。
-

```
$('#loginButton').bind('click', function () {
    var reqURL = '/j_security_check';
    var options = {};
    options.parameters = {
            j_username : $('#usernameInputField').val(),
            j_password : $('#passwordInputField').val()
    };
    options.headers = {};
    LDAPRealmChallengeHandler.submitLoginForm(reqURL, options,
            LDAPRealmChallengeHandler.submitLoginFormCallback);
});

$('#cancelButton').bind('click', function (
    $('#AppDiv').show();
    $('#AuthDiv').hide();
    LDAPRealmChallengeHandler.submitFailure
});
```

フォーム・ベースのオーセンティケーターは、ハードコーディングされた `j_security_check` URL コンポーネントを使用します。このコンポーネントのインスタンスを複数使用することはできません。

クライアント・サイドの認証コンポーネントの作成 (12/14)

- チャレンジ・ハンドラーを作成します。

```
$('#loginButton').bind('click', function (
    var reqURL = '/j_security_check';
    var options = {};
    options.parameters = {
            j_username : $('#usernameInput
            j_password : $('#passwordInput
    };
    options.headers = {};
    LDAPRealmChallengeHandler.submitLoginF
            LDAPRealmChallengeHandler.submitLogin
});

$('#cancelButton').bind('click', function () {
    $('#AppDiv').show();
    $('#AuthDiv').hide();
    LDAPRealmChallengeHandler.submitFailure();
});
```

「キャンセル (cancel)」ボタンをクリックすると、authBody が非表示になり、appBody が表示され、認証が失敗したことが Worklight フレームワークに通知されます。

クライアント・サイドの認証コンポーネントの作成 (13/14)

- チャレンジ・ハンドラーを作成します。

```
LDAPRealmChallengeHandler.submitLoginFormCallback = function(response) {
    var isLoginFormResponse = LDAPRealmChallengeHandler.isCustomResponse(response);
    if (isLoginFormResponse){
        LDAPRealmChallengeHandler.handleChallenge(response);
    } else {
        $('#AppDiv').show();
        $('#AuthDiv').hide();
        LDAPRealmChallengeHandler.submitSuccess();
    }
};
```

コールバック関数は、含まれているサーバー・チャレンジの応答を再び検査します。チャレンジが検出された場合は、`handleChallenge` 関数が再び呼び出されます。

クライアント・サイドの認証コンポーネントの作成 (14/14)

- チャレンジ・ハンドラーを作成します。

```
LDAPRealmChallengeHandler.submitLoginFormCallback = function(response) {
    var isLoginFormResponse = LDAPRealmChallengeHandler.isCustomResponse(response);
    if (isLoginFormResponse){
        LDAPRealmChallengeHandler.handleChallenge(response);
    } else {
        $('#AppDiv').show();
        $('#AuthDiv').hide();
        LDAPRealmChallengeHandler.submitSuccess();
    }
};
```

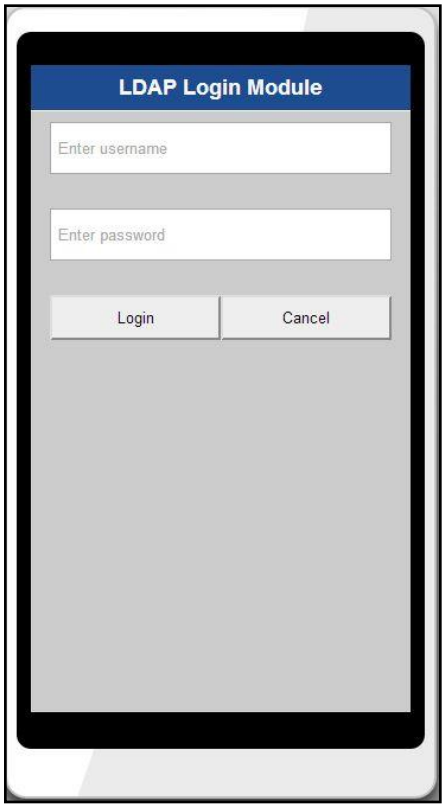
サーバー応答内にチャレンジが存在していないということは、認証が正常に完了したことを意味します。この場合には、AppBody が表示されて、AuthBody が非表示になり、認証に成功したことが Worklight フレームワークに通知されます。

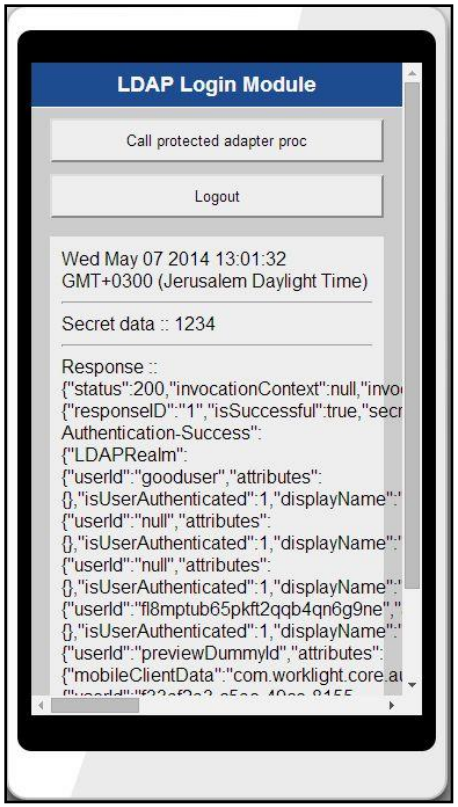
アジェンダ

- LdapLoginModule の概説
- authenticationConfig.xml ファイルの構成
- クライアント・サイドの認証コンポーネントの作成
- 結果の確認

結果の確認

- このトレーニング・モジュールのサンプルは、IBM Worklight Foundation 文書 Web サイト (http://www.ibm.com/mobile-docs) の「入門」ページにあります。





特記事項

- これらの資料は、以下のご使用条件に同意していただける場合に限りご使用いただけます。
- 本書は米国 IBM が提供する製品およびサービスについて作成したものです。
- 本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、または サービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の 製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作または サービスの 評価および検証は、お客様の責任で行っていただきます。
- IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。
  - 〒103-8510
    東京都中央区日本橋箱崎町19番21号
    日本アイ・ビー・エム株式会社
    法務・知的財産
    知的財産権ライセンス渉外
- **以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含む すべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が 禁じられる場合、強行規定の制限を受けるものとします。**
- この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。
- 本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部では ありません。それらの Web サイトは、お客様の責任でご使用ください。
- IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、 自ら適切と信ずる方法で、使用もしくは配布することができるものとします。
- 本プログラムのライセンス保持者で、(i) 独自に作成したプログラムと その他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。
  - IBM Corporation
    Dept F6, Bldg 1
    294 Route 100
    Somers NY 10589-3216
    USA
- 本プログラムに関する上記の情報は、適切な使用条件の下で使用すること ができますが、有償の場合もあります。
- 本書で説明されているライセンス・プログラムまたはその他のライセンス資 料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。
- IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、 他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

著作権使用許諾:

- 本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに 準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性が あることをほのめかしたり、保証することはできません。
- それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次の ように、著作権表示を入れていただく必要があります。
  - © (お客様の会社名) (西暦年).このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。© Copyright IBM Corp. _年を入れる_. All rights reserved.

プライバシー・ポリシーの考慮事項

- サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項を確認ください。
- このソフトウェア・オファリングは、展開される構成に応じて、(アプリケーション・サーバーが生成する) セッション情報を収集するセッションごとの Cookie を使用する場合があります。これらの Cookie は個人情報を含まず、セッション管理のために要求されるものです。加えて、匿名ユーザーの認識および管理のために持続的な Cookie が無作為に生成される場合があります。これらの Cookie も個人情報を含まず、要求されるものです。
- この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。このような目的での Cookie を含む様々なテクノロジーの使用の詳細については、IBM の『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』および『IBM Software Products and Software-as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

サポートおよびコメント

- IBM Worklight の一連の文書、トレーニング資料、および質問をポストできるオンライン・フォーラムはすべて、次の IBM Web サイトからご覧になれます。
  - http://www.ibm.com/mobile-docs
- **サポート**
  - ソフトウェア・サブスクリプション & サポート (ソフトウェア・メンテナンスと呼ばれる場合もあります) は、パスポート・アドバンテージおよびパスポート・アドバンテージ・エクスプレスから購入されたライセンスに含まれています。International Passport Advantage Agreement および IBM International Passport Advantage Express Agreement の追加情報については、次のパスポート・アドバンテージ Web サイトを参照してください。
    - http://www.ibm.com/software/passportadvantage
  - ソフトウェア・サブスクリプション & サポートが有効になっている場合、IBM は、インストールおよび使用法 (ハウツー) に関する短期間の FAQ に対するサポートや、コード関連の質問に対するサポートを提供します。詳しくは、次の IBM ソフトウェア・サポート・ハンドブックを参照してください。
    - http://www.ibm.com/support/handbook
- **ご意見**
  - 本資料に関するご意見をお寄せください。本資料の具体的な誤りや欠落、正確性、編成、題材、または完成度に関するご意見をお寄せください。お寄せいただくご意見は、本マニュアルまたは製品の情報、およびその情報の提示方法に関するもののみとしてください。
  - 製品の技術的な質問および情報、および価格については、担当の IBM 営業所、IBM ビジネス・パートナー、または認定リマーケターにお問い合わせください。
  - IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。IBM またはいかなる組織も、お客様から提示された問題についてご連絡を差し上げる場合にのみ、お客様が提供する個人情報を使用するものとします。
  - どうぞよろしくお願いいたします。
  - 次の IBM Worklight Developer Edition サポート・コミュニティーにご意見をお寄せください。
    - https://www.ibm.com/developerworks/mobile/worklight/connect.html
  - IBM からの回答を希望される場合は、以下の情報をご連絡ください。
    - 氏名
    - 住所
    - 企業または組織
    - 電話番号
    - E メール・アドレス

ありがとうございました