



Oracle A-Team、Lyudmil Pelov、
2013年12月

Oracle WebCenter Portal 11g Release 1 バージョン11.1.1.8.0を使用した タスク・フロー開発ライフ・サイクル



目次

はじめに	3
本書で使用するサンプルについて	3
ステップ1：Fusion Webアプリケーション（GenericServiceConsumer）の作成.....	6
ステップ2：汎用Webサービスの開始	11
ステップ3：Webサービス・プロキシの作成	16
ステップ3a：Webサービス・プロキシ・ファサードの作成	23
ステップ3b：プロキシ・ファサードを使用したデータ・コントロールの作成 （GenericServiceConsumer）	29
ステップ4：バインド・タスク・フローの作成	31
ステップ5：タスク・フロー内でのデータ・コントロールの消費	35
ステップ6：サンプル・タスク・フローのテスト用ページの作成.....	39
ステップ7：Oracle ADFライブラリのJARファイルへのサンプル・タスク・フローのデプロイ.....	44
ステップ8：Deployerプロジェクトの作成.....	50
ステップ9：WebCenter PortalへのDeployerプロジェクトとサンプル・タスク・フロー （WARファイル）のデプロイ	62
ステップ10：WebCenter Portalへのサンプル・タスク・フローの登録	66
ステップ11：WebCenter PortalへのPotalSharedLibraryのデプロイ	75
ステップ12：WebCenter PortalへのWebサービス接続の登録	79
ステップ13：WebCenter Portalリソース・カタログへのサンプル・タスク・フローの追加.....	83
ステップ14：サンプル・タスク・フローの変更と再デプロイ	88
トラブルシューティング	95

はじめに

このドキュメントでは、WebCenter Portal 11.1.1.8.0を使用したサンプル・タスク・フローの開発ライフ・サイクルについて、段階的に説明します。この例では、Oracle JDeveloperでタスク・フローを開発、テストする方法を示した後で、WebCenter Portalでのタスク・フローのデプロイ、再デプロイ、使用方法について説明します。本書は『**Planning a Portal with WebCenter Portal**』に記載された情報を補足するものです。このドキュメントは次のWebサイトから参照できます。

http://docs.oracle.com/cd/E51625_01/webportal.htm

本書で使用するサンプルについて

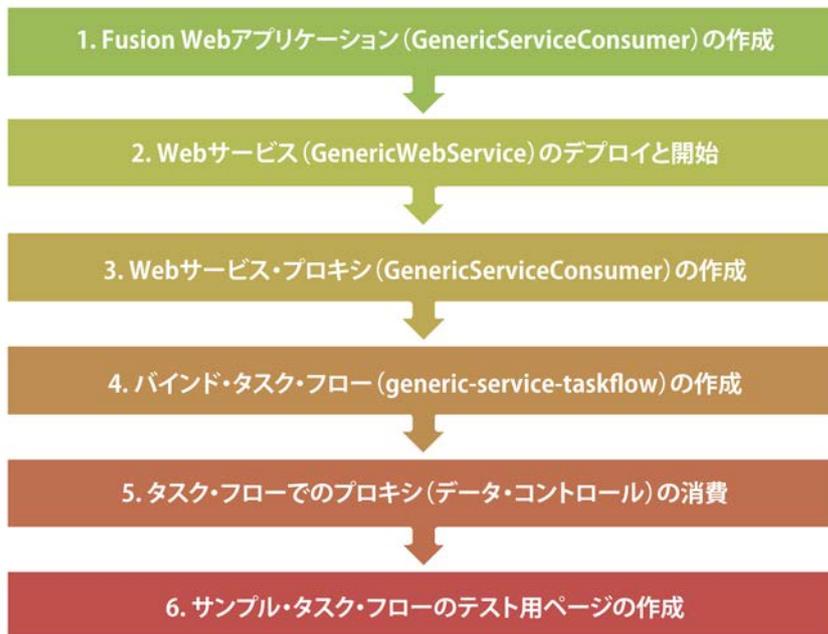
このドキュメントには次のOracle JDeveloperアプリケーションがサンプルとして付属しています。

- **GenericWebService** – シンプルなユーザー・コレクションに情報を付加して返すWebサービスを含みます。このサンプルWebサービスを消費するのがサンプル・タスク・フローです。このドキュメントのステップを実行するには、はじめにGenericWebServiceプロジェクトをJDeveloperにインポートし、WebCenter Portalからアクセスできる管理対象サーバーにWebサービスをデプロイする必要があります。
- **GenericServiceConsumer** – サンプル・タスク・フローと、WebCenter Portalとは切り離してタスク・フローをテストできるテスト・ページを含みます。
- **Deployer** – GenericServiceConsumerアプリケーションの一部であり、WebCenter Portalの管理対象サーバーにサンプル・タスク・フローを（Oracle Application Development Framework (Oracle ADF) 共有ライブラリのWARファイルとして）デプロイするために使用されます。

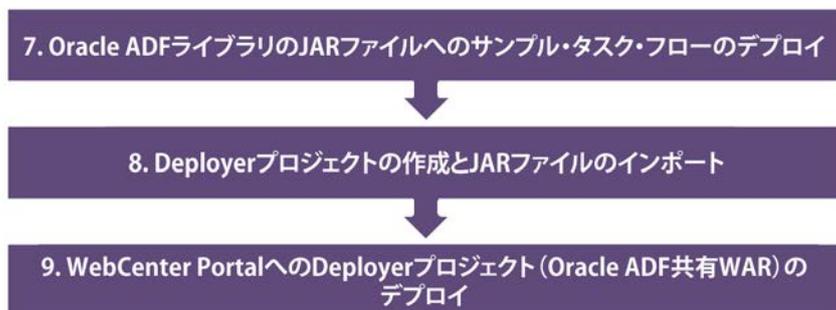
上記サンプルを使用するには、JDeveloper 11.1.1.7と最新のWebCenter Portal拡張バージョン11.1.1.8を使用する必要があります。

本書では、概要レベルのステップに加えて各ステップを構成する個別のアクションを示します。

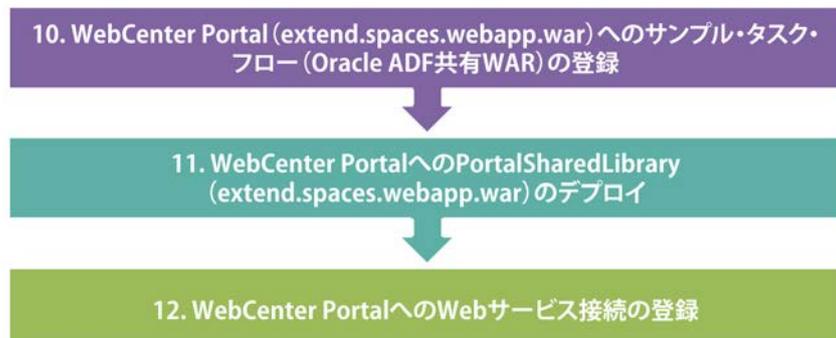
サンプル・タスク・フローの開発とテスト



サンプル・タスク・フローのデプロイ



WebCenter Portalへのサンプル・タスク・フローの登録



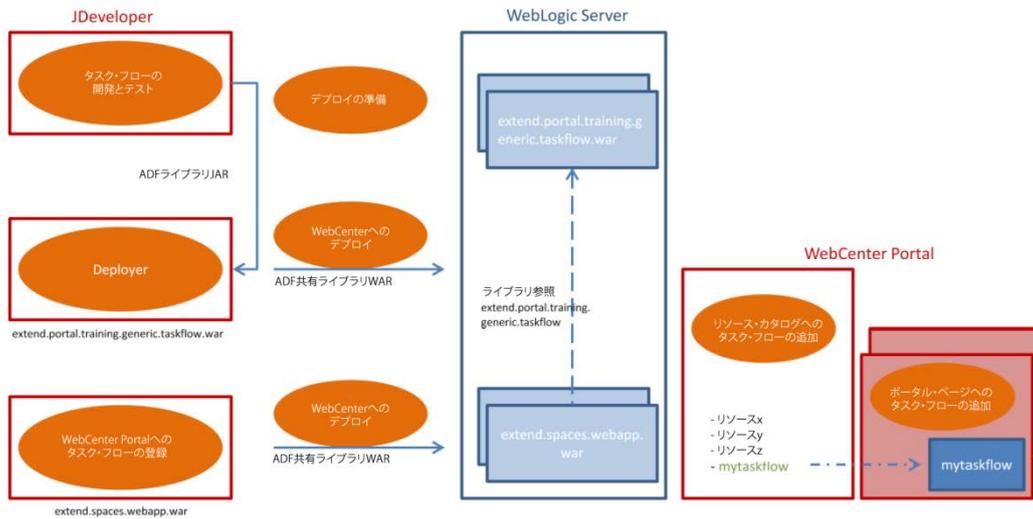
WebCenter Portalでのサンプル・タスク・フローの使用

13. リソース・カタログへのサンプル・タスク・フローの追加とページへのドロップ

サンプル・タスク・フローの更新の再デプロイ

14. サンプル・タスク・フローの変更と再デプロイ

次にフローの全体図を示します。

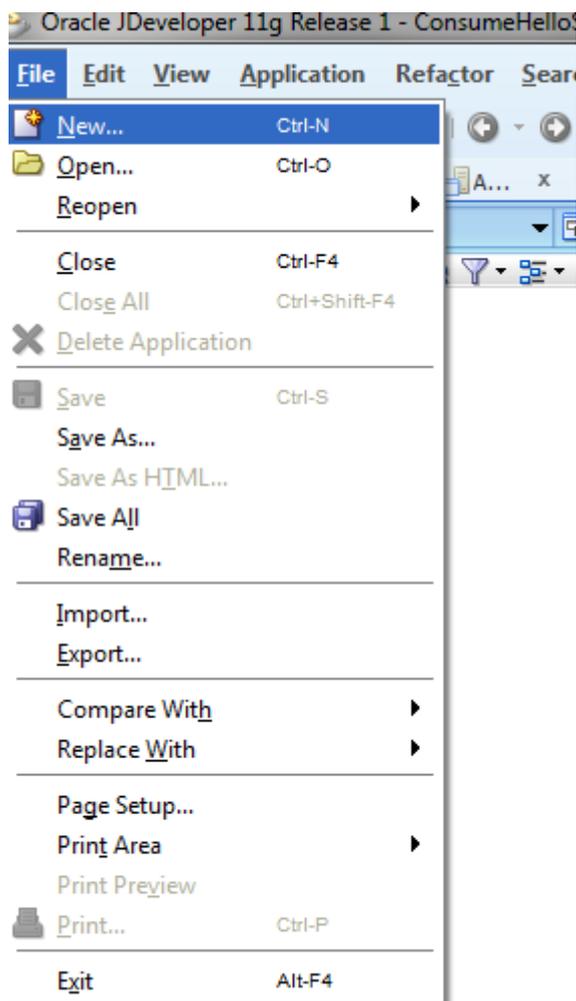


ここからは、プロセスに含まれる各ステップについて詳しく説明します。

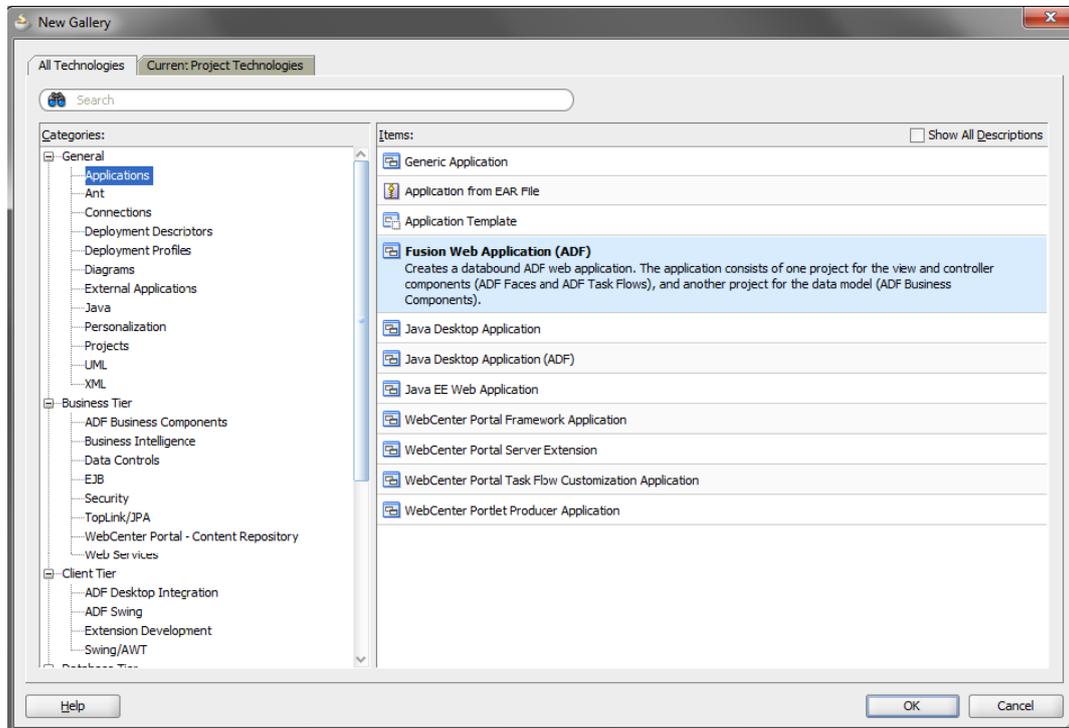
ステップ1：Fusion Webアプリケーション（GenericServiceConsumer）の作成

Fusion Webアプリケーションは、タスク・フローを作成およびテストするためのホストとしての役割を果たします。アプリケーションを作成するには、次のステップに従います。

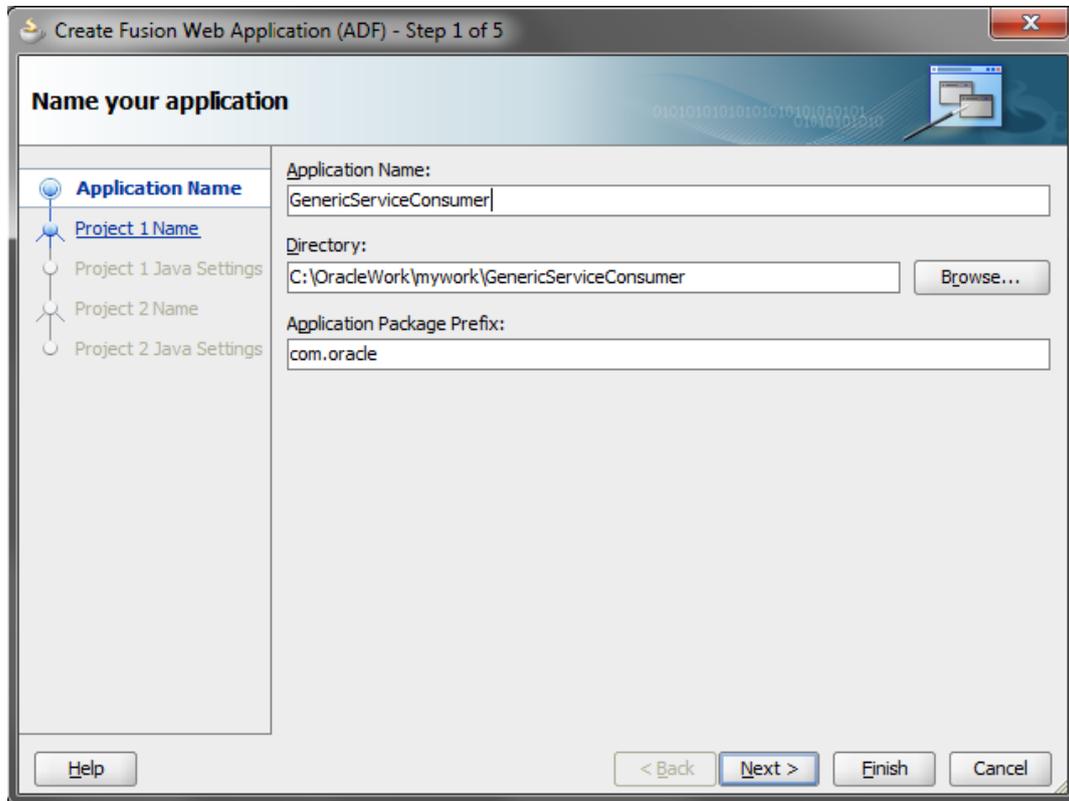
1. JDeveloperで「File」→「New…」を選択します。



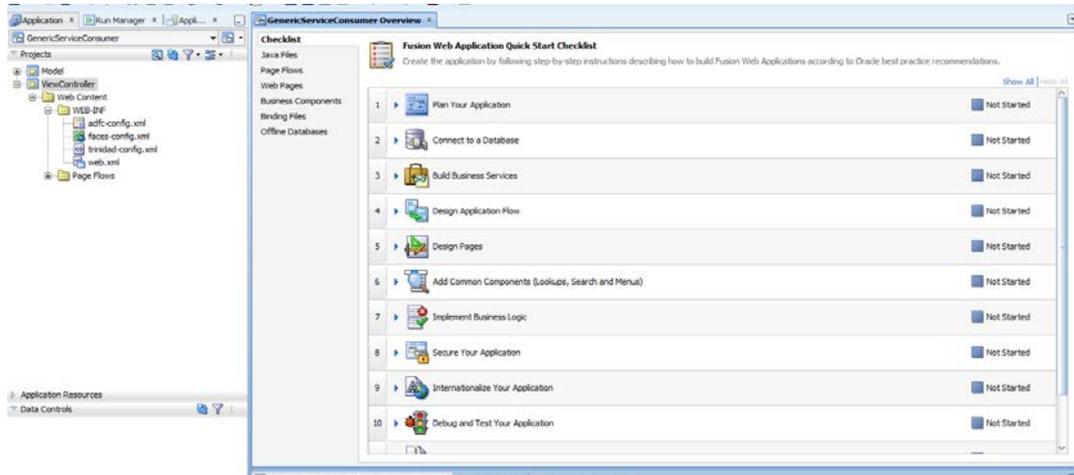
2. New Galleryウィンドウで「General」→「Applications」を選択し、「Fusion Web Application (ADF)」をクリックします。



- Application NameフィールドにGenericServiceConsumerと入力してPackage Prefixを入力したら、「Next」をクリックします。



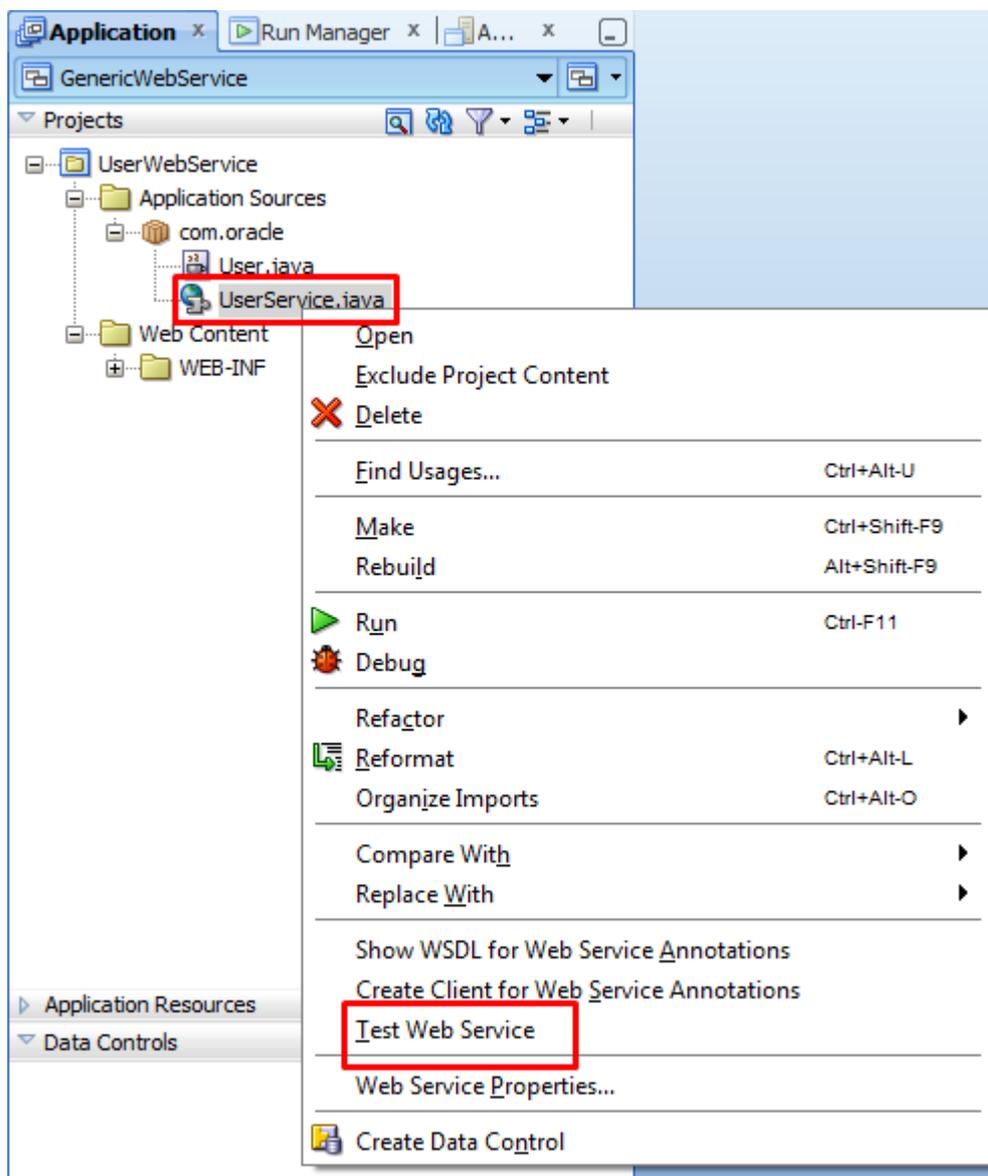
- 次に示すように、GenericServiceConsumerという名前のアプリケーションがOracle ADFプロジェクトとともに表示されます。



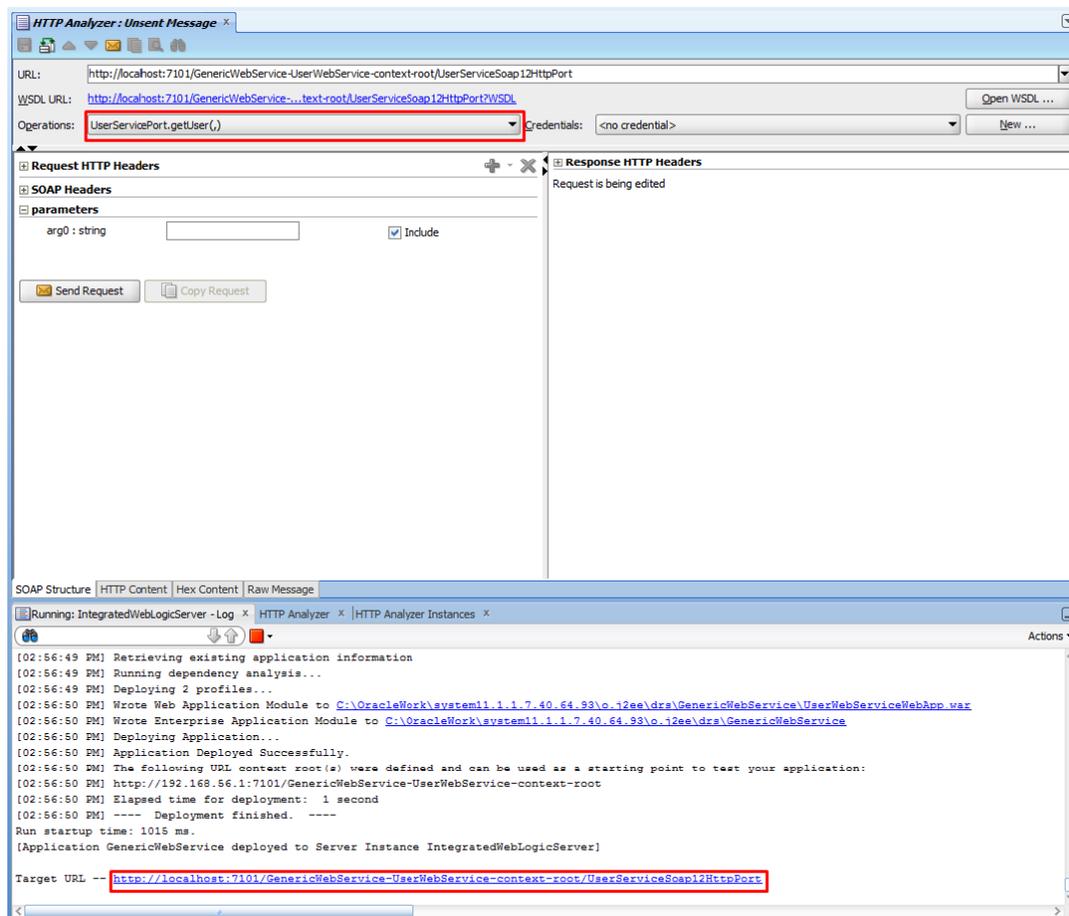
ステップ2：汎用Webサービスの開始

注：このガイドの実行には任意のパブリックWebサービスを使用できます。

1. Webサービスをテストするには、本書で提供しているサンプルの「GenericWebService」アプリケーションを開き、「UserService.java」クラスを右クリックして「Test Web Service」を選択します（下図を参照）。



このステップでは、統合WebLogic ServerにWebサービスをデプロイした後で、このサービスを開始してテストの準備をします（下図を参照）。



Operations ドロップダウンから処理を1つ選択して、特定のWebサービス・メソッドをテストします。たとえば、Fill、Peter、Rickという名前を引数として使用して、getUser()メソッドをテストできます。

HTTP Analyzerを使用して新しいWebサービス・テストを実行するには、統合Webサービス・ログ内のTarget URLリンク（上のスクリーンショットの赤枠内）をクリックします。これによりHTTP Analyzerが再起動され、新しいWebサービス機能をテストできます。テストが終了したら、HTTP Analyzerウィンドウを閉じます。

GenericWebServiceアプリケーションには次の2つのクラスが含まれています。

```
package com.oracle;

public class User {
    private int id;
    private String name;
    private int age;

    public User() {
        super();
    }

    public User(int id, String name, int age) {
        this.id = id;
        this.name = name;
        this.age = age;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getAge() {
        return age;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getId() {
        return id;
    }
}
```

```
}
```

```
package com.oracle;

import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import javax.ws.WebService;

@WebService(serviceName = "UserServiceWS")
public class UserService {
    private List<User> users = new ArrayList<User>();

    public UserService() {
        users.add(new User(1, "Fill", 42));
        users.add(new User(2, "Peter", 32));
        users.add(new User(3, "Rick", 41));
    }

    public String getMessage(String s) {
        if (s == null || s.equalsIgnoreCase(""))
            return "Hello nobody!";

        return "Hello: " + s;
    }

    public void addUser(String name, int age) {
        users.add(new User(users.size() + 1, name, age));
    }

    public User getUser(String name) {
        for (User user : this.users) {
            if (user.getName().equals(name)) {
                return user;
            }
        }
        return null;
    }

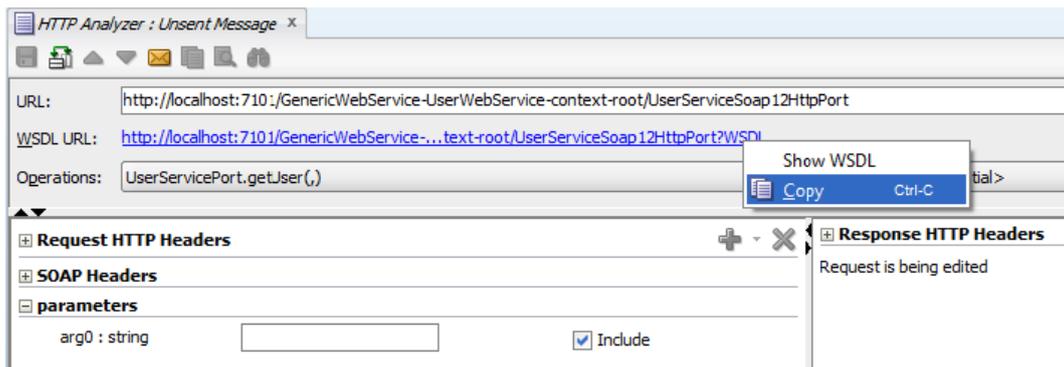
    public List<User> getUsers() {
```

```

return users;
}
}

```

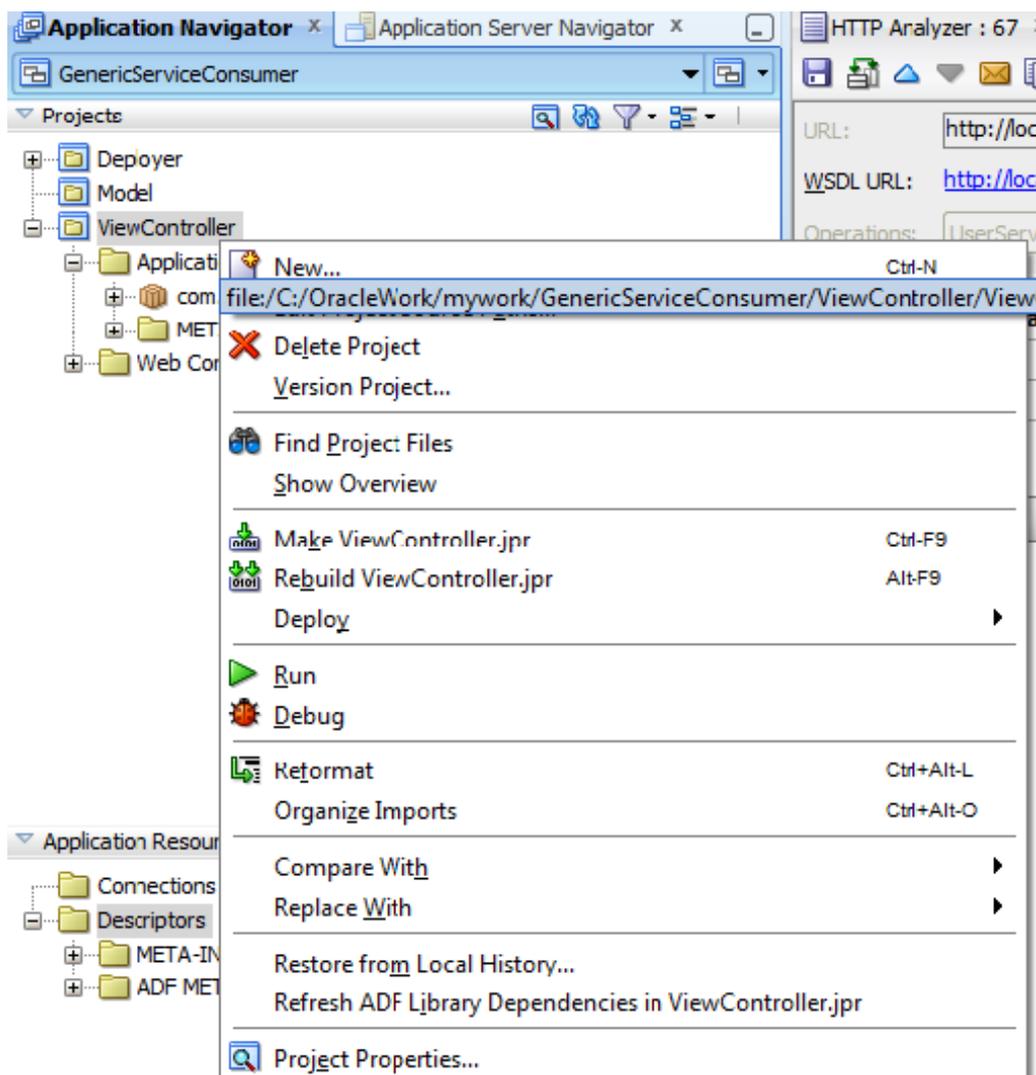
2. 後でjavaプロキシを作成するときに必要なため、WSDL URLをコピーします。このためには、リンクを右クリックして「Copy」を選択します。



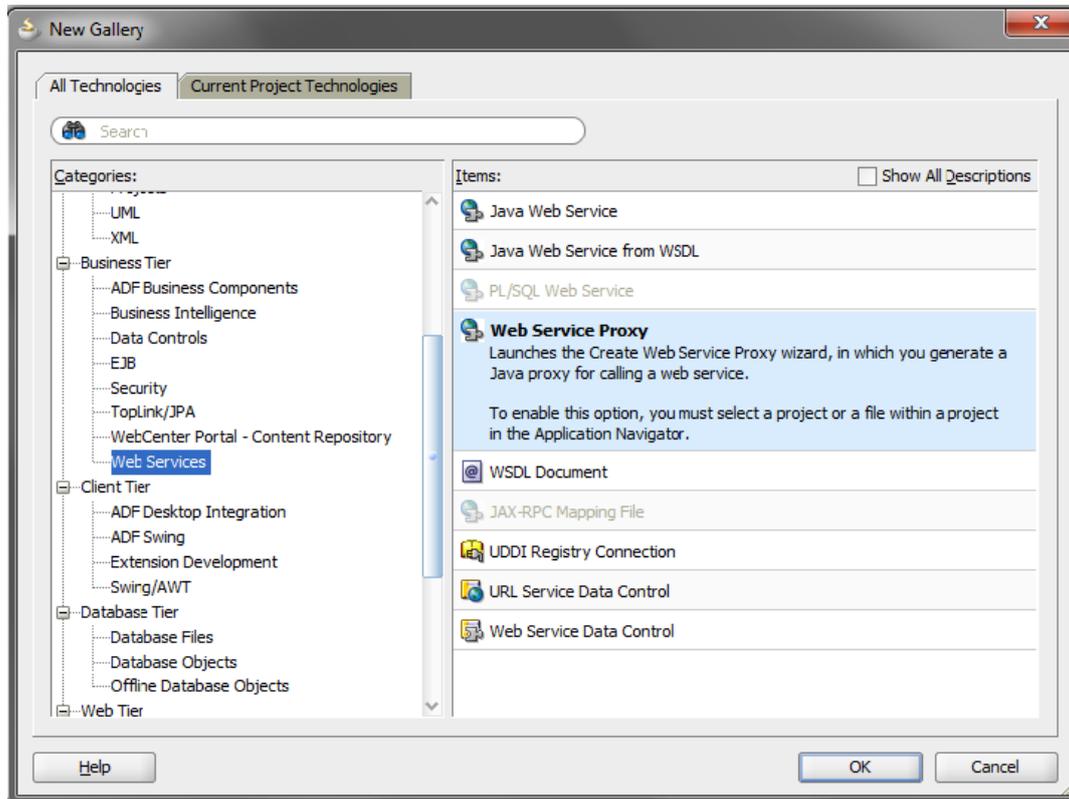
ステップ3：Webサービス・プロキシの作成

このステップでは、汎用Webサービスを消費するためのWebプロキシを作成します。

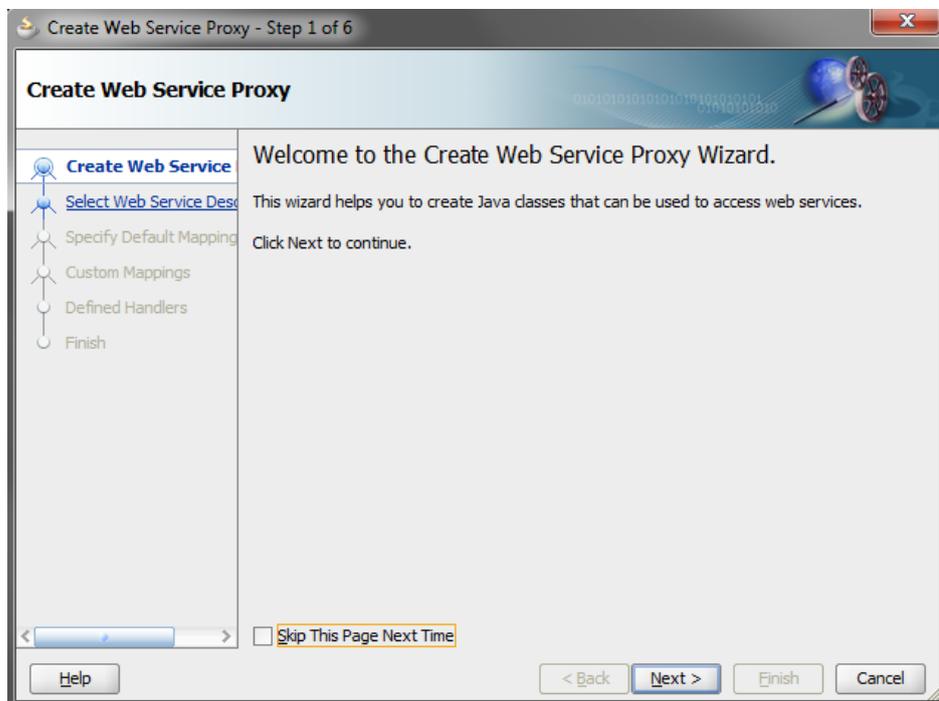
1. JDeveloperで、ステップ1で作成した「GenericServiceConsumer」アプリケーションを開きます。「ViewController」を右クリックして「New…」を選択します。



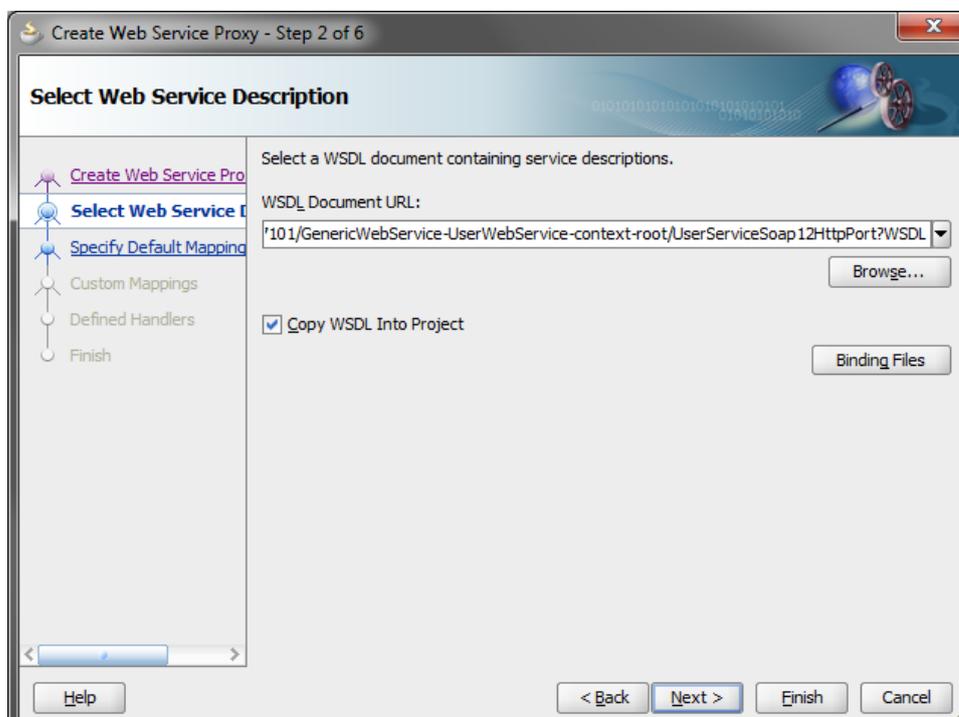
2. New GalleryのAll Technologiesタブで、左側のナビゲーションからBusiness Tierの下にある「Web Services」を選択し、右側から「Web Service Proxy」を選択して「OK」をクリックします。



3. Create Web Service ProxyのWelcome画面で「Next」をクリックします。



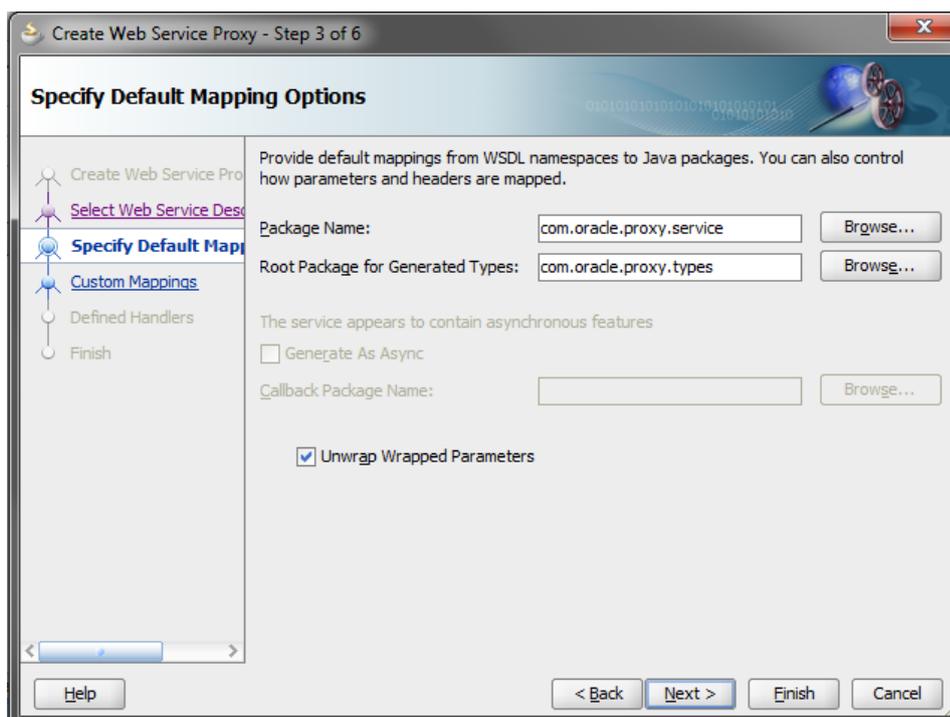
4. 使用するWebサービスのWSDL URLを貼り付け、残りはデフォルトのままにして「Next」をクリックします。



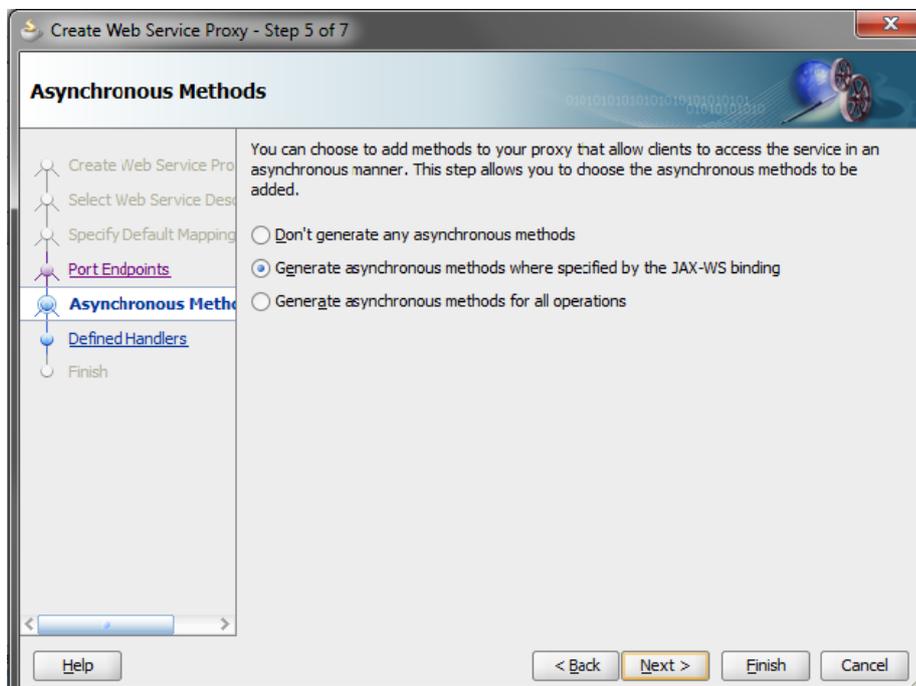
5. インポート後に作成されるパッケージの名前空間を次のように指定し、「Next」をクリックします。

Package Name : com.oracle.proxy.service

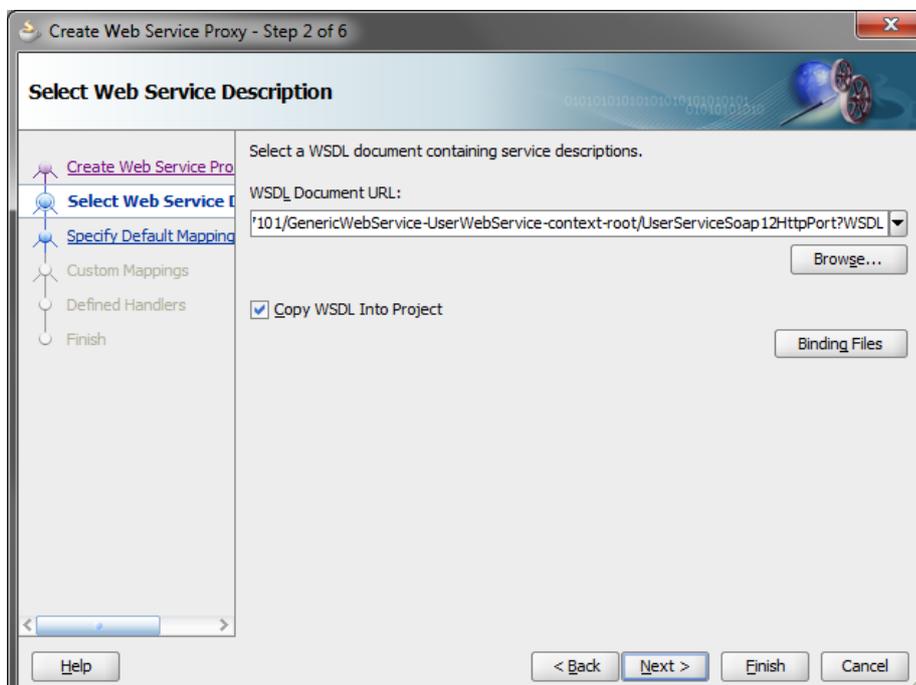
Root Package for Generated Types : com.oracle.proxy.types



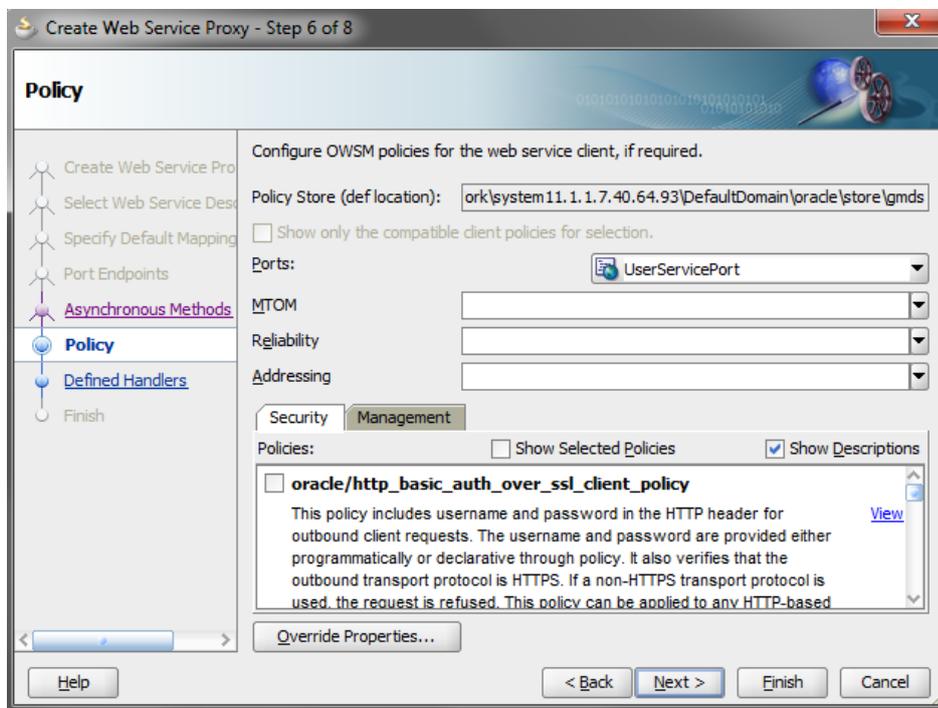
6. エンドポイント構成を確認して「Next」をクリックします。



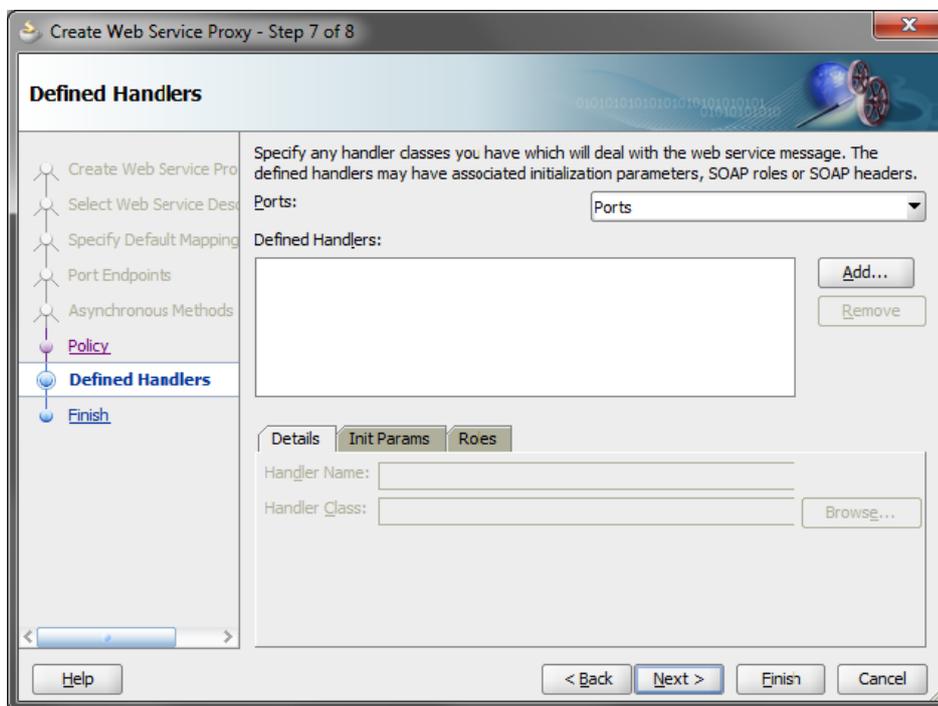
7. Asynchronous Methodsのオプションはデフォルトのままにして「Next」をクリックします。



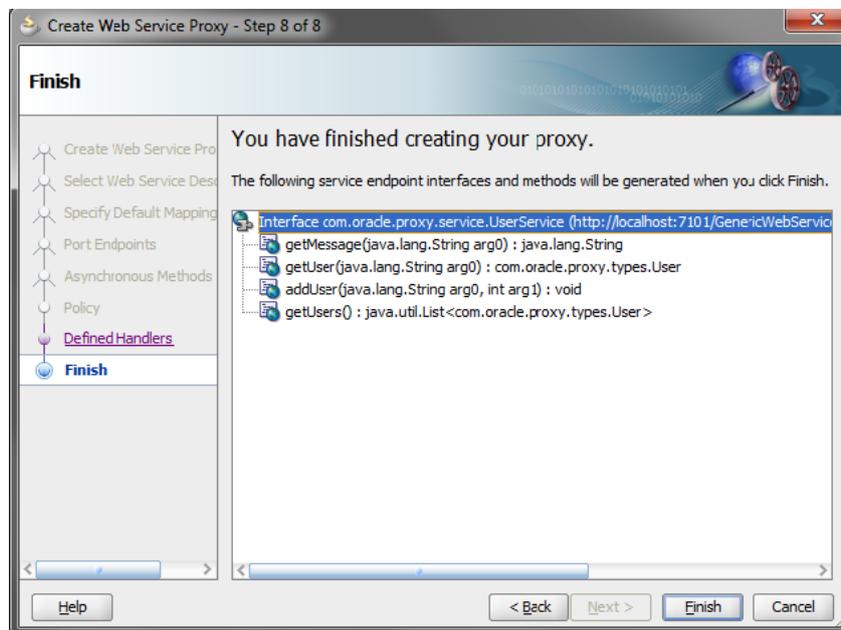
8. デフォルトのまま「Next」をクリックします。



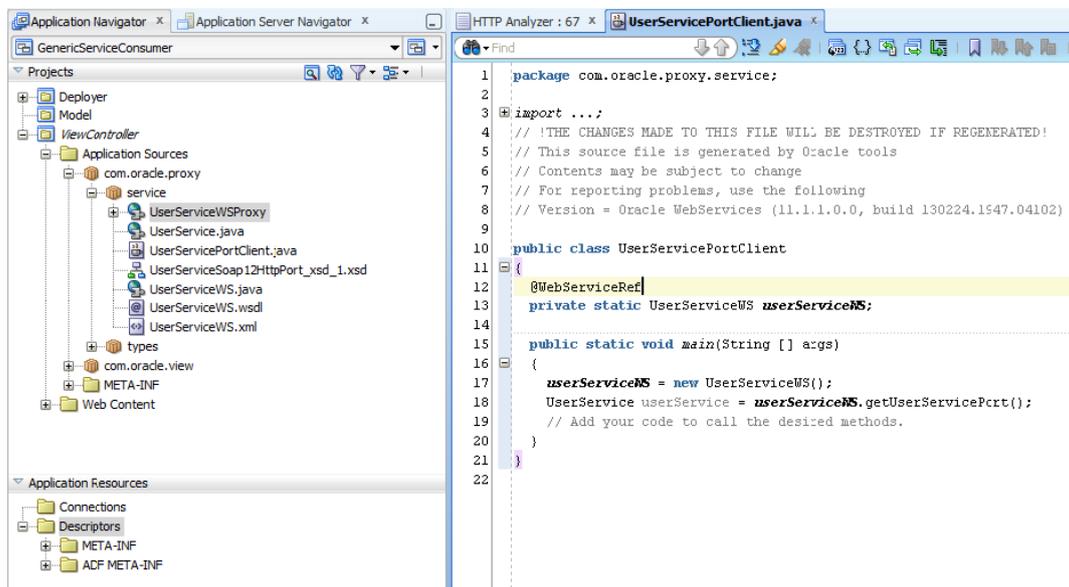
9. デフォルトのまま「Next」をクリックします。



10. Webサービスから生成されるメソッドを確認したら、「Finish」をクリックしてプロキシを作成します。



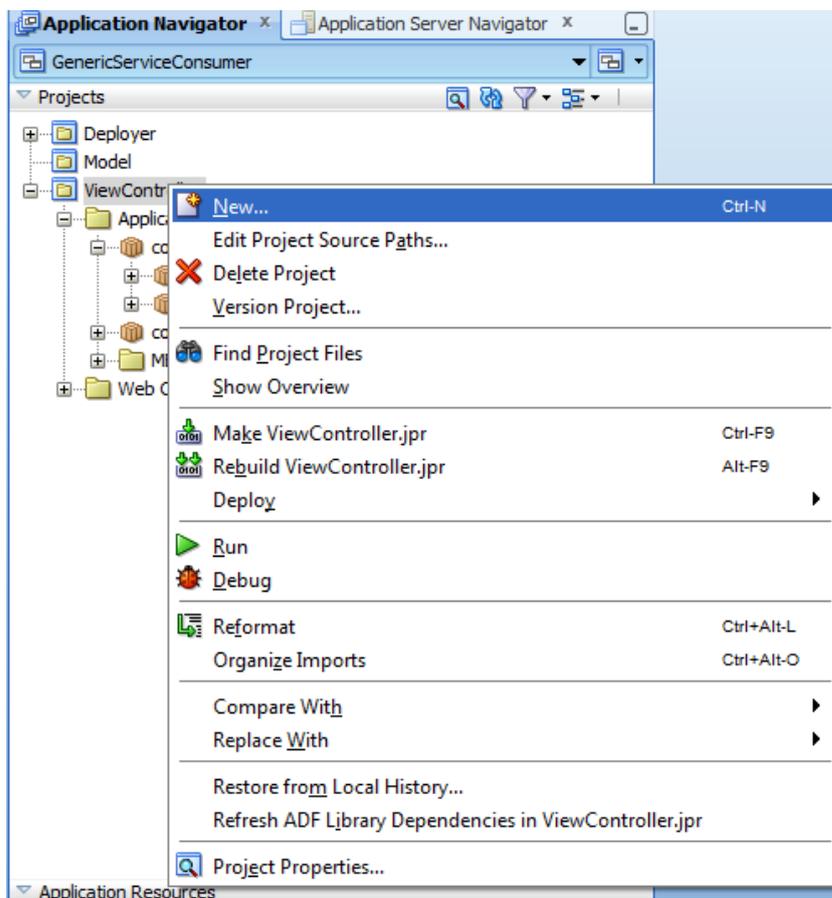
プロキシが正しく構成されている場合、次のような画面が表示されます。



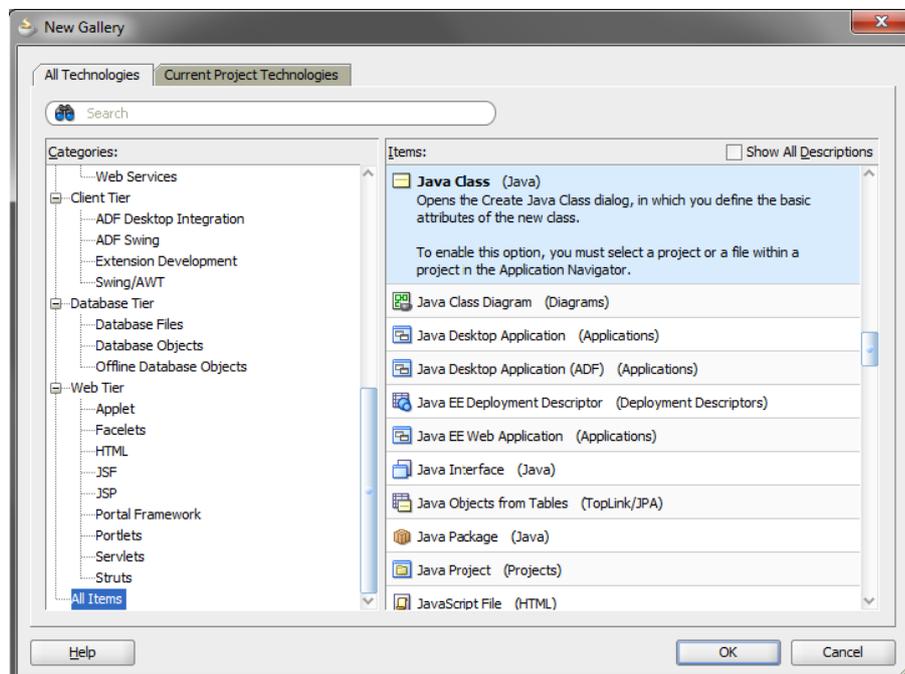
ステップ3a：Webサービス・プロキシ・ファサードの作成

Webサービス・プロキシをデータ・コントロール内で使用するには、プロキシ・ファサードを作成してWebサービス・プロキシからロジックを抽象化する必要があります。これを行うには、次のステップを実行します。

1. JDeveloperで、「GenericServiceConsumer」アプリケーションを開きます。
2. 「ViewController」を右クリックして「New…」を選択します。



3. New Galleryから「Java Class」を選択します。



4. Javaクラスの名前を入力して「OK」をクリックします。

Enter the details of your new class.

Name: ProxyFacade

Package: com.oracle.view

Extends: java.lang.Object

Optional Attributes

Implements:

Access Modifiers

- public
- package protected

Other Modifiers

- <None>
- abstract
- final

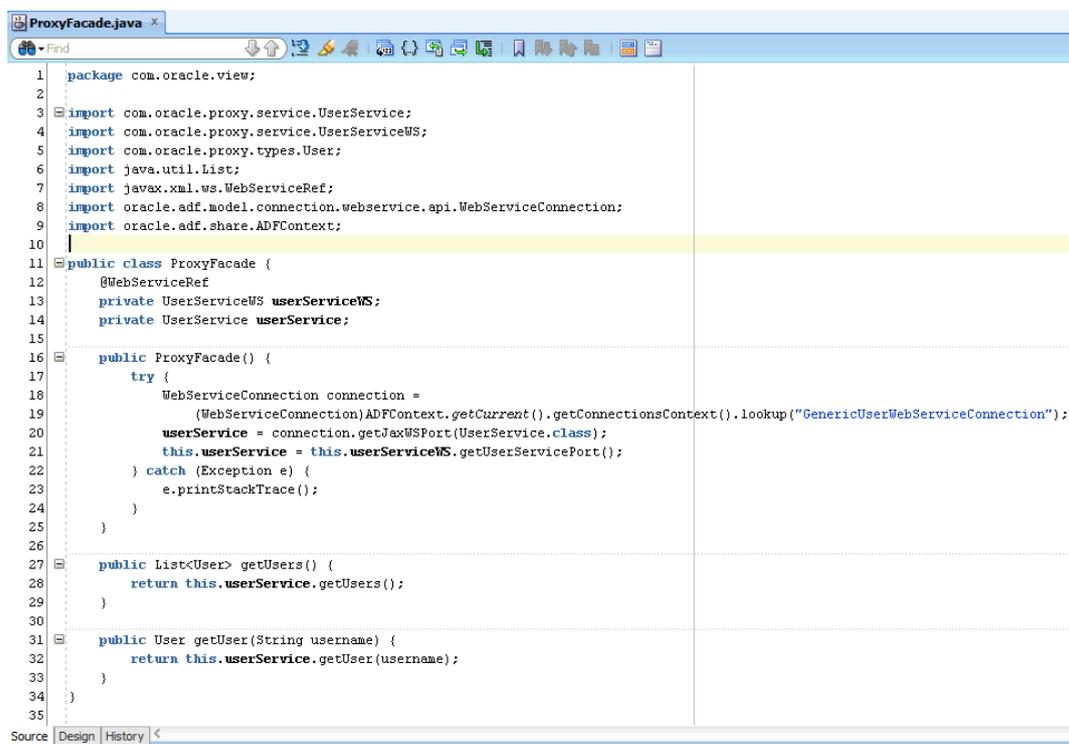
Constructors from Superclass

Implement Abstract Methods

Main Method

Help OK Cancel

5. ProxyFacadeを変更して、Oracle ADF接続構成（後から指定可能）からWebサービス接続を参照できるようにします。lookup検索で 사용되는名前は接続名と同じにする必要があります。



```
1 package com.oracle.view;
2
3 import com.oracle.proxy.service.UserService;
4 import com.oracle.proxy.service.UserServiceWS;
5 import com.oracle.proxy.types.User;
6 import java.util.List;
7 import javax.xml.ws.WebServiceRef;
8 import oracle.adf.model.connection.webservice.api.WebServiceConnection;
9 import oracle.adf.share.ADFContext;
10
11 public class ProxyFacade {
12     @WebServiceRef
13     private UserServiceWS userServiceWS;
14     private UserService userService;
15
16     public ProxyFacade() {
17         try {
18             WebServiceConnection connection =
19                 (WebServiceConnection)ADFContext.getCurrent().getConnectionsContext().lookup("GenericUserWebServiceConnection");
20             userService = connection.getJaxWSPort(UserService.class);
21             this.userService = this.userServiceWS.getUserServicePort();
22         } catch (Exception e) {
23             e.printStackTrace();
24         }
25     }
26
27     public List<User> getUsers() {
28         return this.userService.getUsers();
29     }
30
31     public User getUser(String username) {
32         return this.userService.getUser(username);
33     }
34 }
35
```

ProxyFacadeクラスのソース・コードを次に示します。

```
package com.oracle.view;

import com.oracle.proxy.service.UserService;
import com.oracle.proxy.service.UserServiceWS;
import com.oracle.proxy.types.User;
import java.util.List;
import javax.xml.ws.WebServiceRef;
import oracle.adf.model.connection.webservice.api.WebServiceConnection;
import oracle.adf.share.ADFContext;

public class ProxyFacade {
    @WebServiceRef
    private UserServiceWS userServiceWS;
    private UserService userService;
```

```

public ProxyFacade() {
    try {
        WebServiceConnection connection =

        (WebServiceConnection)ADFContext.getCurrent().getConnectionsContext().look
        up("GenericUserWebServiceConnection");
        userService = connection.getJaxWSPort(UserService.class);
        this.userService = this.userServiceWS.getUserServicePort();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public List<User> getUsers() {
    return this.userService.getUsers();
}

public User getUser(String username) {
    return this.userService.getUser(username);
}
}

```

コンストラクタ内で、現在のコンテキストはconnection.xmlファイル内のWebサービス構成を探します。

この接続は、次のXML構文を使用してローカルのJDeveloperプロジェクトのconnection.xmlファイル内に指定する必要があります。

```

<?xml version = '1.0' encoding = 'UTF-8'?>
<References xmlns="http://xmlns.oracle.com/adf/jndi">
  <Reference name="GenericUserWebServiceConnection"
  className="oracle.adf.model.connection.webservice.impl.WebServiceConnectio
  nImpl" manageInOracleEnterpriseManager="true" deployable="true" xmlns="">
    <Factory
    className="oracle.adf.model.connection.webservice.api.WebServiceConnection
    Factory"/>
    <RefAddresses>
      <XmlRefAddr addrType="WebServiceConnection">
        <Contents>
          <wsconnection description="http://localhost:8888/GenericWebService-
          UserWebService-context-root/UserServiceSoap12HttpPort?WSDL"
          service="{http://oracle.com/}UserServicePort"/>
        </Contents>
      </RefAddr>
    </RefAddresses>
  </Reference>
</References>

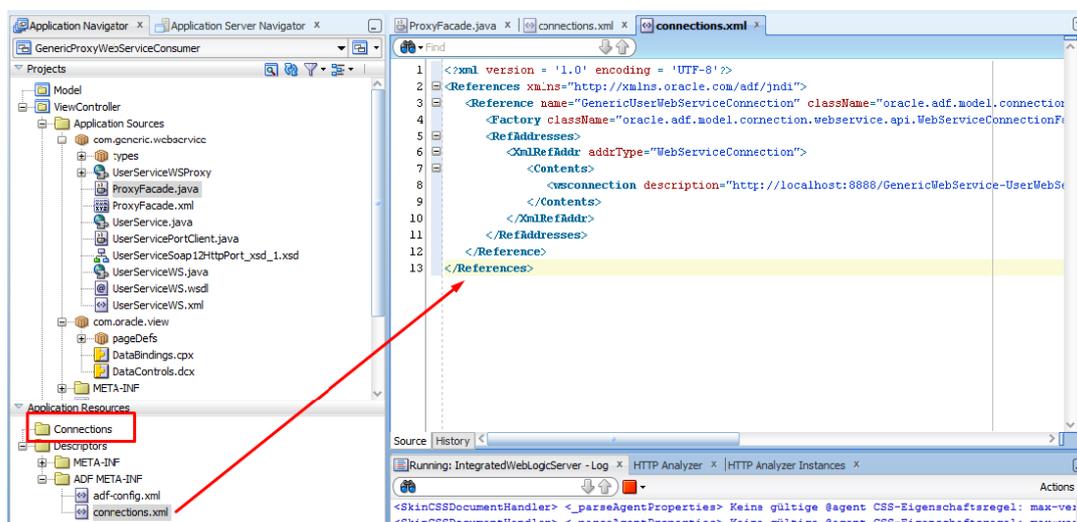
```

```

</XmlRefAddr>
</RefAddresses>
</Reference>
</References>

```

この抜粋箇所では、WebCenter Portalから使用されるWebサービス構成が示されている点に注意してください。この接続がローカルJDeveloperプロジェクトのconnection.xmlファイルに含まれていても、この接続を手動で追加しない限り、下に示すスクリーンショットのConnectionsセクションには表示されません。



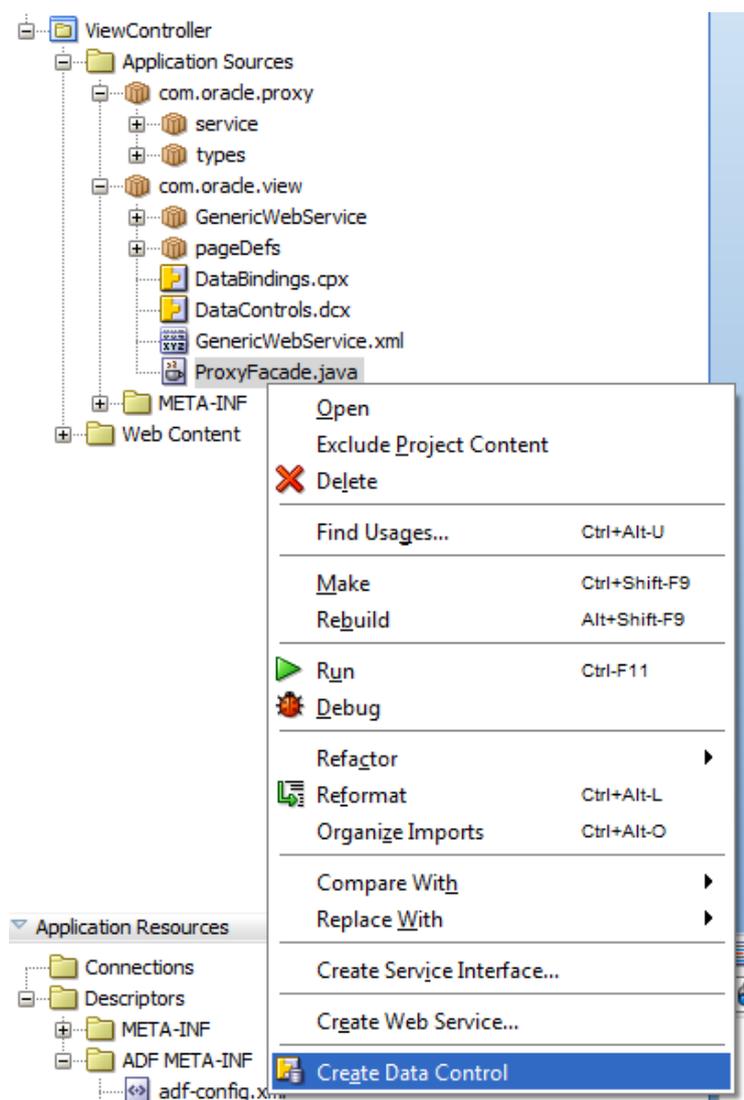
接続を追加するには、「Consumer」アプリケーションを開きます。Application Resourcesの「Descriptions」→「ADF META-INF」から「connections.xml」ファイルを開き、提供済みのXML参照スニペットを追加します（上のスクリーンショットを参照）。

注：各自の開発マシン上でアプリケーションのローカル・テストを行うには、この接続構成の追加が非常に重要です。

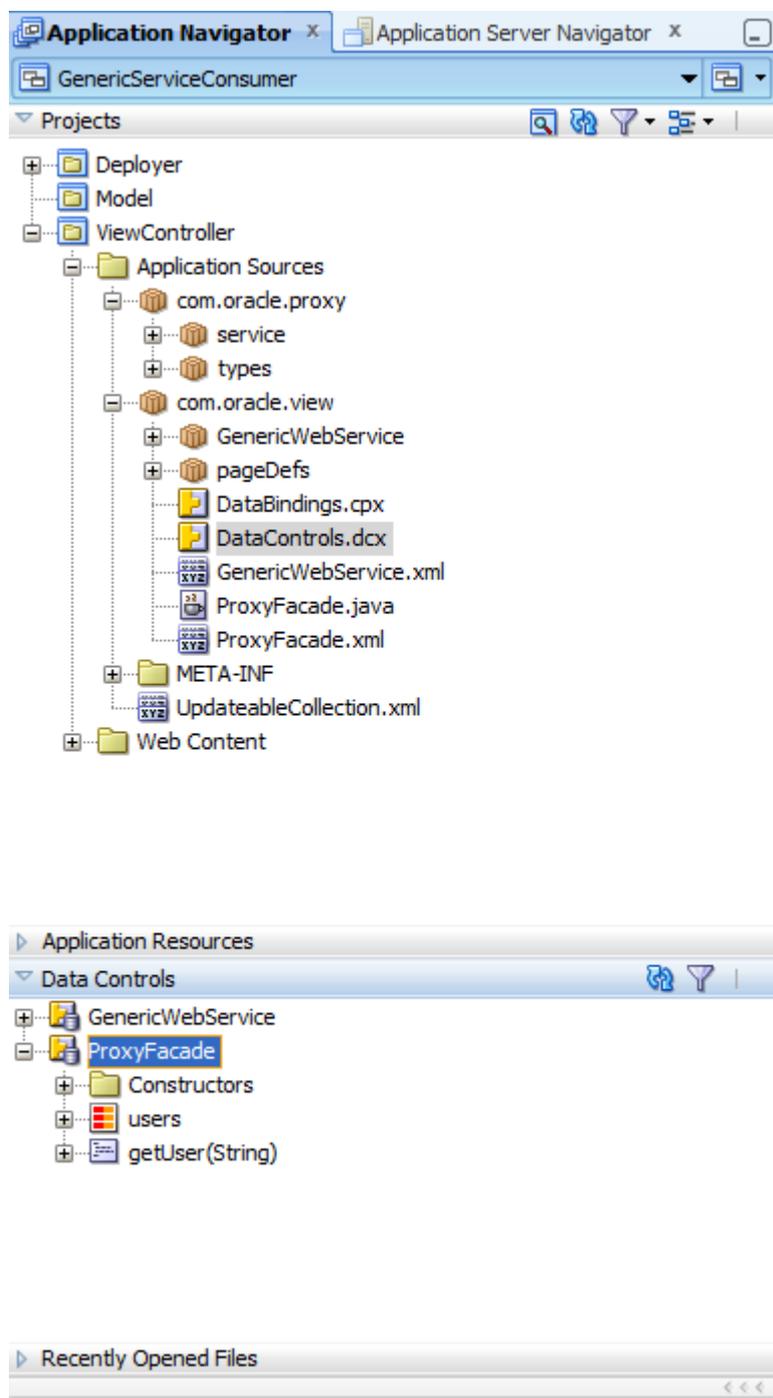
ステップ3b：プロキシ・ファサードを使用したデータ・コントロールの作成 (GenericServiceConsumer)

このステップでは、ステップ4で作成したプロキシ・ファサードを使用してデータ・コントロールを作成します。

1. JDeveloperで「GenericServiceConsumer」プロジェクトを表示し、「ProxyFacade」クラスを右クリックして「Create Data Control」を選択します。



2. JDeveloperが構成を準備するまで待ちます。数秒後にデータ・コントローラ が作成されます。



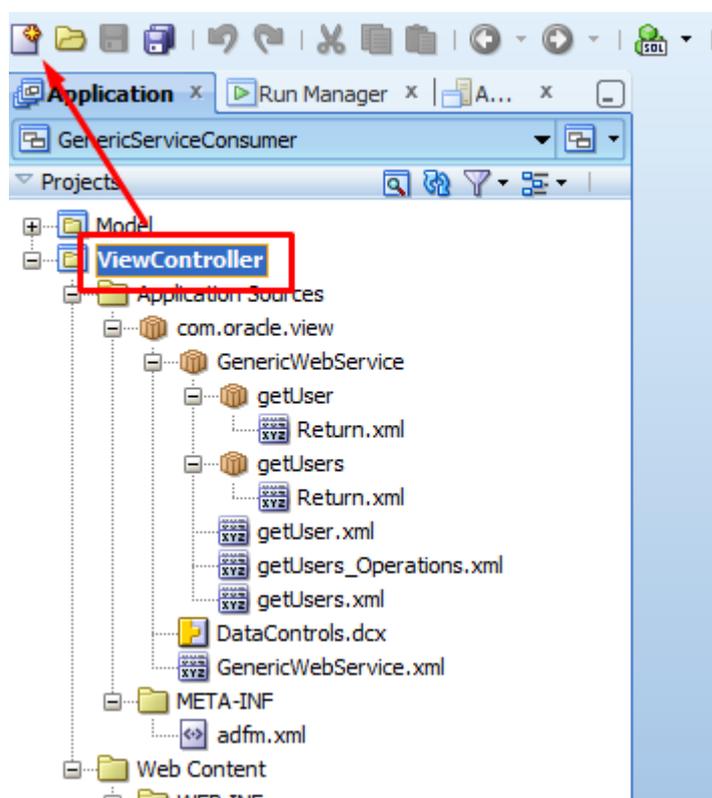
新しいデータ・コントロールはOracle ADFページまたはタスク・フロー内で使用できます。

ステップ4：バインド・タスク・フローの作成

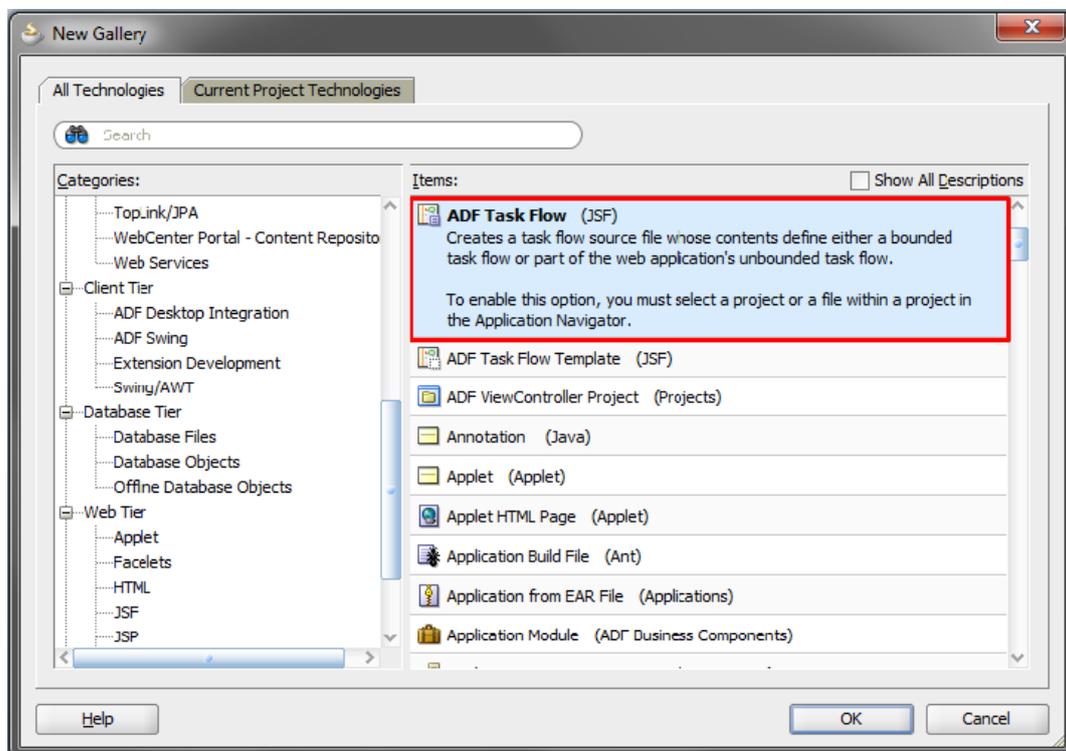
このステップでは、バインド・タスク・フローを作成します。このタスク・フローは後からポータル・サーバーにデプロイし、WebCenter Portalで使用します。

注：この作業はViewControllerプロジェクト内で実施することが重要です。後続のステップで、ポータル・サーバーへのデプロイ準備が整ったOracle ADF JARライブラリに対して、ViewControllerプロジェクトからコンテンツをエクスポートします（または、別のプロジェクトにビジネス・ロジックを作成してViewControllerプロジェクトにインポートすることもできます）。

1. 「ViewController」プロジェクトを選択し、「New」アイコンをクリックします。

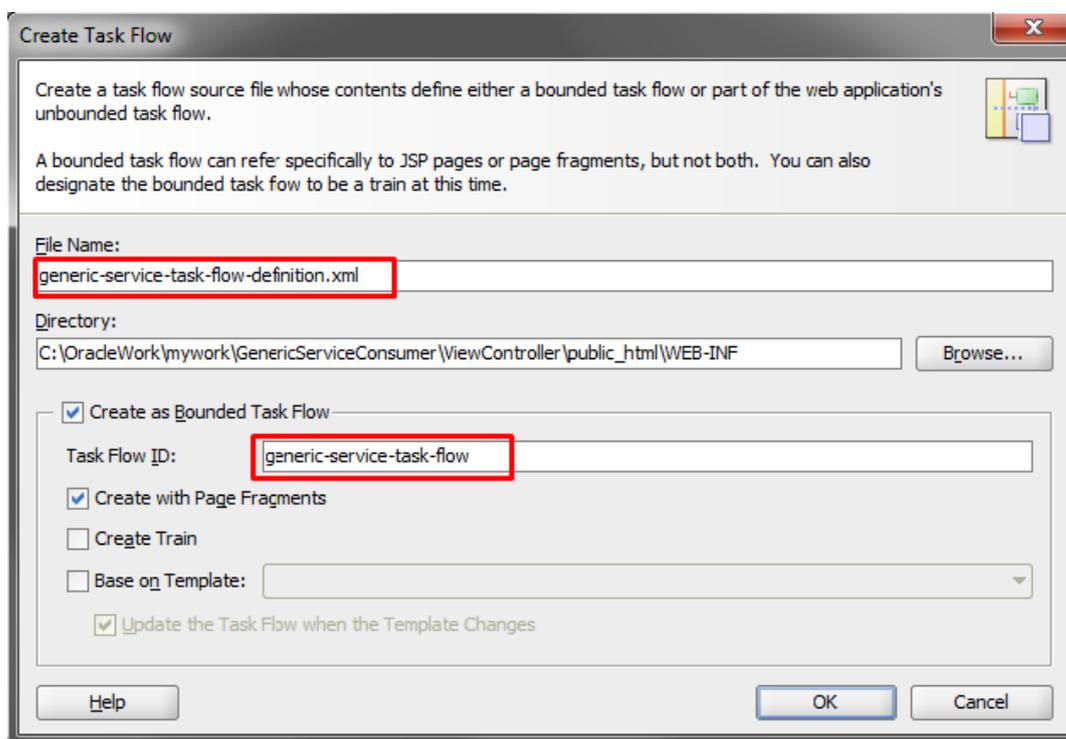


2. New Galleryで、「ADF Task Flow (JSF)」を選択して「OK」をクリックします。

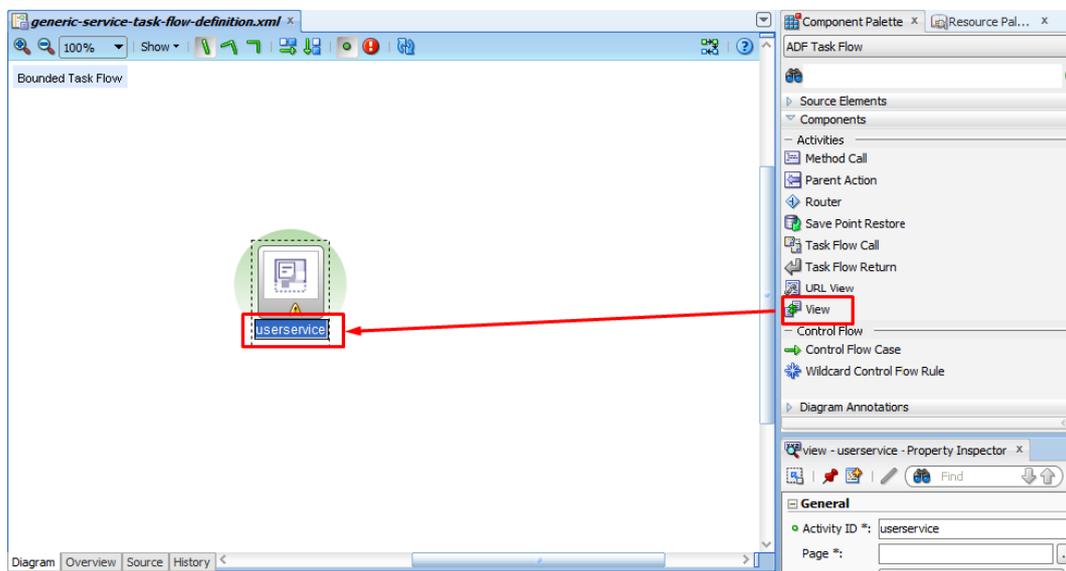


3. File Nameフィールドにファイル名を入力します。
4. File Nameを入力すると、Task Flow IDが自動的に変更されます。

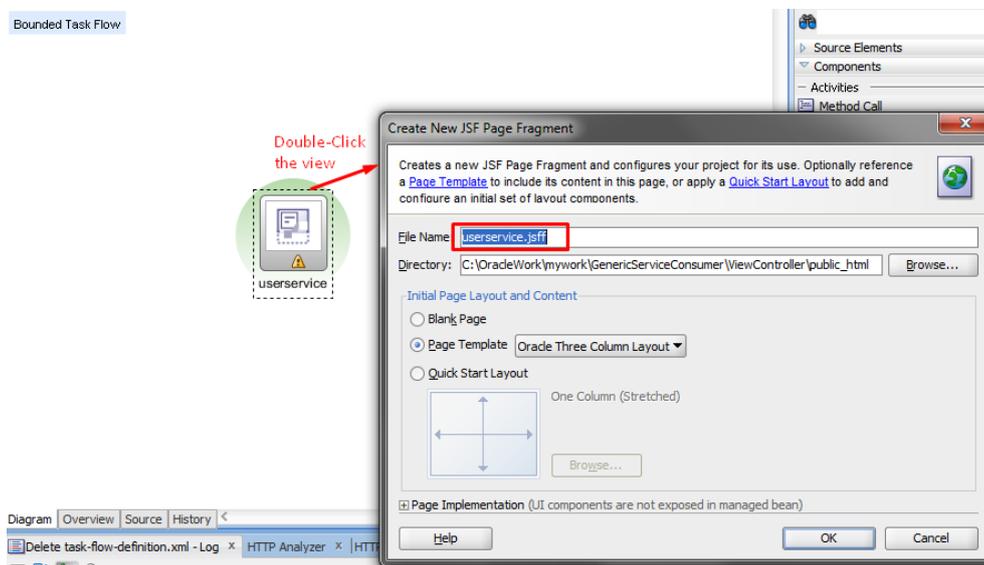
重要：名前は慎重に選択してください。ここで指定したTask Flow ID (generic-service-task-flow) は、WebCenter Portalのリソース・カタログからタスク・フローを識別するために使用されます。



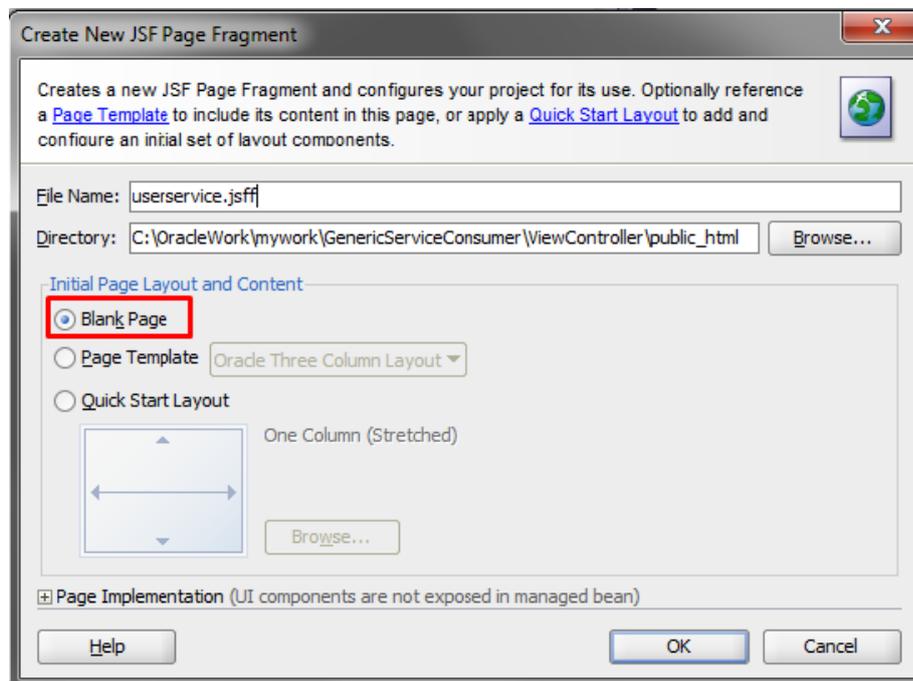
5. 右側のパレットから「View」を選択し、タスク・フローの定義ペイン上にドラッグ・アンド・ドロップします。ビュー名としてuserserviceを入力します。



- ビューをダブルクリックして新しいフラグメントを作成します。File Nameフィールドに `userservice.jsff` と入力します。



- ページ・テンプレートとして「Blank Page」を選択し、「OK」をクリックします。

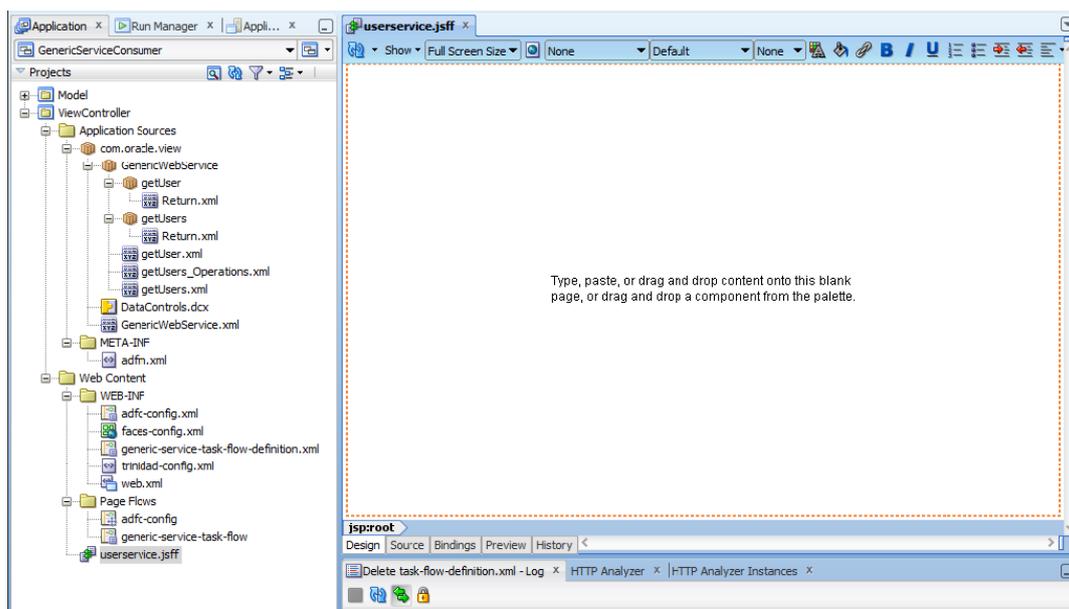


- すべての変更内容を保存します。

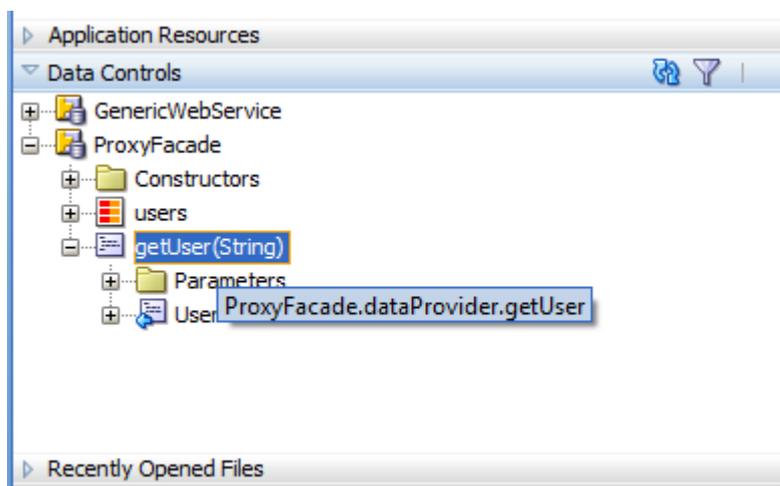
ステップ5：タスク・フロー内でのデータ・コントロールの消費

このステップでは、先ほど作成したデータ・コントロールをタスク・フロー内で消費します。

1. 「userservice.jsff」 ページ・フラグメントを開きます。

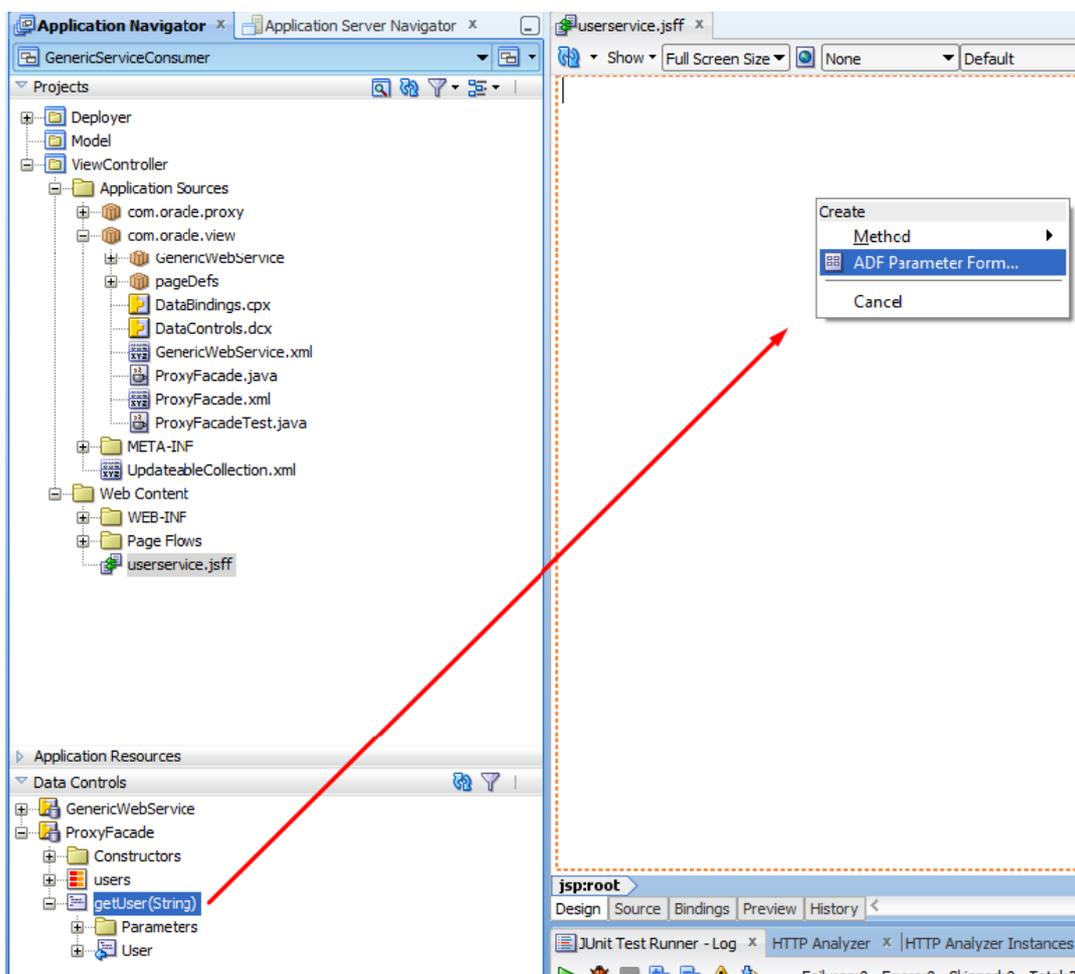


2. 「Data Controls」 タブを開き、「ProxyFacade」 → 「getUser(String)」 を選択します。



注：はじめにテストとして、`getUser`サービスのコールのみに基づいたシンプルな例を作成します。サービスの残りの部分は後で使用します。

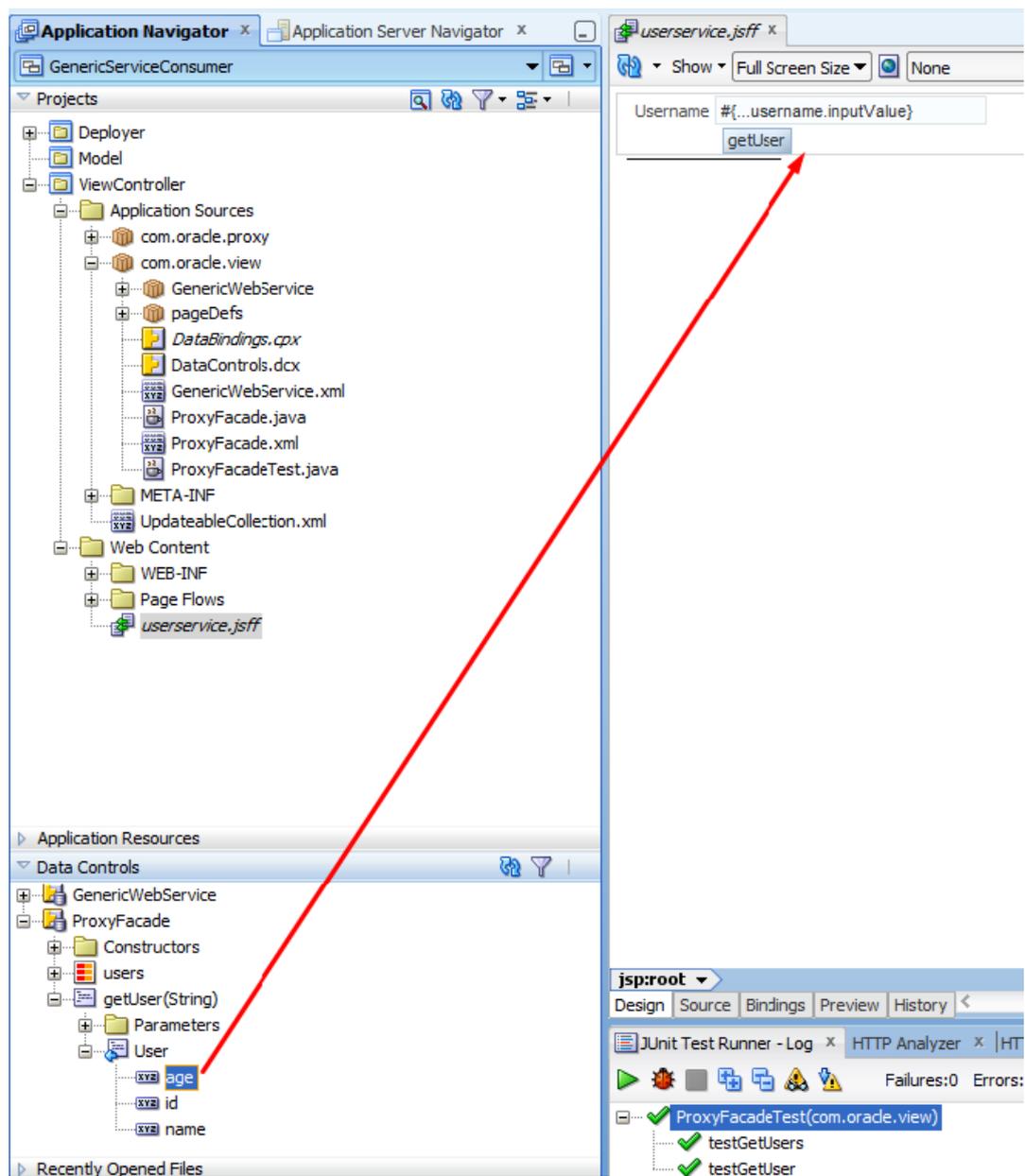
3. Data Controlsナビゲータから「getUser」をドラッグしてタスク・フロー・ページ・フラグメントにドロップし、ドロップダウン・リストから「ADF Parameter Form」を選択します。



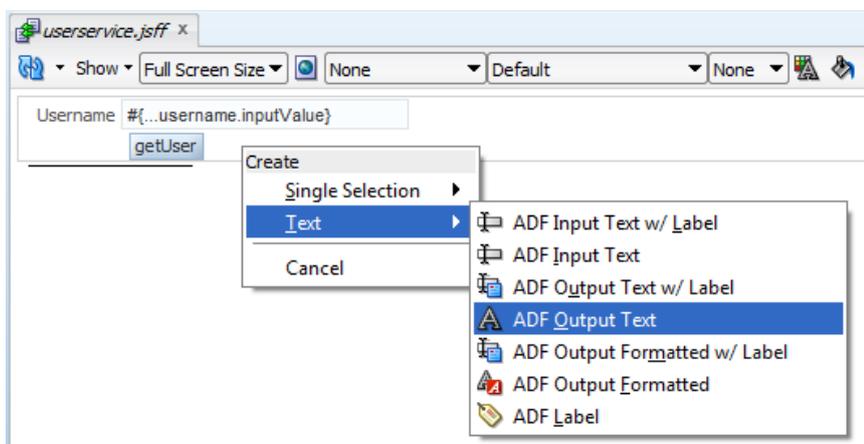
ADF Parameter Formオプションは、ユーザー名に基づいてユーザー情報を返すフォームを作成します。



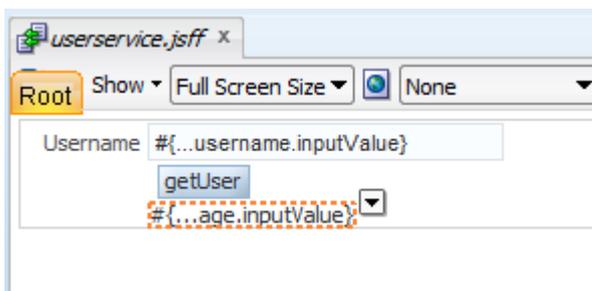
- 関数getUserから「age」（Returnパラメータの下）をドラッグし、ページ上にドロップします。ageパラメータは、返される情報の中にユーザーの年齢を表示します。



- ドロップダウン・メニューから好みの出力オプションを選択します。



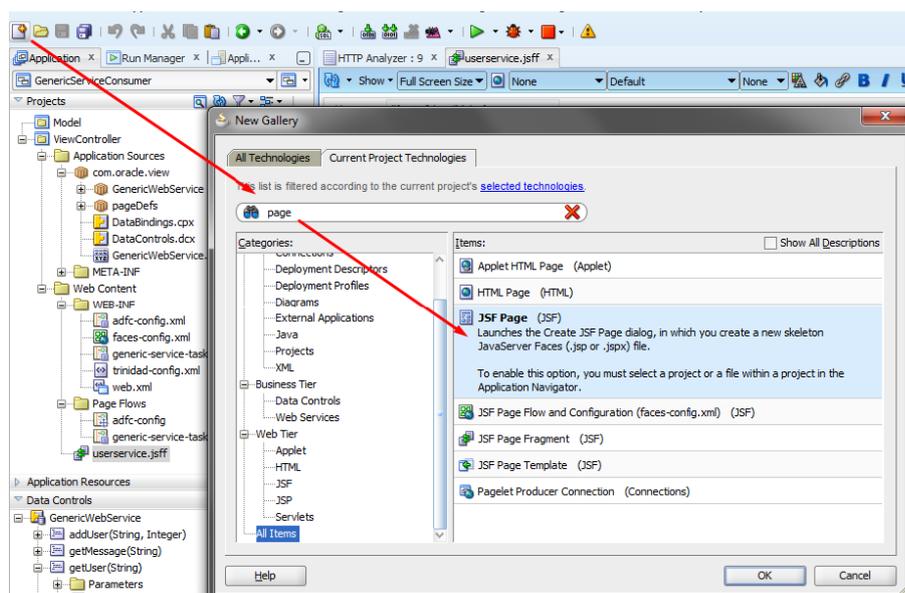
ページは次のようになります。



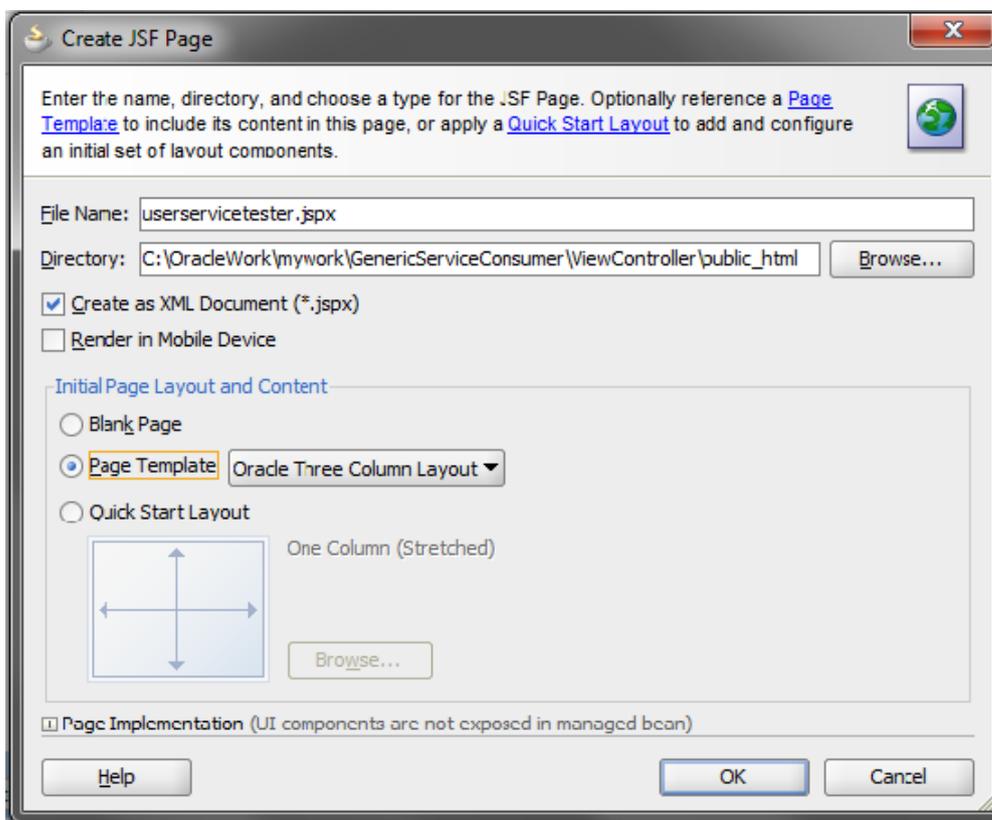
ステップ6：サンプル・タスク・フローのテスト用ページの作成

タスク・フローをテストするには、ページを作成して、このページのバインディングとしてタスク・フローをドラッグ・アンド・ドロップします。

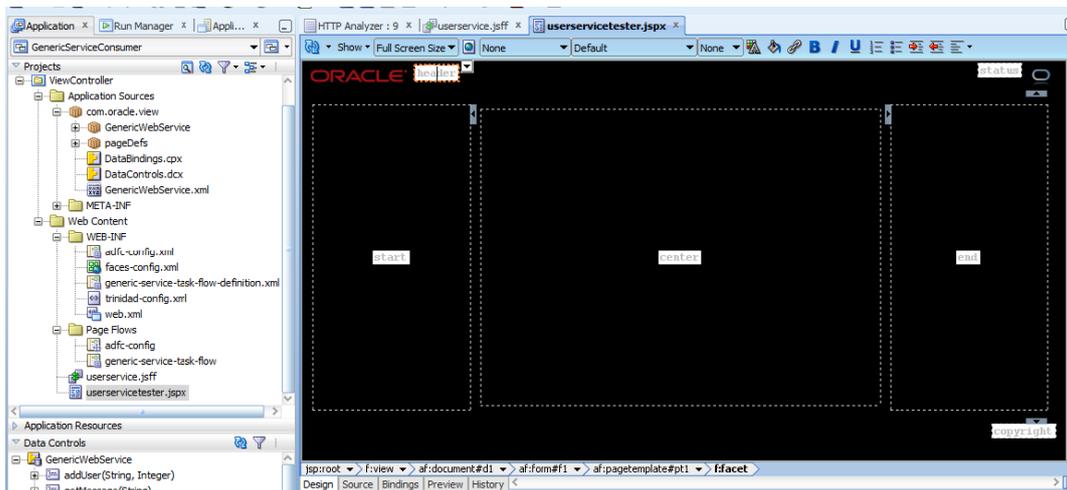
1. JDeveloperのGenericServiceConsumerアプリケーションで「New」アイコンをクリックし、「JSF Page」を選択して新規ページを作成します。



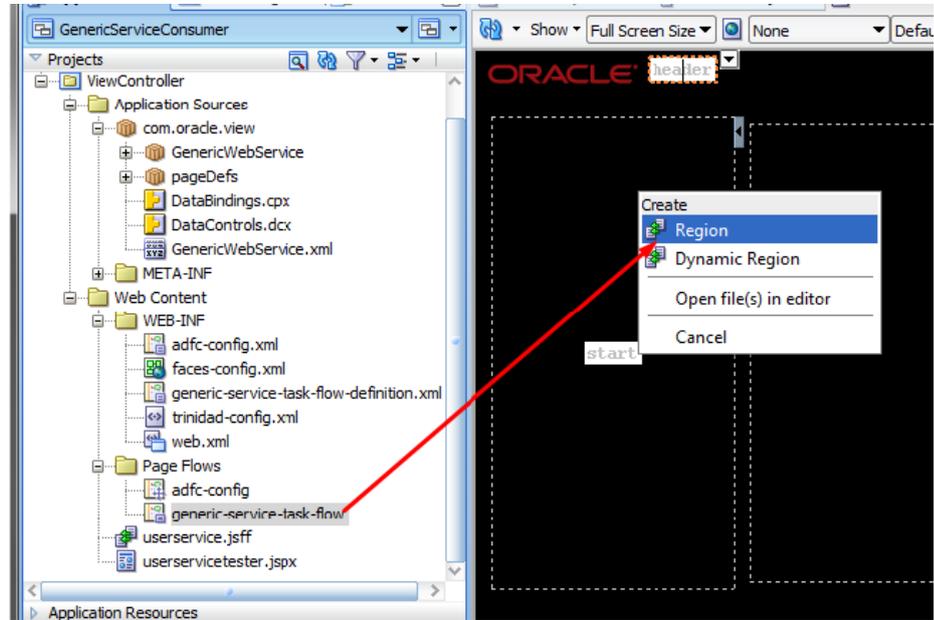
2. 新しいページの名前（例：userservicetester.jspax）を入力し、好みのテンプレートを選択します。このページはタスク・フローのテストのみで使用するため、空白のテンプレートを使用しても構いません。



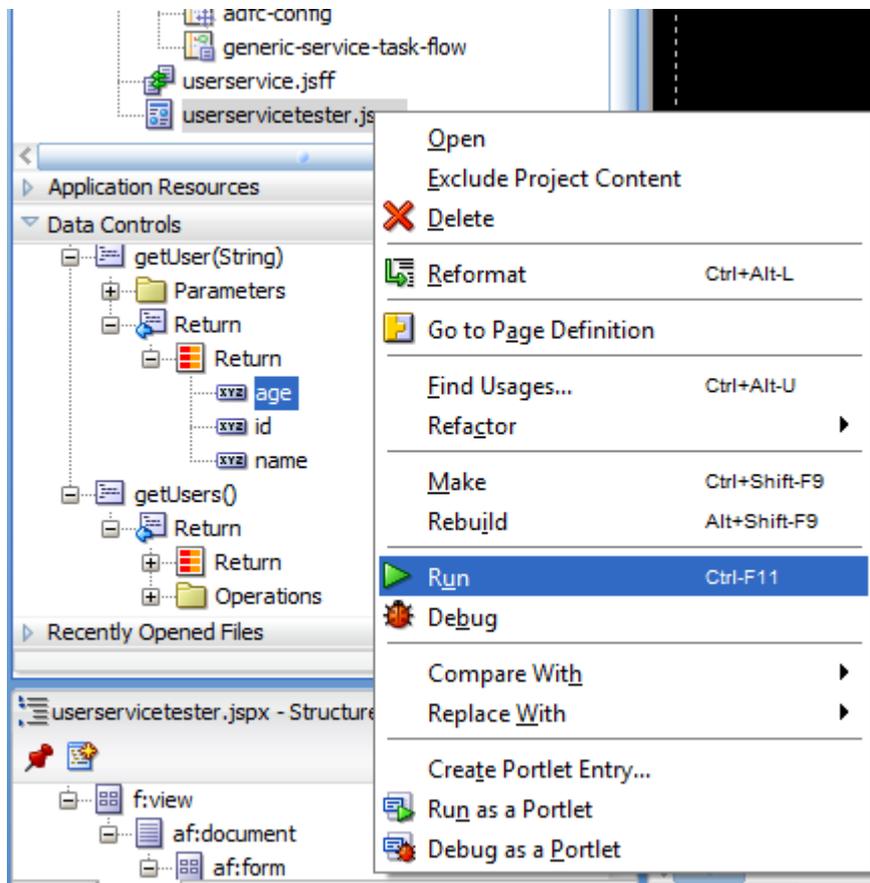
ページが作成されると、ページは次のようになります。



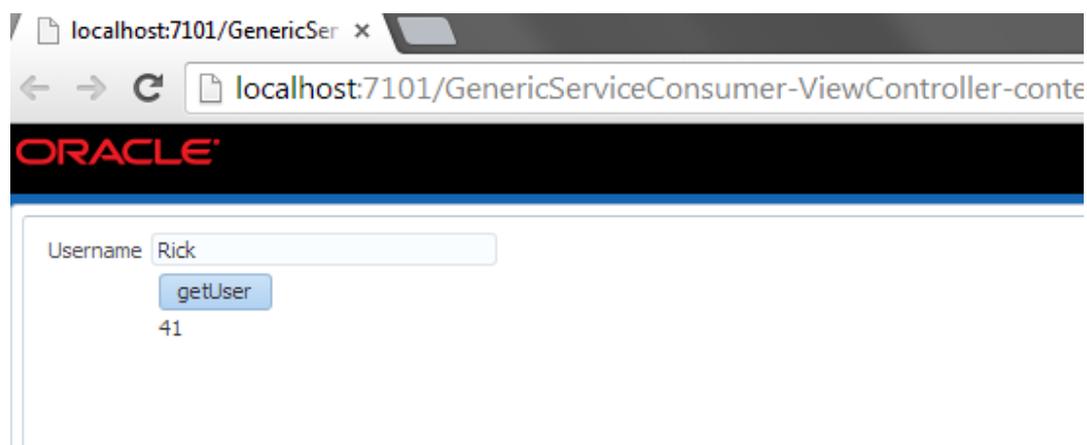
3. タスク・フローをページ上にドラッグ・アンド・ドロップします（下図を参照）。ドロップダウン・メニューから「Region」を選択します。



4. ページを実行してタスク・フローをテストします。



エラー・メッセージが表示されることなくページが起動され、デフォルト・ブラウザに表示されます（下図を参照）。



5. **Username**フィールドにユーザー名を入力すると、ユーザーの年齢が表示されます。デフォルトのWebサービスには次のユーザー名が登録されています。

Fill：年齢42、**Peter**：年齢32、**Rick**：年齢41

3つのユーザーすべてをテストし、Webサービスとタスク・フローが正しく動作することを確認します。

ステップ7：Oracle ADFライブラリのJARファイルへのサンプル・タスク・フローのデプロイ

このステップでは、WebCenter Portalで使用できるOracle ADFライブラリ (JARファイル) に対して、Webサービス・プロキシとデータ・コントロールを含むサンプル・タスク・フローをデプロイします。

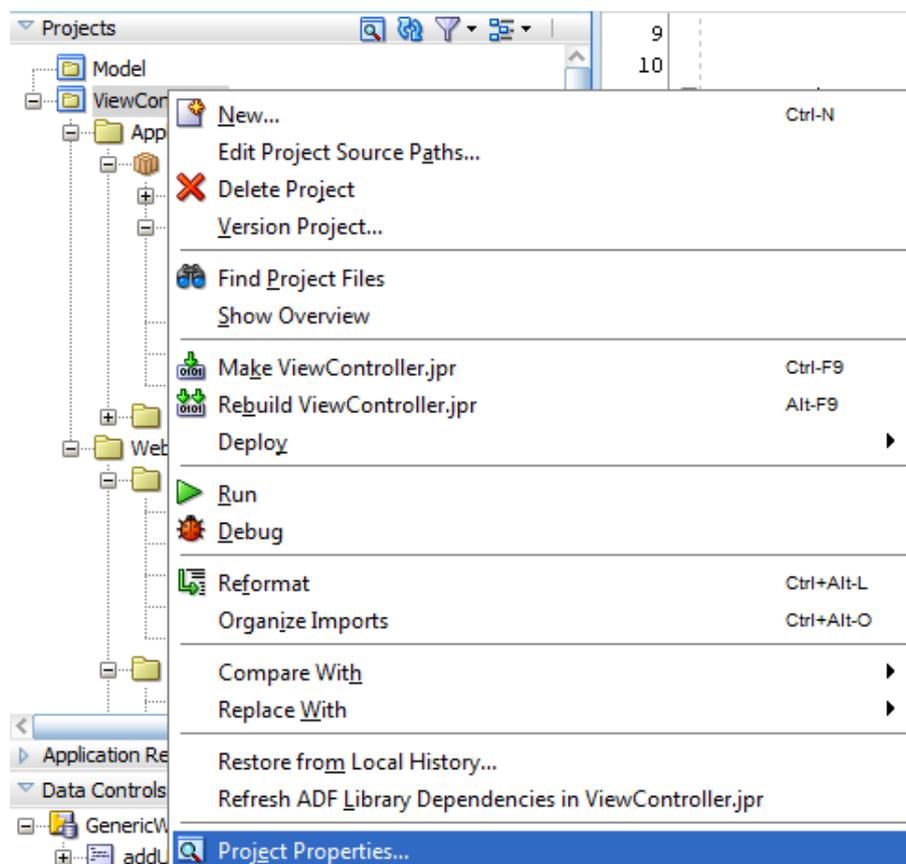
注：タスク・フローのデプロイには次の2つの段階があります。

- Oracle ADFライブラリのJARファイルへのタスク・フロー・プロジェクトのデプロイ
- Oracle ADFライブラリのJARファイルをターゲット・サーバーにデプロイするための、別のデプロイメント・プロファイルの作成

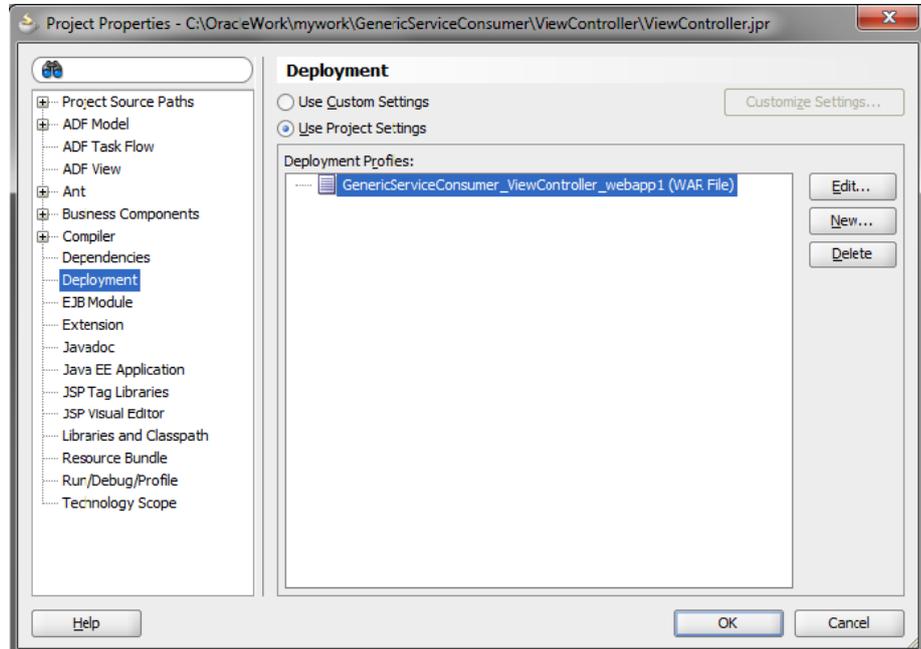
このステップでは、最初の段階 (JARファイルへのデプロイ) について説明します。

JARファイルへのGenericServiceConsumerタスク・フロー・プロジェクトのデプロイ

1. ViewControllerプロジェクトをOracle ADFライブラリのJARファイルとしてエクスポートするには、新しいデプロイメント・プロファイルを作成する必要があります。「ViewController」プロジェクトを右クリックし、「Project Properties」を選択します。

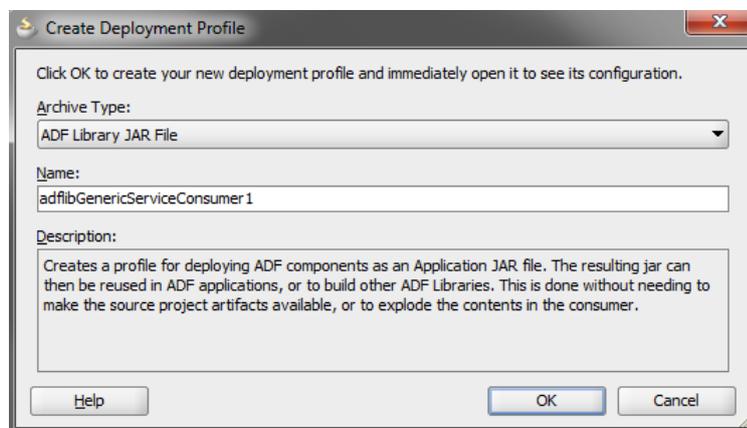
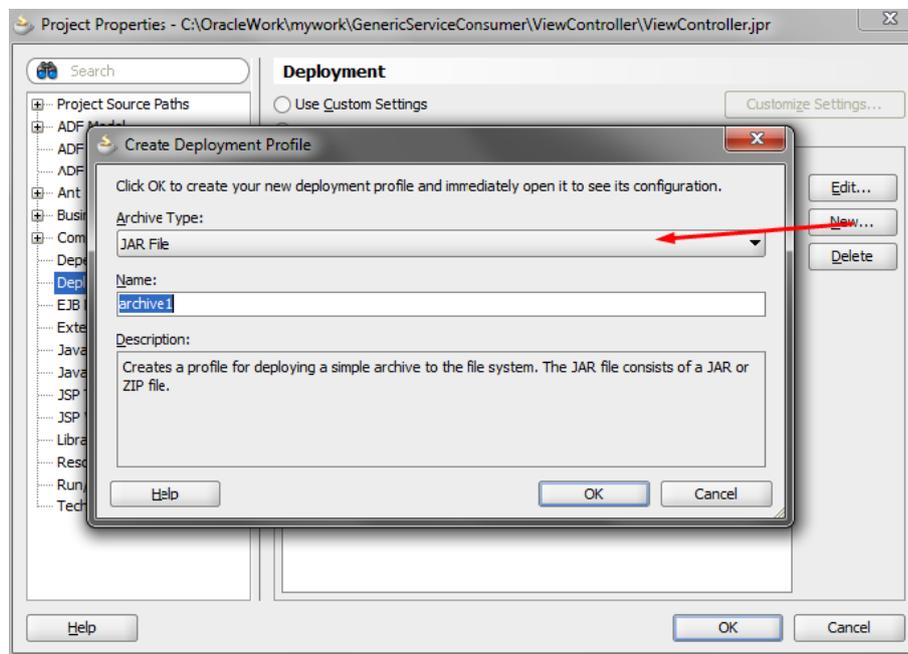


2. 新しいウィンドウで「Deployment」を選択します。

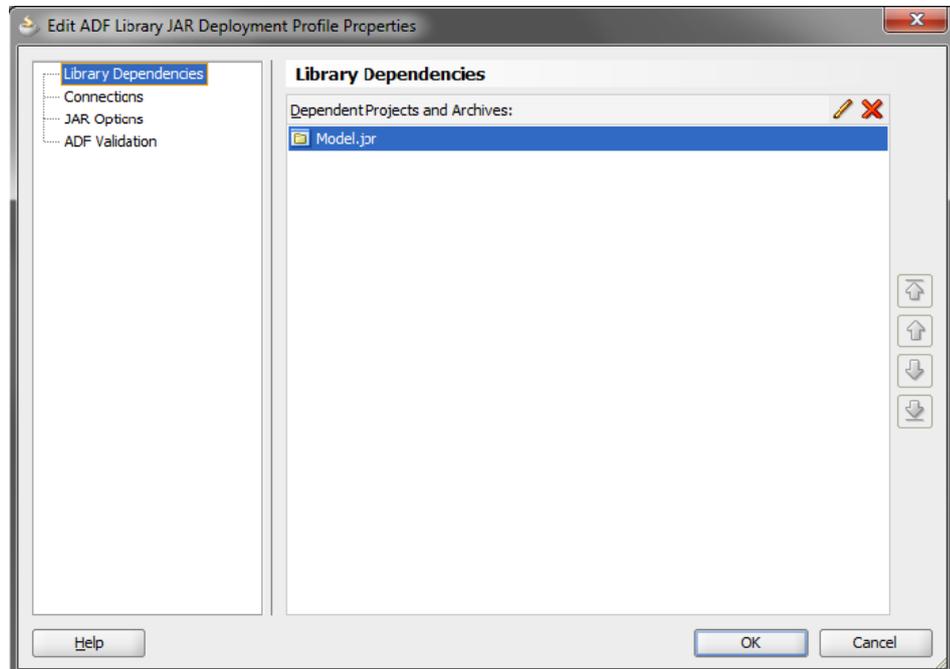


3. 右側のパネルから「New」を選択して新しいWARプロファイルを作成します。

4. Archive Typeの下から「ADF Library JAR File」を選択し、ファイル名（例：adflibGenericServiceConsumer1）を入力します。「OK」をクリックします。

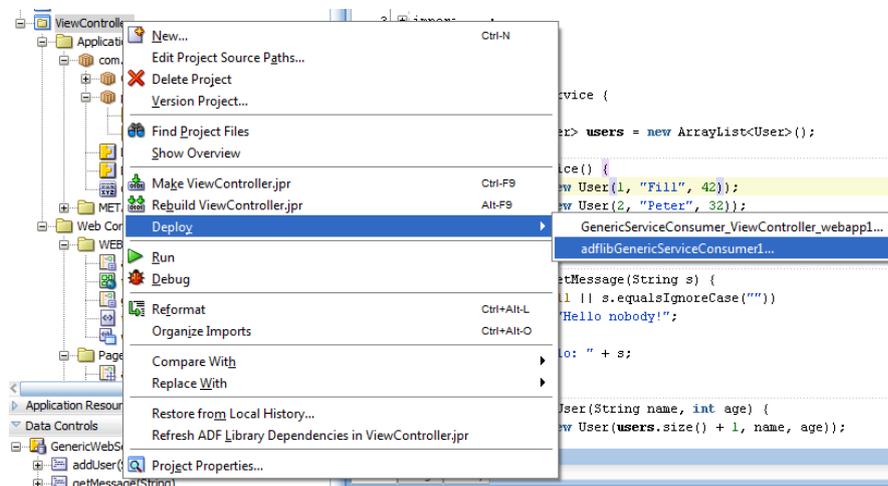


5. 次の画面でデフォルト設定のままにして「OK」をクリックし、新しいデプロイメント・プロファイルを保存します。このプロファイルは次のステップで使用します。

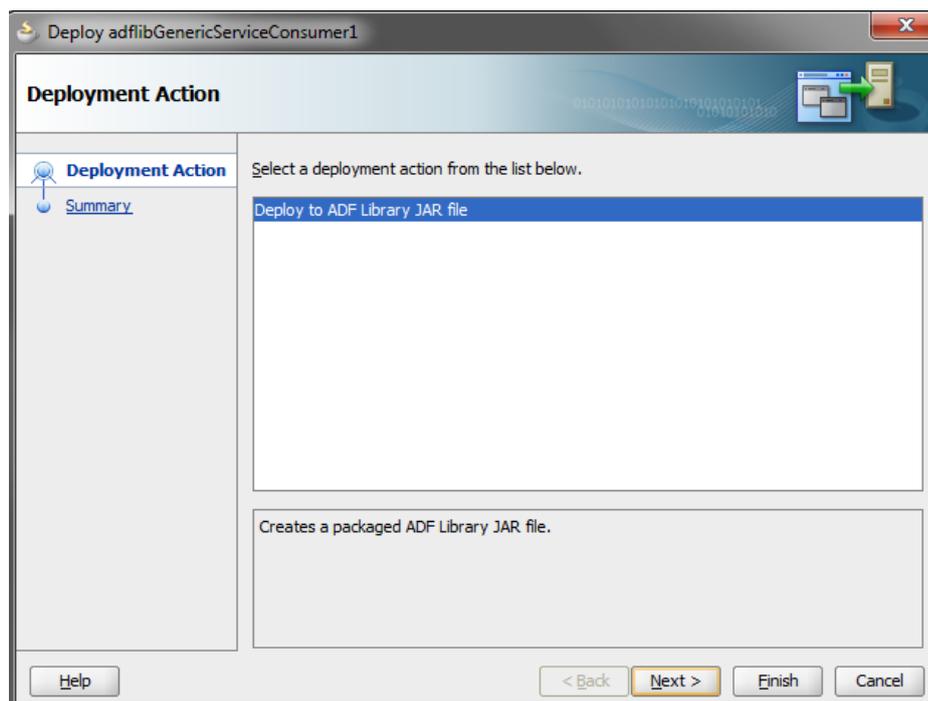


6. プロファイルをデプロイして、Oracle ADFライブラリのJARファイルを作成します。「ViewController」プロジェクトを右クリックし、「Deployment」→「<デプロイメント・プロファイル名>」を選択します。

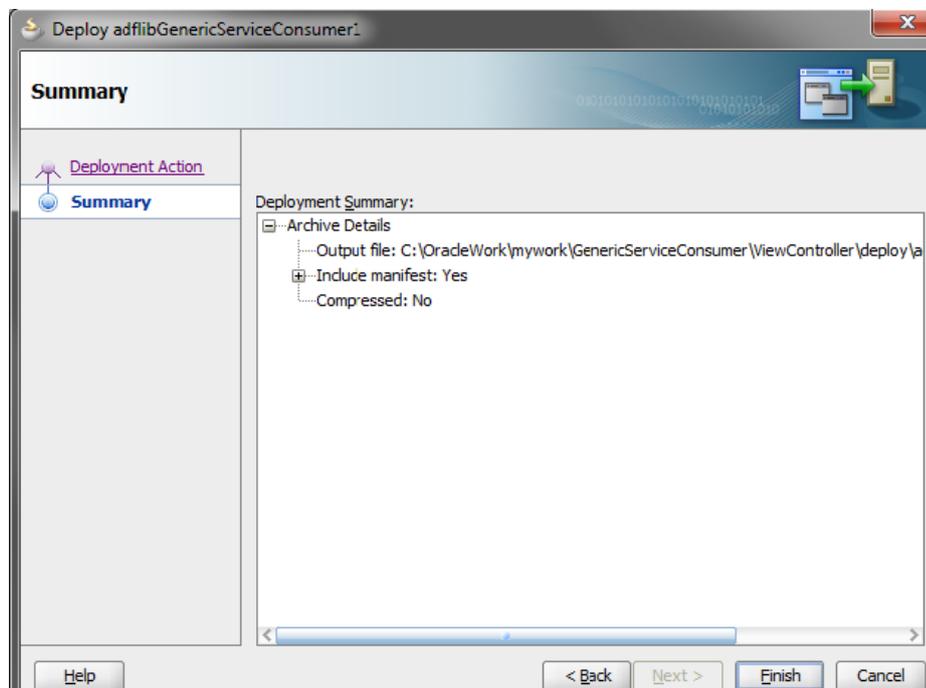
これにより、ローカルのJDeveloperプロジェクト内にJARファイルが作成されます。



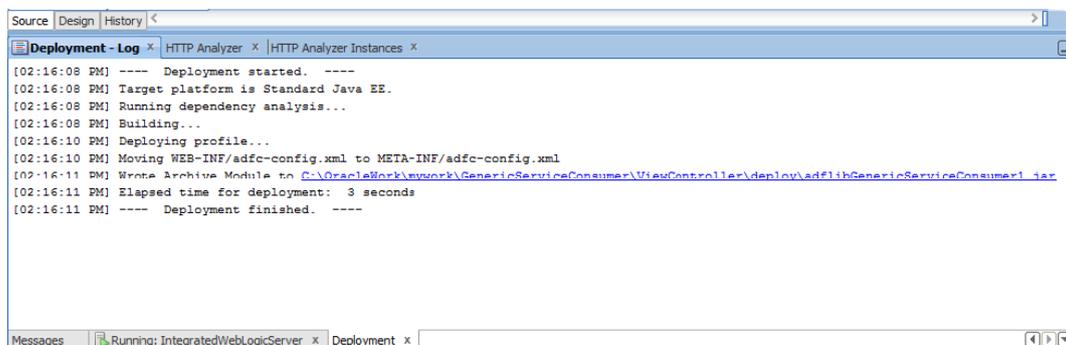
7. Deployment Action画面で、「Deploy to ADF Library JAR file」を選択して「Next」をクリックします。



8. サマリー情報を確認したら、「Finish」をクリックしてJARファイルを作成します。



デプロイメント・プロファイルが正しく実行されると、ログ内にOracle ADFライブラリのJARファイルを含むフォルダが表示されます。

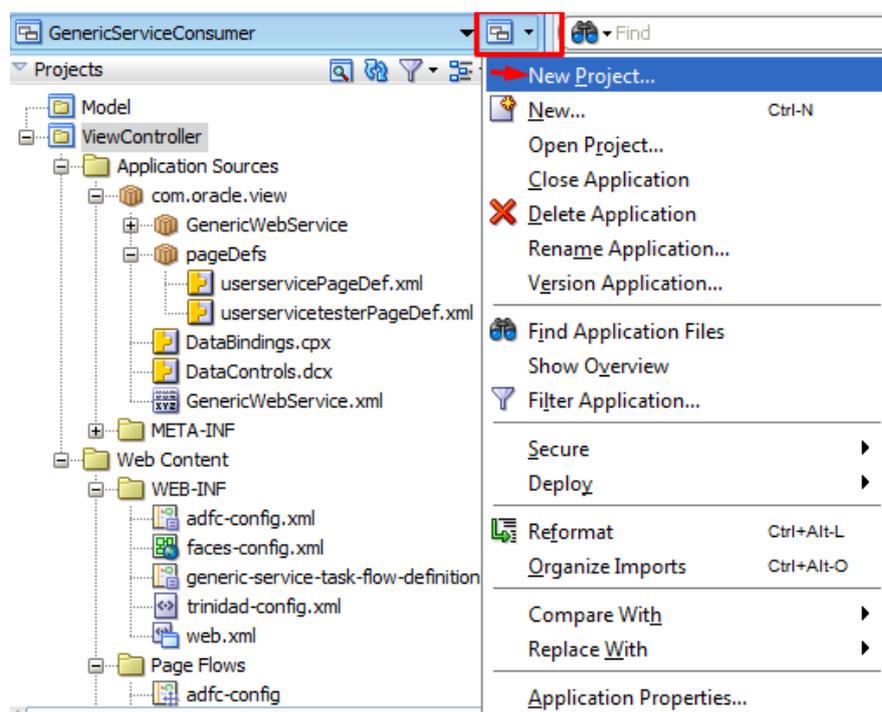


次のステップでは、Deployerプロジェクトを使用し、WebCenter Portalの管理対象サーバーに対して、新しく作成したOracle ADFライブラリのJARファイル（タスク・フローを含む）をデプロイします。

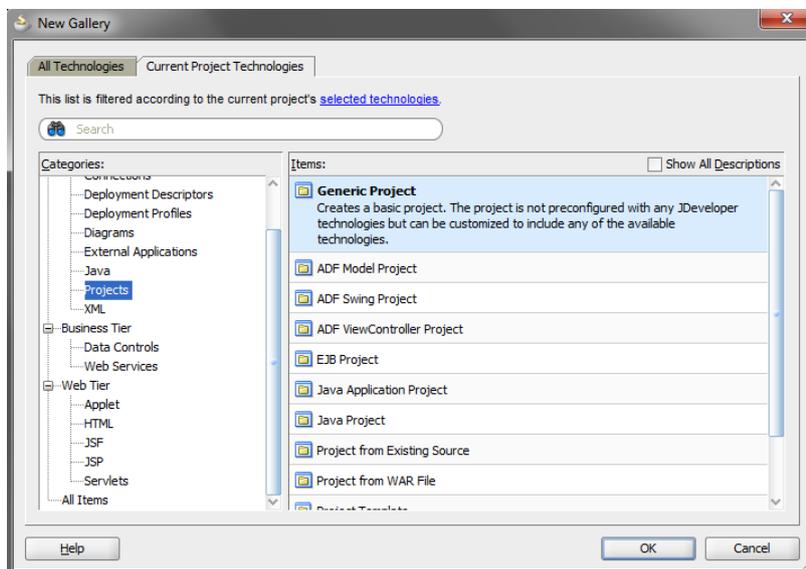
ステップ8：Deployerプロジェクトの作成

このステップでは、1つまたは複数のタスク・フロー・プロジェクトをデプロイできる汎用プロジェクトを作成します。このプロジェクトは、Oracle ADFライブラリ（JARファイル）をWebCenter Portalの管理対象サーバーにデプロイするためだけに使用します。

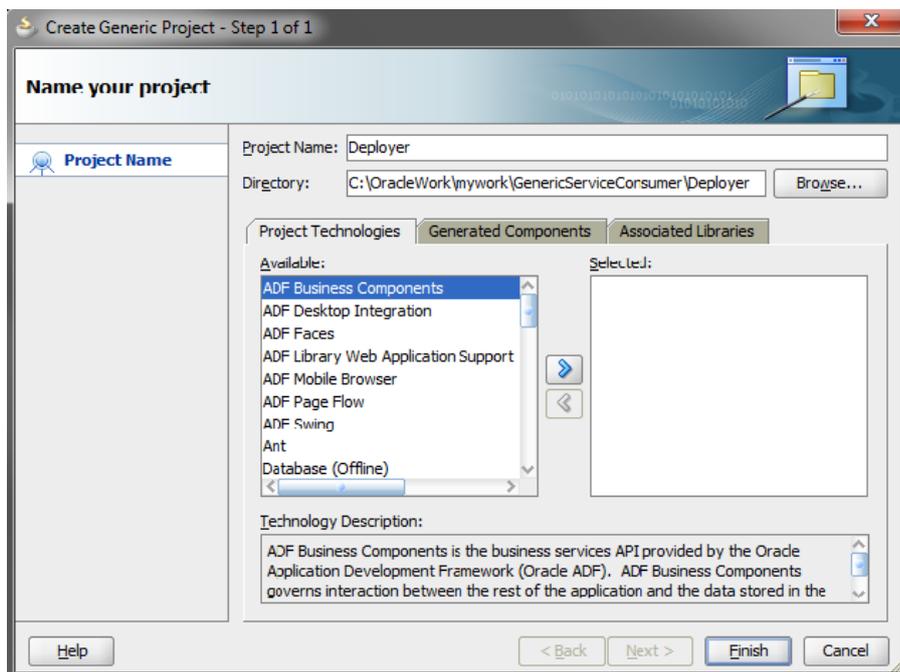
1. GenericServiceConsumerアプリケーション内にDeployerプロジェクトを作成するには、「New...」→「New Project...」を選択します。



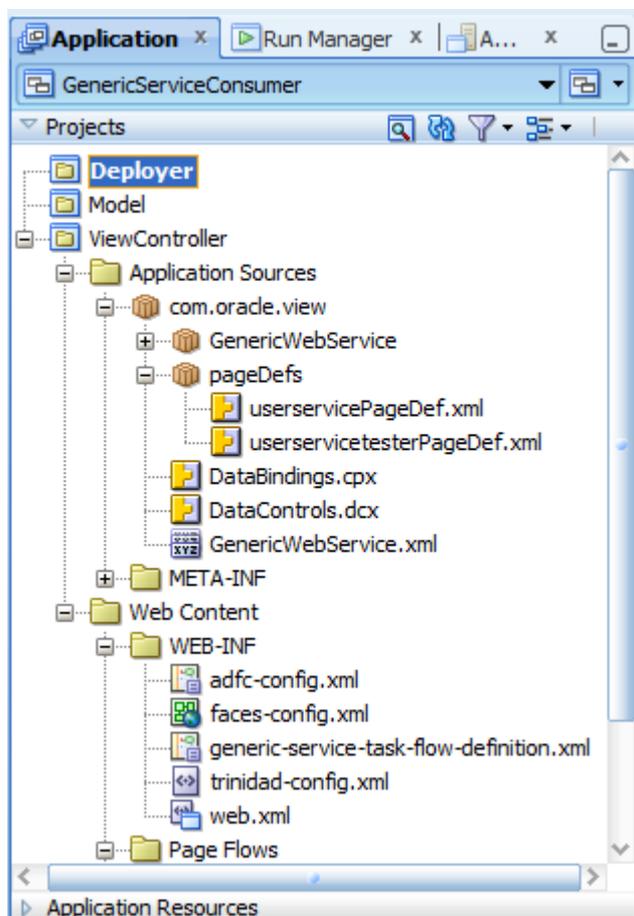
2. 次の画面で「Generic Project」を選択し、「OK」をクリックします。



3. Project NameフィールドにDeployerと入力し、「Finish」をクリックします。

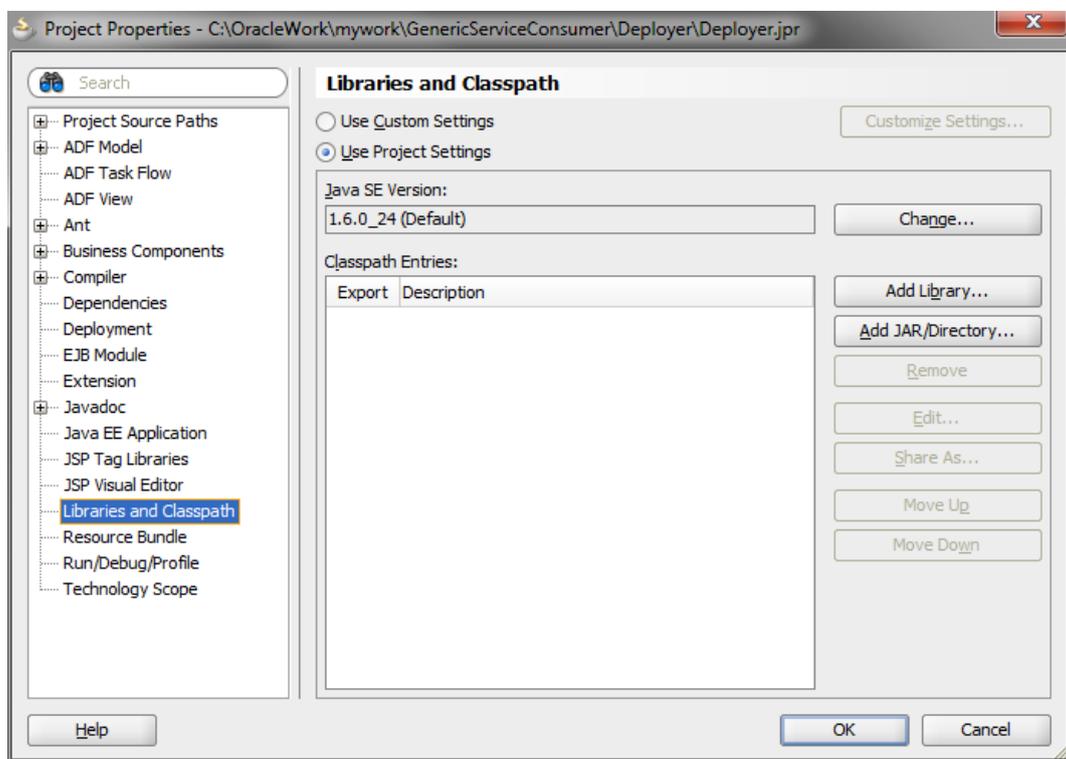


次に示すように、Deployerプロジェクトがナビゲータに表示されます。

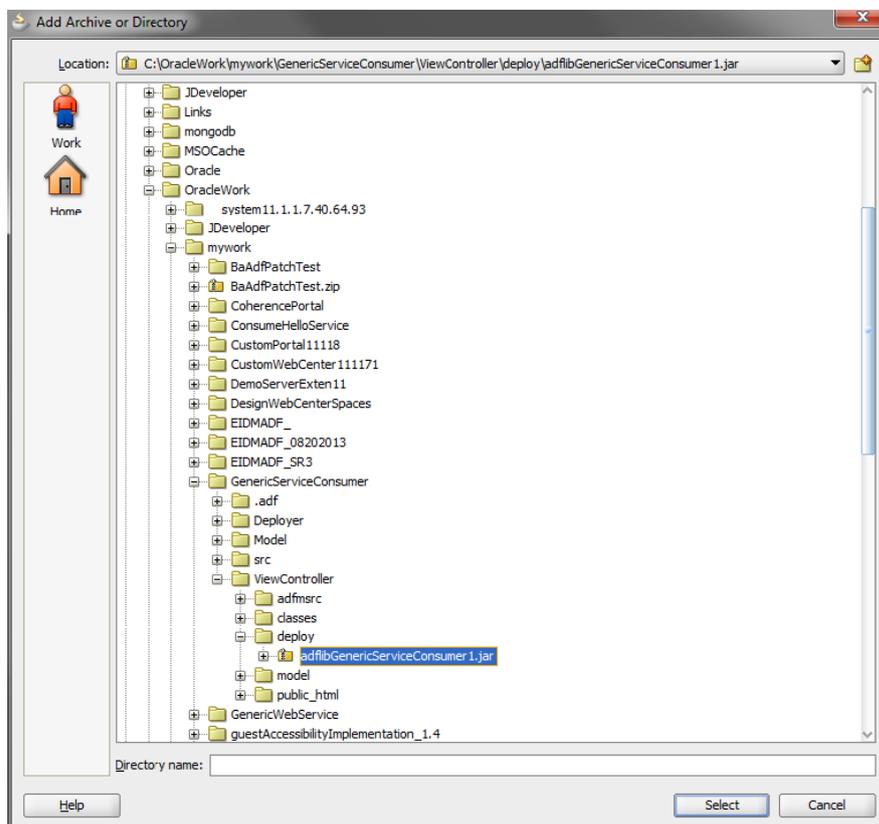


- 次に、ViewControllerデプロイメント・プロファイルから作成したOracle ADFライブラリのJARを新しいDeployerプロジェクトに追加する必要があります。これを行うには、次のステップを実行します。

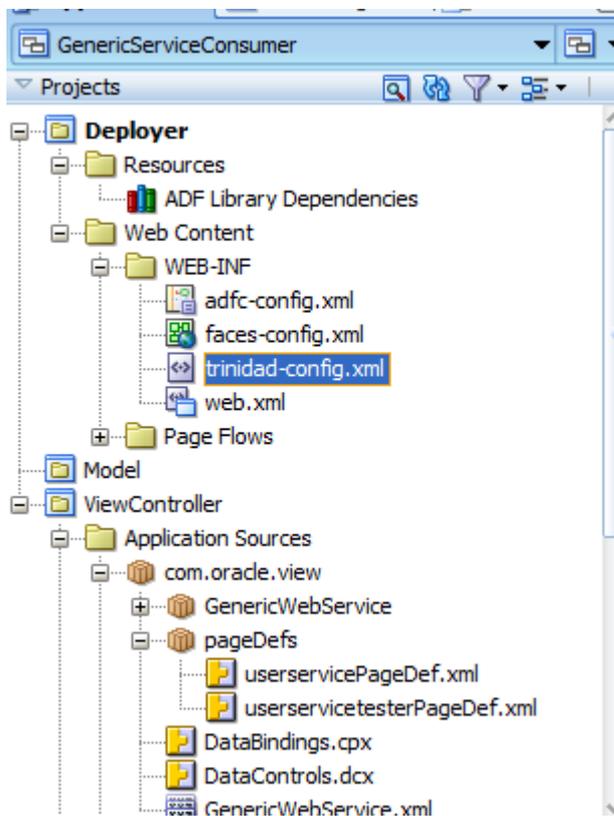
- a) 「Deployer」プロジェクトを右クリックし、「Project Properties」を選択します。
次の画面で「Libraries and Classpath」を選択します。



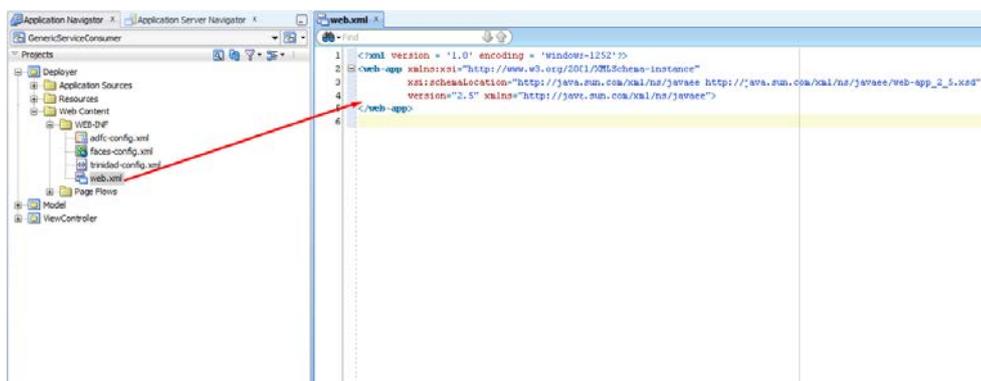
- b) 「Add JAR/Directory」をクリックします。次の画面でOracle ADFライブラリのJARファイルを選択します（下図を参照）。「Select」をクリックしてから、「OK」をクリックして確定します。



このアクションにより、汎用プロジェクトがOracle ADFへの依存性を持つプロジェクトに変更されます（下図を参照）。



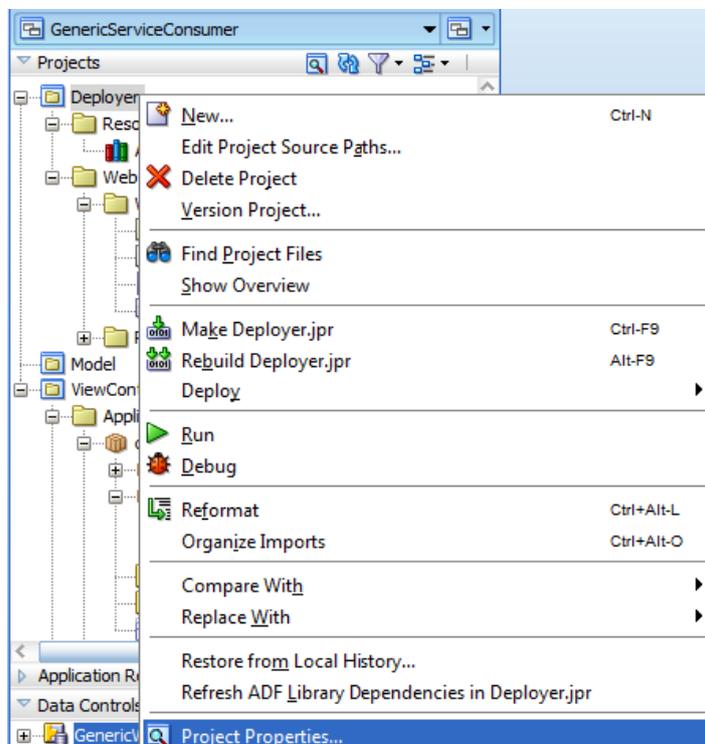
重要：WEB.XMLファイルにエントリが含まれていないことを再確認してください。この作業はDeployerプロジェクトに何らかのライブラリを追加するときに実行する必要があります。WEB.XMLファイル内にエントリがあると、デプロイ時に例外が発生します。詳しくは、巻末のトラブルシューティング・セクションを参照してください。WEB.XMLファイルが次のスクリーンショットのようになっていれば問題ありません。



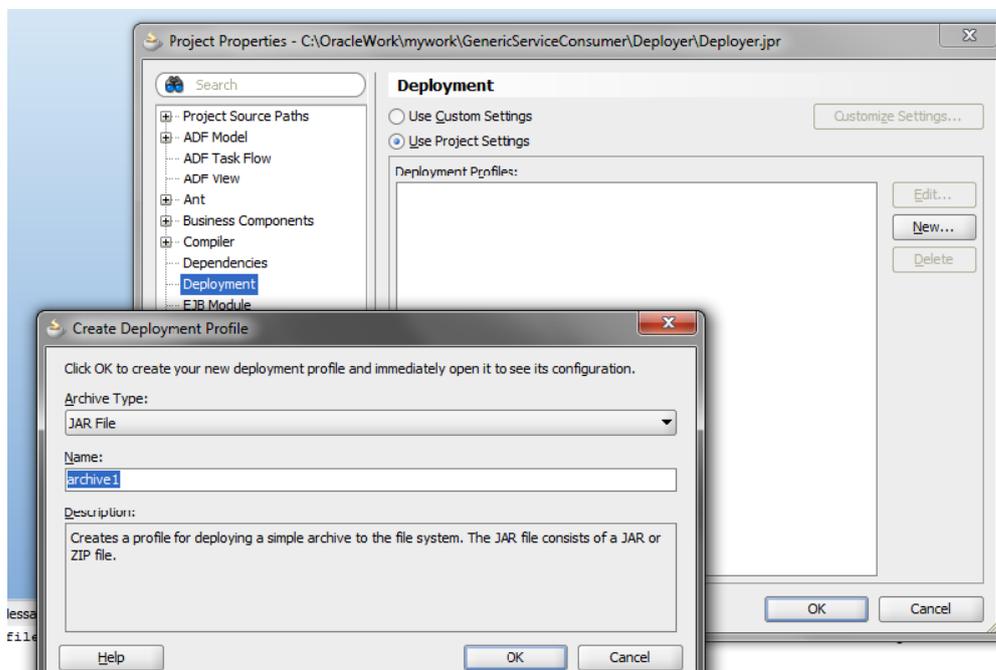
c) 変更を保存します。

次のステップでは、WebCenter Portalの管理対象サーバーに対して、同じプロジェクトの複数バージョンをデプロイするためのマニフェスト・ファイルを作成します。同じバージョンを持つOracle ADF共有ライブラリを複数デプロイすることはできませんが、マニフェスト・ファイルを使用してデプロイ・バージョンを上げることができます。

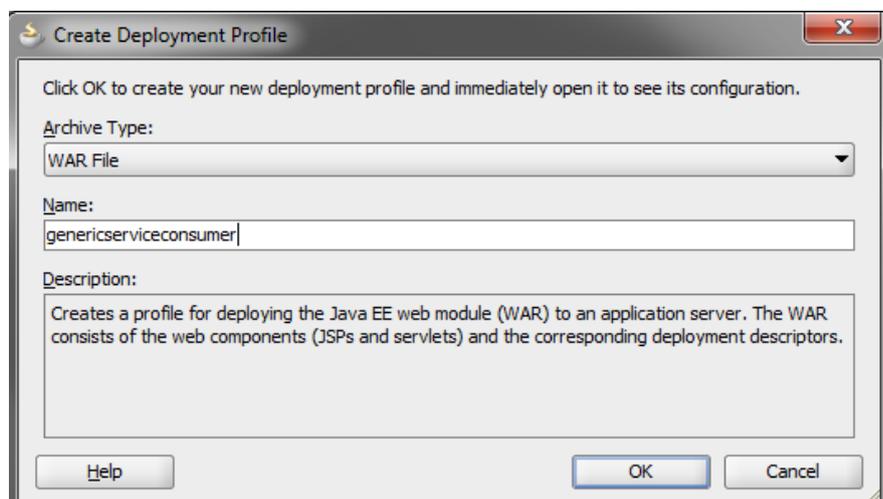
5. Deployerプロジェクトにマニフェスト・ファイルを追加する前に、新しいデプロイメント・プロファイルを作成する必要があります。デプロイメント・プロファイルを作成するには、「Deployer」プロジェクトをクリックして「Project Properties」を選択します。



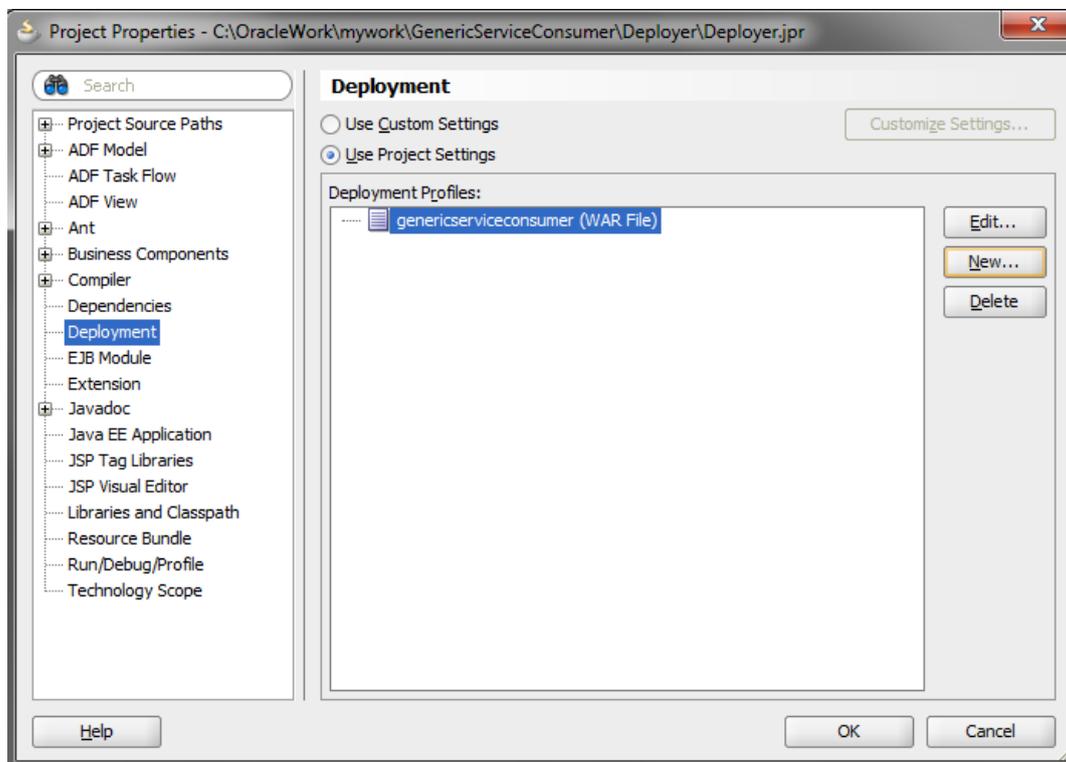
6. Project Properties画面で、「Deployment」→「New…」の順に選択します。



7. Archive Typeドロップダウンから「WAR File」を選択します。デプロイメント・プロファイル名（ここではgenericserviceconsumer）を入力し、「OK」をクリックします。メッセージが表示されたらデフォルトを受け入れます。

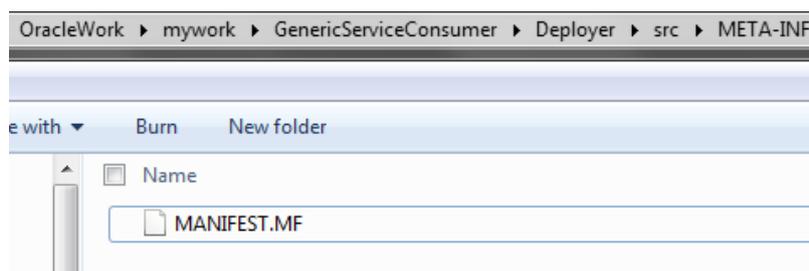


8. Deployment profile画面で、もう一度「OK」をクリックします。



9. マニフェスト・ファイルを作成します。プロジェクトのファイル・システムへ移動し、/src/META-INF/フォルダ内にMANIFEST.MFという名前のファイルを作成します（必要に応じて、フォルダ自体を作成します）。

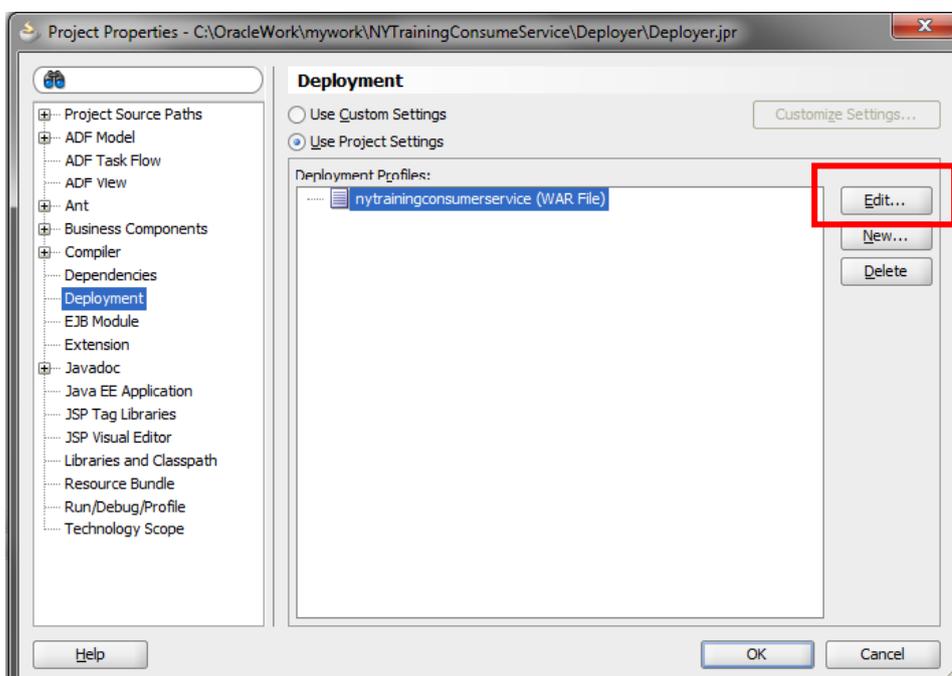
注：このステップはJDeveloperで実行することもできます。この場合、フォルダを2つ作成してからファイルを1つ作成します。

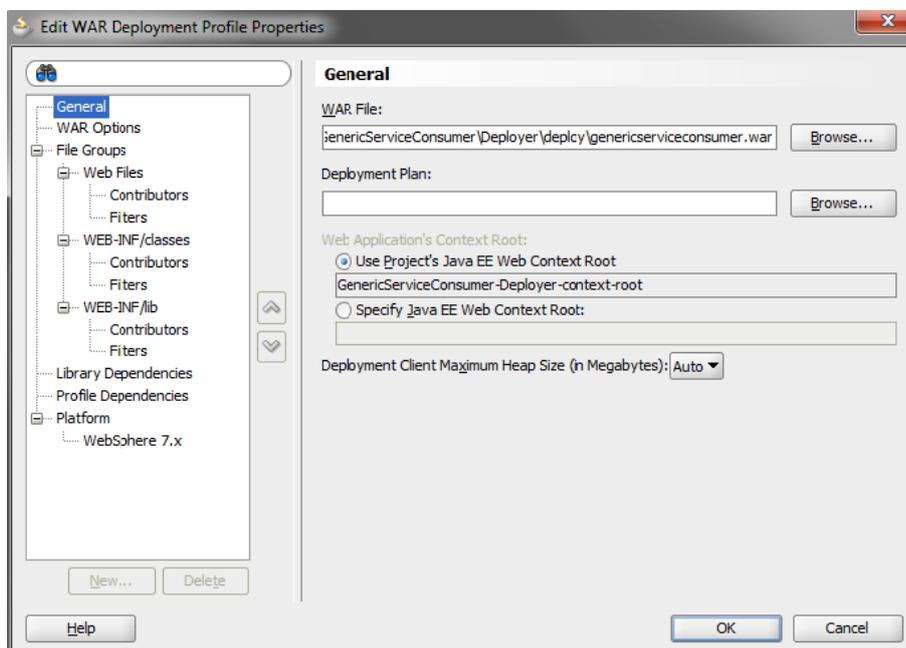


10. マニフェスト・ファイルにタスク・フロー情報を追加します。次に例を示します。

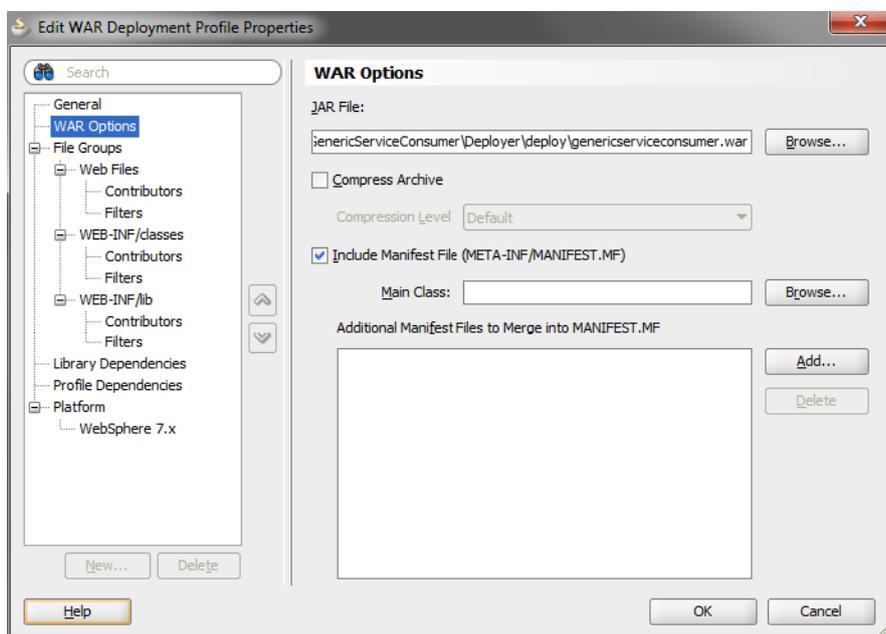
```
Manifest-Version: 1.0
Created-By: A-Team
Implementation-Title: training.generic.taskflow
Extension-Name: extend.portal.training.generic.taskflow
Specification-Version: 1.0.0
Implementation-Version: 1.0.0
Implementation-Vendor: Oracle
```

11. プロジェクトのデプロイメント・プロファイルにマニフェスト・ファイルを追加します。「Deployer」プロジェクトを右クリックし、「Project Properties」→「Deployment」の順に選択してからデプロイメント・プロファイルを選択します。「Edit...」をクリックしてプロファイルを編集します。

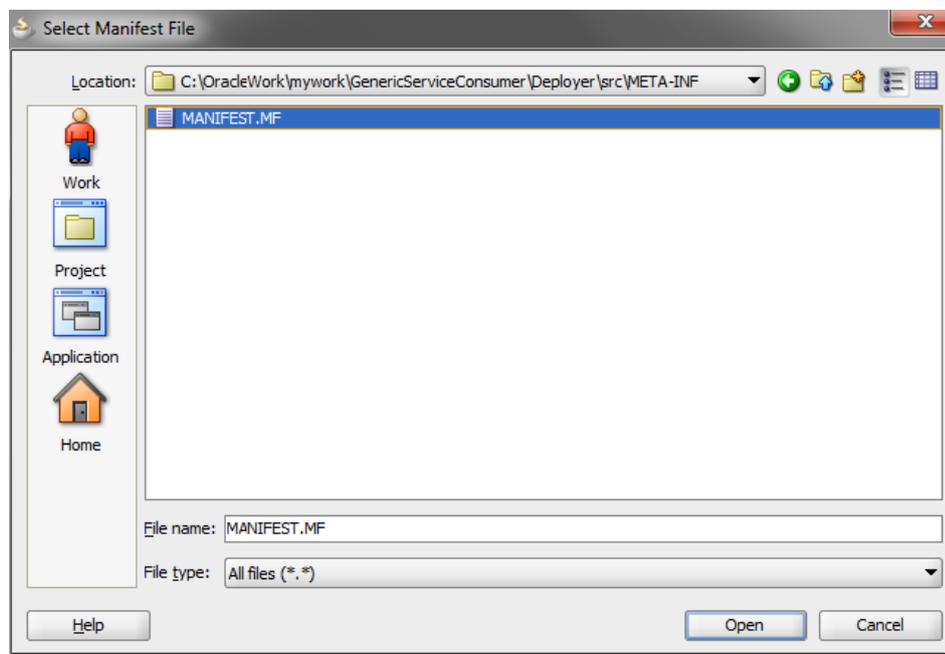




12. Edit WAR Deployment Profile画面で「WAR Options」を選択してから、「Include Manifest File」チェック・ボックスを選択します。

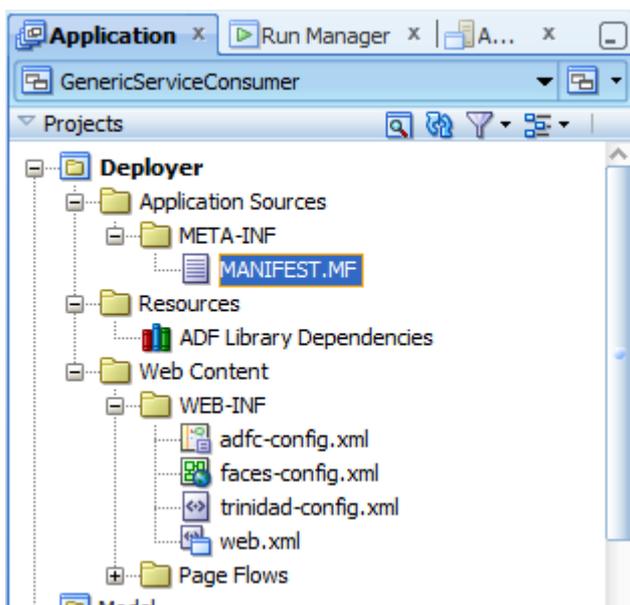


- 「Add…」をクリックしてマニフェスト・ファイルを選択し、デプロイメント・プロファイルにこれを追加します。次の画面でMETA-INFフォルダを開き、マニフェスト・ファイルを選択して「Open」をクリックし、「OK」をクリックして保存します（下図を参照）。



以上で、Deployerプロジェクトにサンプル・タスク・フローをデプロイする準備が整いました。

Deployerプロジェクト内にMANIFEST.MFファイルが表示されます。

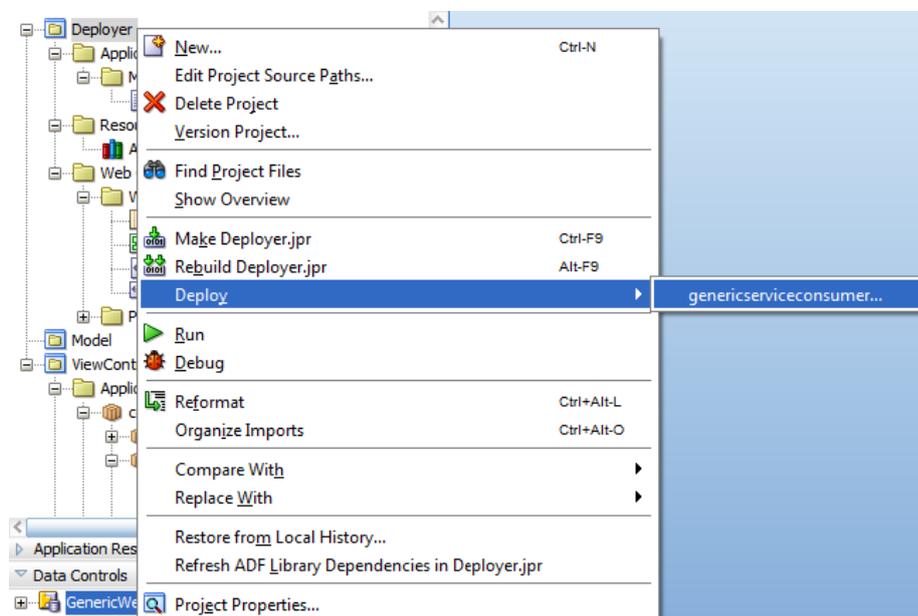


ステップ9：WebCenter PortalへのDeployerプロジェクトとサンプル・タスク・フロー（WARファイル）のデプロイ

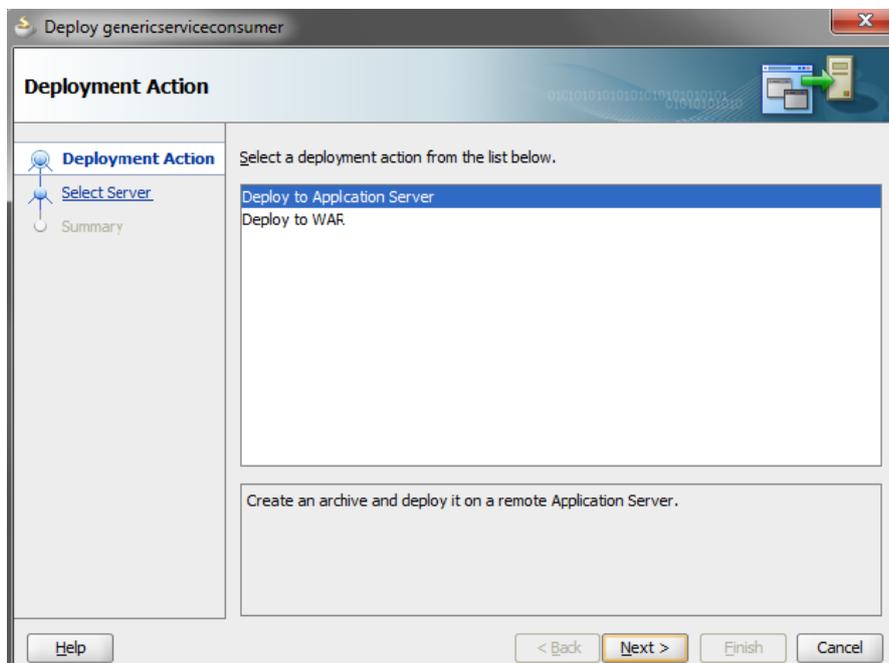
このステップでは、Deployerプロジェクトを使用して、ポータル・サーバーにサンプル・タスク・フローをデプロイします。

注：このステップを開始する前に、WebCenter Portalの管理対象サーバー（11.1.1.8.0）が稼働中であることを確認してください。

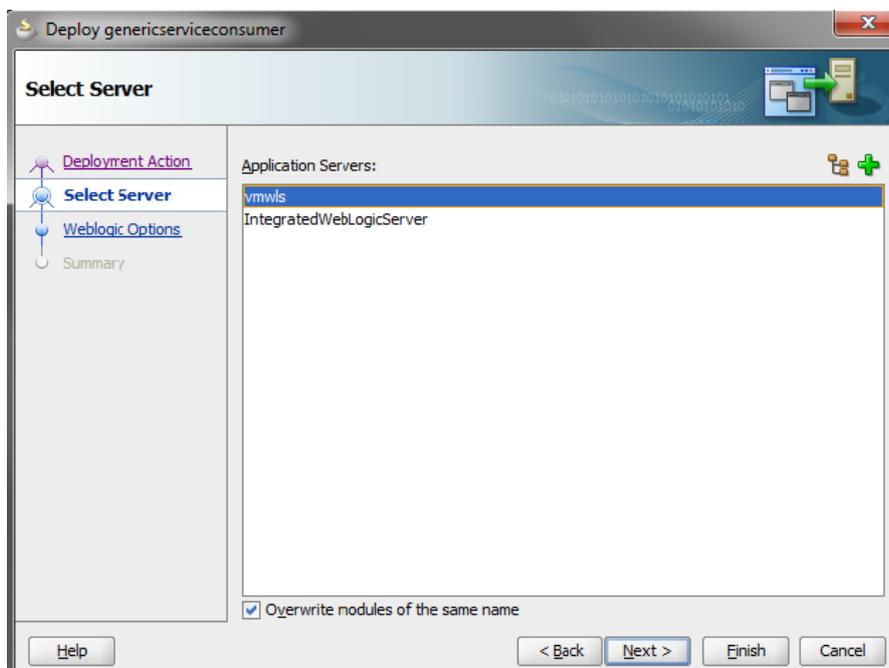
1. 「Deployer」プロジェクトを右クリックして「Deploy」を選択し、先ほど作成したデプロイメント・プロファイルを選択します。



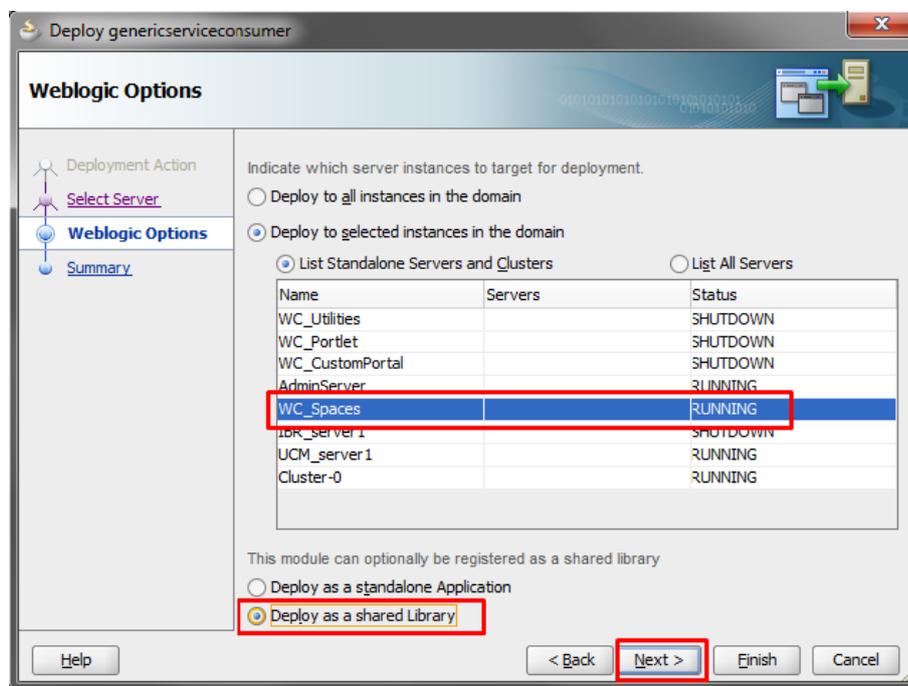
- Deployment Action画面で、「Deploy to Application Server」を選択して「Next」をクリックします。



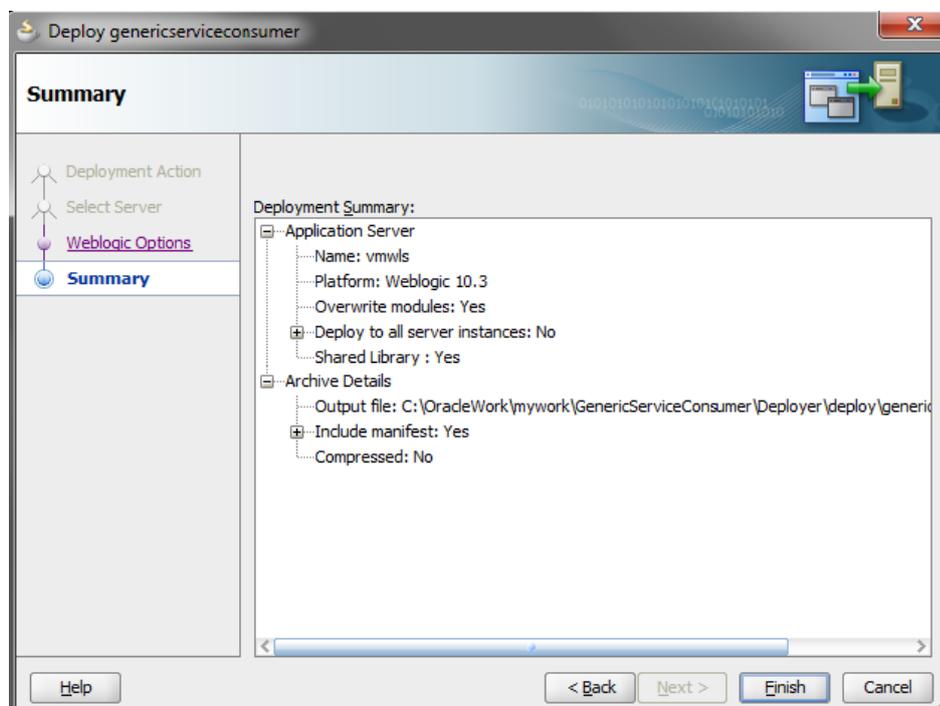
- 共有ライブラリのデプロイ先となるWebLogic Serverを指す接続を選択します。この例では、WebCenter PortalはVirtualBox (vmwls) 上で稼働しています。



4. Weblogic Options画面で、WebCenter Portalをデプロイする管理対象サーバー（「WC_Spaces」）を選択し、「Deploy as a shared library option」を選択します。



5. 各オプションを確認したら「Finish」をクリックして、サンプル・タスク・フローを含む Oracle ADF共有ライブラリをデプロイします。



デプロイが正しく完了すると、次のようなデプロイメント・ログが表示されます。

```

Deployment - Log x HTTP Analyzer x HTTP Analyzer Instances x
[03:36:04 PM] ---- Deployment started. ----
[03:36:04 PM] Target platform is (Weblogic 10.3).
[03:36:05 PM] Retrieving existing application information
[03:36:05 PM] Running dependency analysis...
[03:36:05 PM] Building...
[03:36:05 PM] Deploying profile...
[03:36:06 PM] Wrote Web Application Module to C:\OracleNork\mwork\GenericServiceConsumer\Deployer\deploy\generic-service-consumer_war
[03:36:06 PM] Deploying Application...
[03:36:07 PM] [Deployer:149191]Operation 'deploy' on application 'extend.portal.training.generic.taskflow (LibSpecVersion=1.0.0,LibImplVersi
[03:36:07 PM] [Deployer:149192]Operation 'deploy' on application 'extend.portal.training.generic.taskflow (LibSpecVersion=1.0.0,LibImplVersi
[03:36:07 PM] Application Deployed Successfully.
[03:36:07 PM] Elapsed time for deployment: 3 seconds
[03:36:07 PM] ---- Deployment finished. ---Deployment messages
  
```

- WebLogic管理コンソールを使用して、管理対象サーバーにライブラリが正しくデプロイされたことを確認します。たとえば、次の画面からは、バージョン1.0.0がマニフェスト・ファイルに指定されたとおりにデプロイされたことが分かります。

NAME	STATUS	LIBRARY	SIZE
emcore	Active	Library	100
extend.portal.training.generic.taskflow(1.0.0,1.0.0)	Active	Library	100
extend.spaces.shared.lib.training.taskflow(1.0.0,1.0.0)	Active	Library	100
extend.spaces.shared.lib.training.taskflow(1.0.0,1.0.1)	Active	Library	100
extend.spaces.shared.lib.training.taskflow(1.0.0,1.0.2)	Active	Library	100
extend.spaces.webapp(11.1.1,11.1.2)	Active	Library	100

注：上のスクリーンショットに示すように、デプロイされたライブラリは **extend.portal.training.generic.taskflow** という形式になっているはずですが、これは、MANIFEST.MF ファイルに指定した名前です。共有ライブラリの末尾にバージョンが表示されます。

WebCenter Portalでタスク・フローを使用する前に、タスク・フローをデプロイする管理対象サーバーを再起動する必要があります。ただし、サーバーを再起動する前に、次の項で説明するWebCenter Portal Server Extension Projectを使用して、WebCenter PortalにOracle ADF共有ライブラリを登録する必要があります。

注：タスク・フローから使用できるように、サンプルの汎用Webサービスがデプロイされていることを確認してください。このWebサービスは、WebCenter Portalからアクセスできる任意の管理対象サーバーにインストールできます。この例では、WebサービスはWebCenter Portalの管理対象サーバーにインストールされており、次のURLが割り当てられています。

<http://localhost:8888/GenericWebService-UserWebService-context-root/UserServiceSoap12HttpPort>

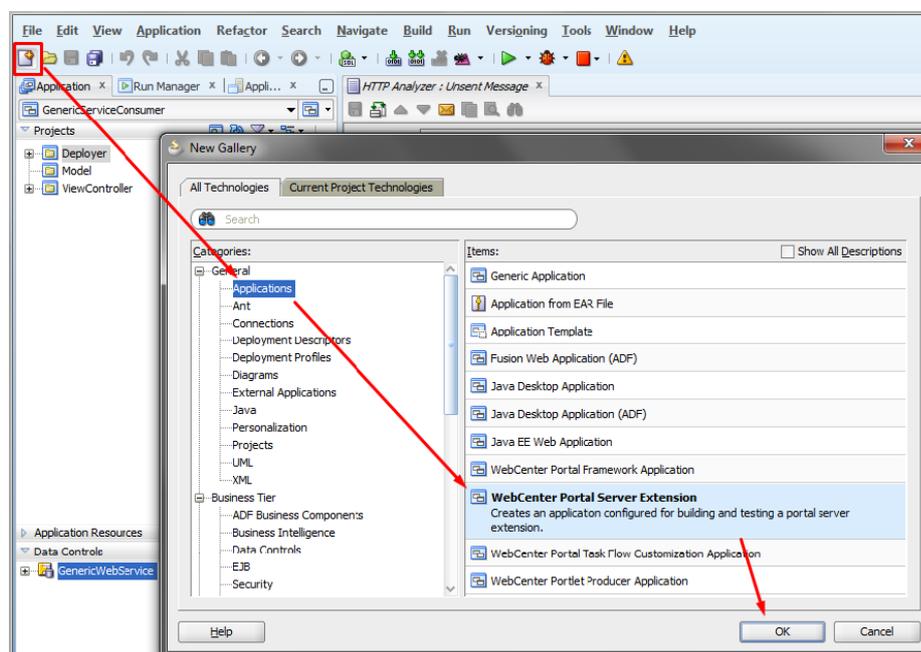
ブラウザからこのURLを開いてWebサービスをテストします（下図を参照）。

Endpoint		Information	
Service Name:	{http://oracle.com /}UserServiceWS	Address:	http://localhost:8888/GenericWebService-UserWebService-context-root/UserServiceSoap12HttpPort
Port Name:	{http://oracle.com /}UserServicePort	WSDL:	http://localhost:8888/GenericWebService-UserWebService-context-root/UserServiceSoap12HttpPort?wsdl
		Implementation class:	com.oracle.UserService

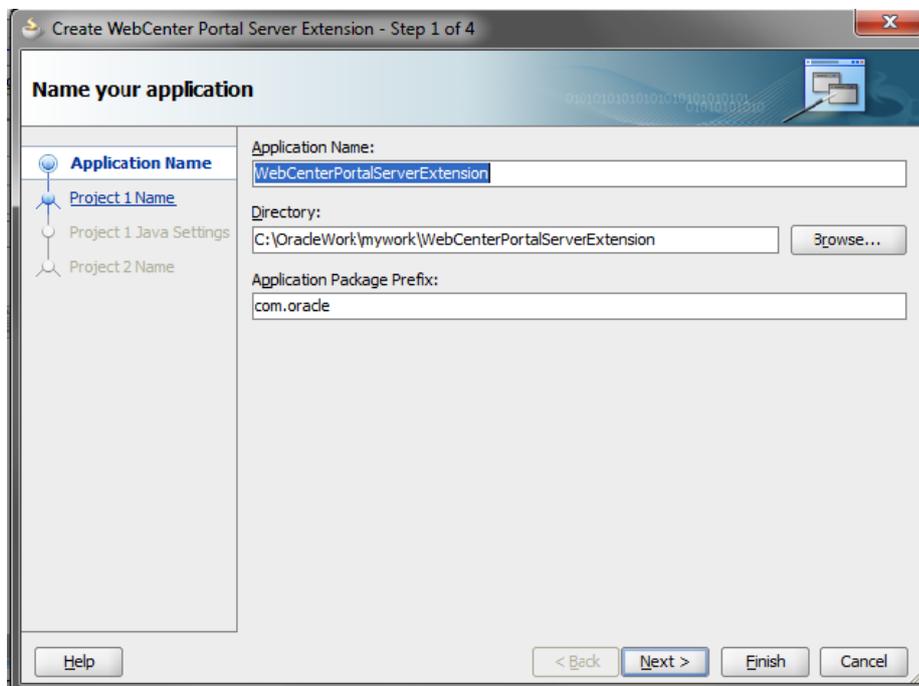
ステップ10：WebCenter Portalへのサンプル・タスク・フローの登録

タスク・フローを含むOracle ADF共有ライブラリ・ファイルのデプロイは完了しており、使用する準備ができています。しかし、WebCenter PortalにOracle ADF共有ライブラリを登録して、WebCenter Portalがデプロイされている管理対象サーバーを再起動するまで、WebCenter Portalのリソース・カタログにタスク・フローは表示されません。このステップでは、WebCenter Portal Server Extension Projectを使用して、追加のOracle ADF共有ライブラリをWebCenter Portalに登録する方法について説明します。

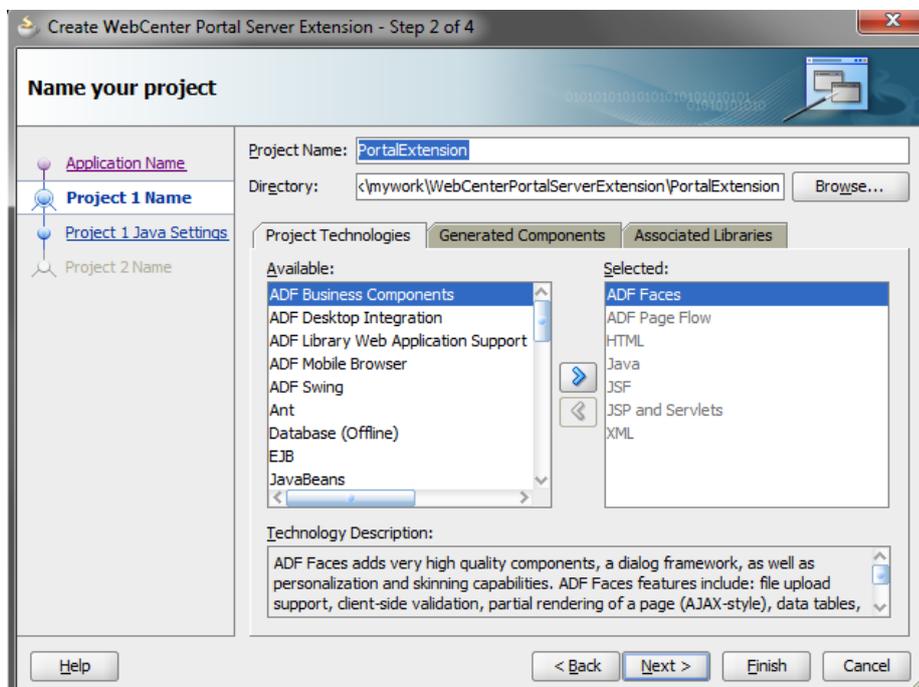
1. WebCenter Portal Server Extensionアプリケーションを作成するには、「New」アイコンをクリックし、「Applications」を選択してから「WebCenter Portal Server Extension」を選択します。「OK」をクリックします。



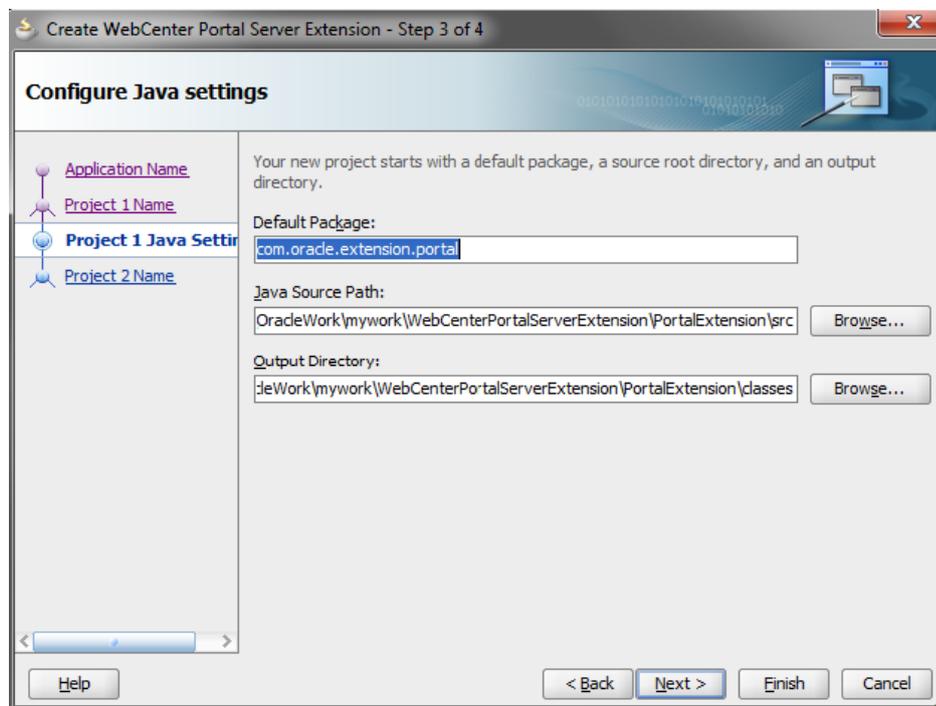
2. Name your application画面でアプリケーション名を入力し（またはデフォルトを受け入れ）、「Next」をクリックします。



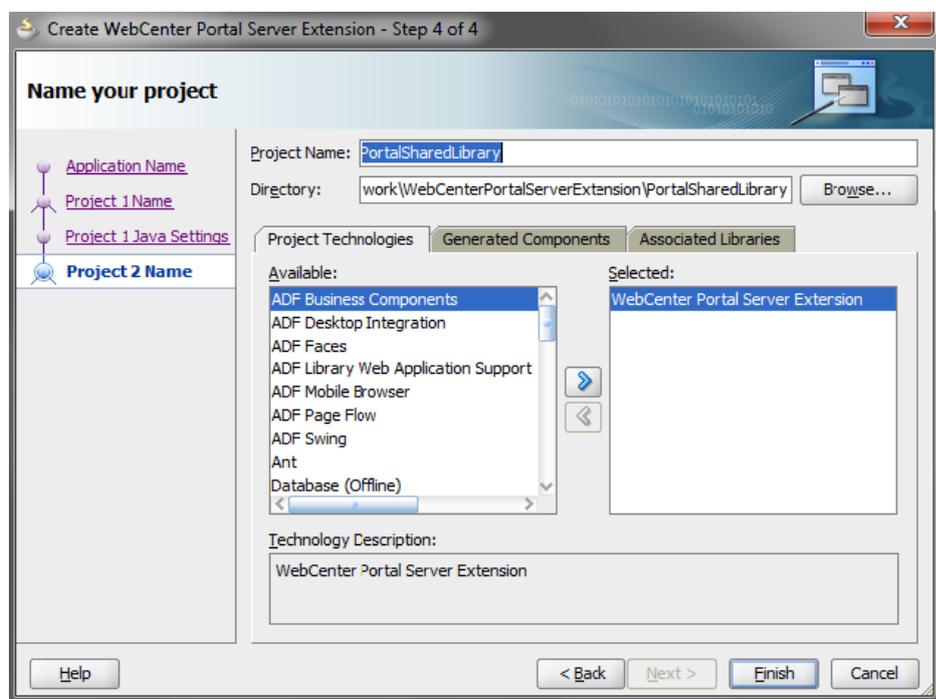
3. Project 1 Name画面でデフォルト設定のままにして「Next」をクリックします。



4. **Configure Java Settings**画面でデフォルト設定のままにするか、新しいデフォルト・パッケージ名を入力します。



5. **Project 2 Name**画面でデフォルト設定のままにして「Finish」をクリックします。

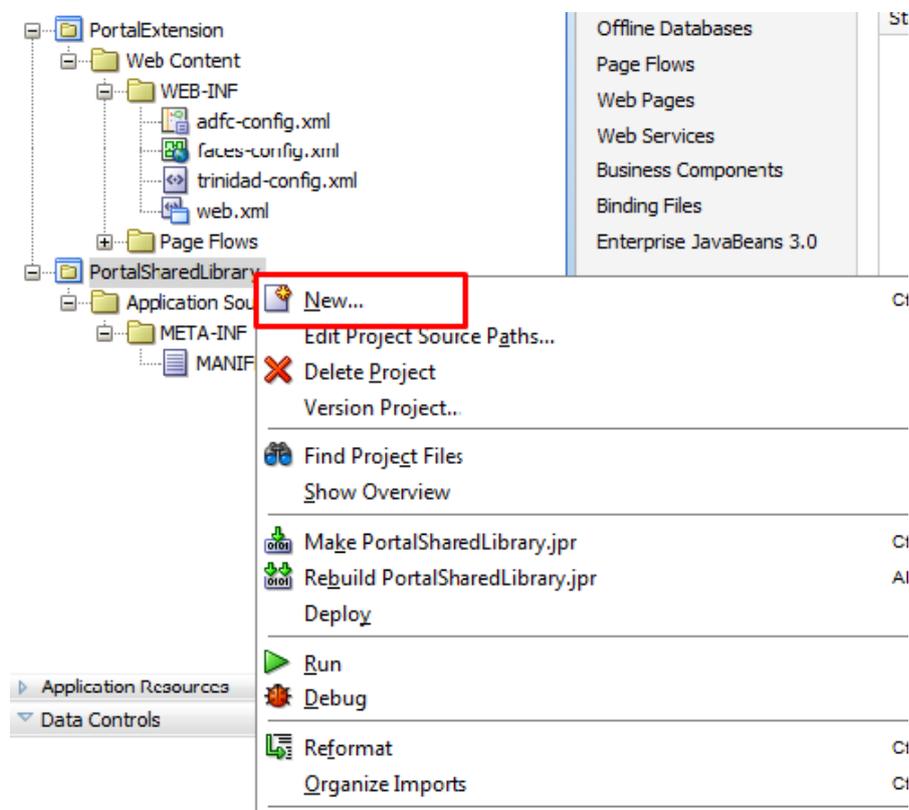


JDeveloperに次の2つのプロジェクトが表示されます。

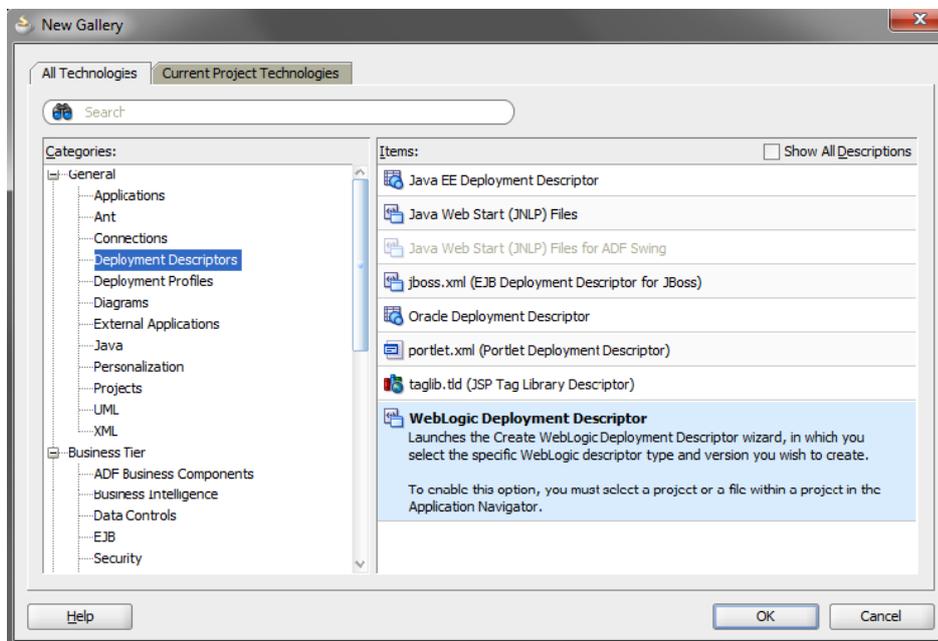
- PortalSharedLibrary** : 以降のステップでは、PortalSharedLibraryプロジェクトを使用して、WebCenter Portalで使用するOracle ADF共有ライブラリをweblogic.xml内に登録します。このプロジェクトには、extend.spaces.webappと呼ばれるWARデプロイメント・プロファイルが含まれています。後でextend.spaces.webapp.warをデプロイするときに、weblogic.xmlに追加で登録したすべてのカスタム共有ライブラリが、extend.spaces.webapp.warの依存ライブラリとして追加され、WebCenter Portalで利用可能になります。
- PortalExtension** : PortalExtensionはスタータ・プロジェクトですが、タスク・フローやデータ・コントロール、マネージドBeanといったカスタムのOracle ADFコンポーネントを作成するためにも使用できます。このプロジェクトを使用する場合、PortalExtensionプロジェクトをOracle ADFライブラリのJARファイルにデプロイし、PortalSharedLibraryプロジェクトから生成されたextend.spaces.webapp.war共有ライブラリ・ファイルの依存ライブラリとして追加する必要があります。

注：本書の例ではPortalExtensionプロジェクトを*使用していません*。

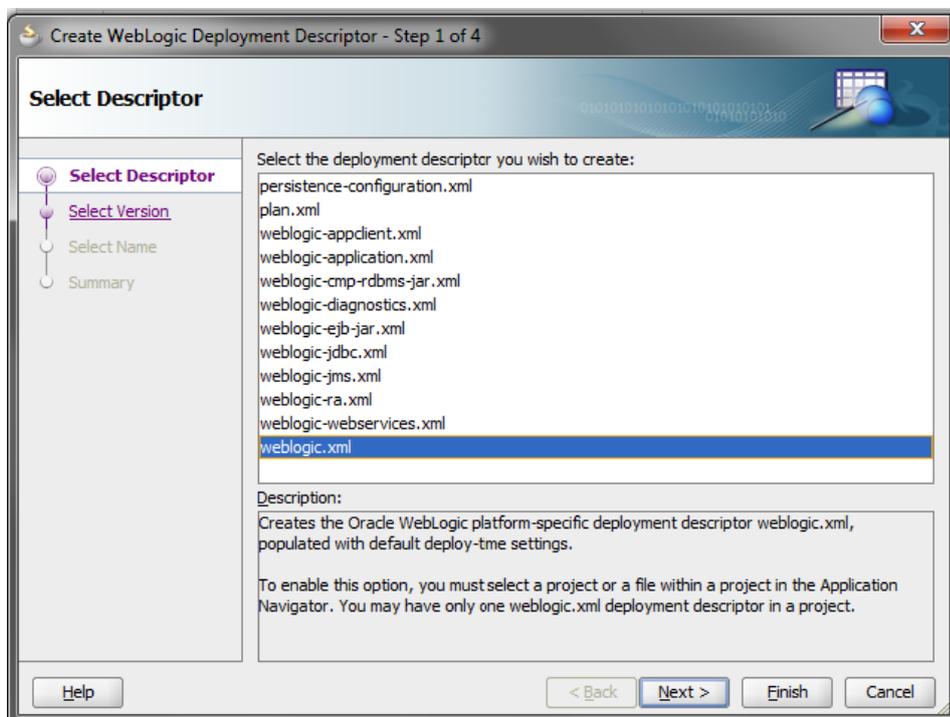
- デフォルトでは、PortalSharedLibraryプロジェクトにはWebLogic Serverのデプロイメント・ディスクリプタ（weblogic.xml）が含まれていません。これを作成するには、「PortalSharedLibrary」プロジェクトを右クリックして「New」を選択します。



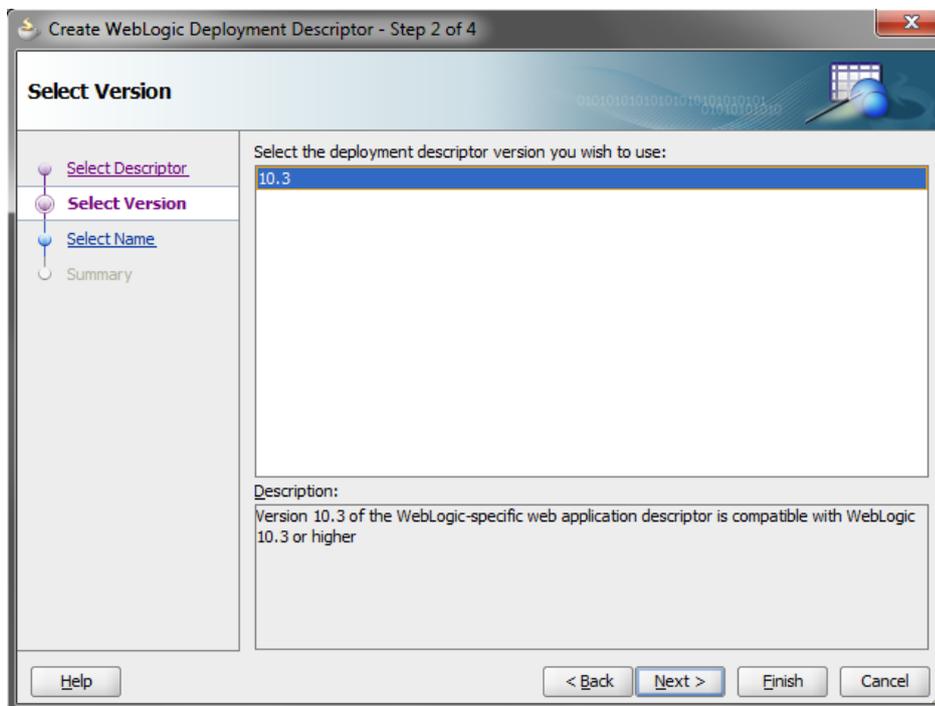
7. New Gallery画面で「Deployment Descriptors」を選択してから「WebLogic Deployment Descriptor」を選択し、「OK」をクリックします。



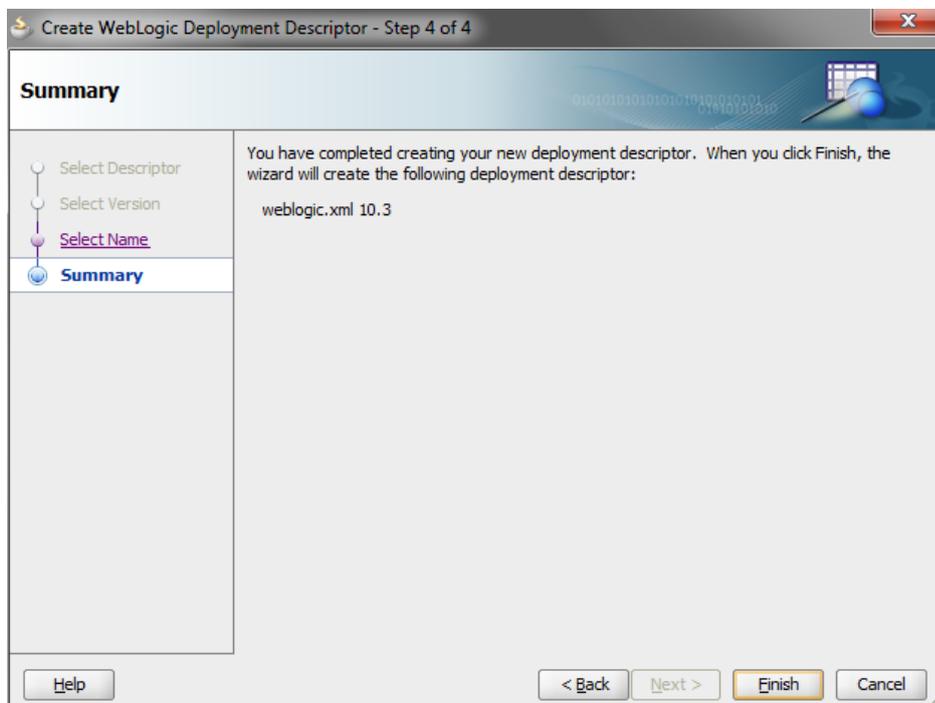
8. Select Descriptor画面で「weblogic.xml」を選択し、「Next」をクリックします。



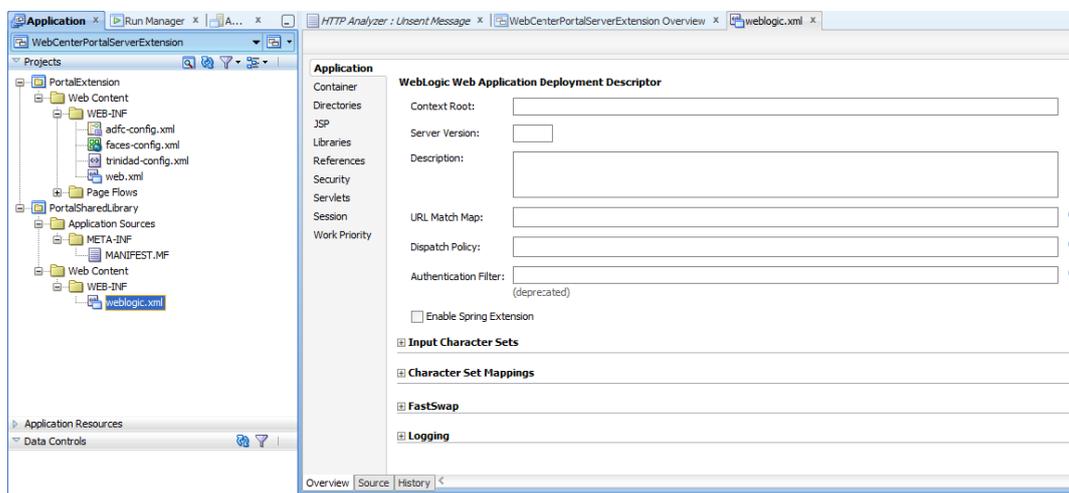
9. このディスクリプタがサポートするWebLogic Serverバージョンを選択し、「Next」をクリックします。



10. 「Finish」をクリックしてデプロイメント・ディスクリプタを作成します。

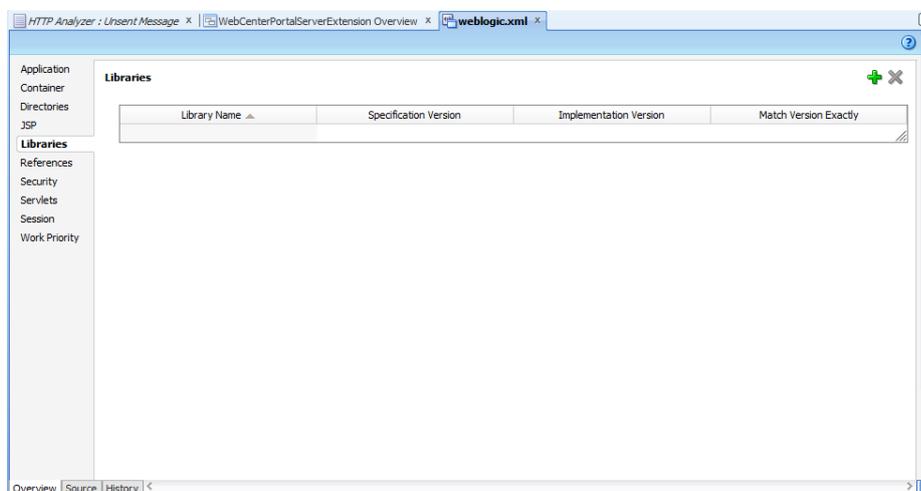


これで、このプロジェクトにweblogic.xmlファイルが含まれ、先ほどデプロイしたOracle ADF共有ライブラリのJARファイルをここに登録できるようになりました。

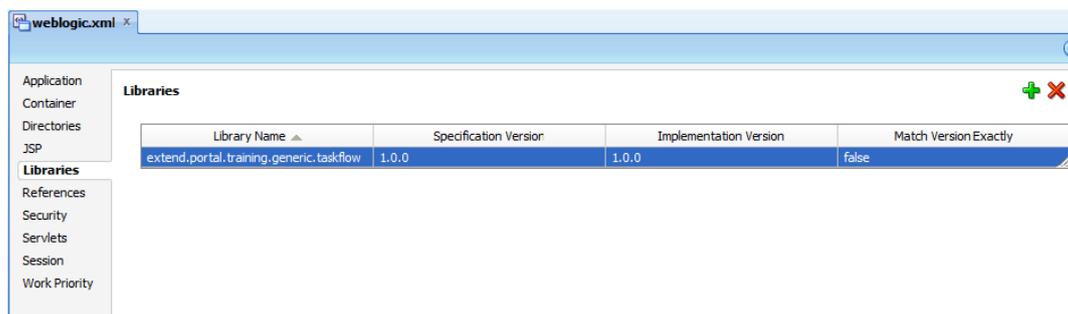


11. weblogic.xmlファイルをまだ開いていない場合はダブルクリックして開きます。
12. 左側のタブ・メニューから「Libraries」を選択します。Librariesページを使用して、WebCenter Portalで使用するOracle ADF共有ライブラリの名前を指定できます。

注：WebCenter Portalで使用するOracle ADF共有ライブラリはすべて、同じWebCenter Portal Server Extensionプロジェクト内に指定することを推奨します。共有ライブラリごとに新しいプロジェクトを作成する必要はありません。何らかの理由で複数のWebCenter Portal Server Extensionプロジェクトを作成する場合、同じWebCenter管理対象サーバーにデプロイするには、マニフェスト・ファイルの構成（MANIFEST.MF）を変更して異なる名前空間を指定する必要があります。

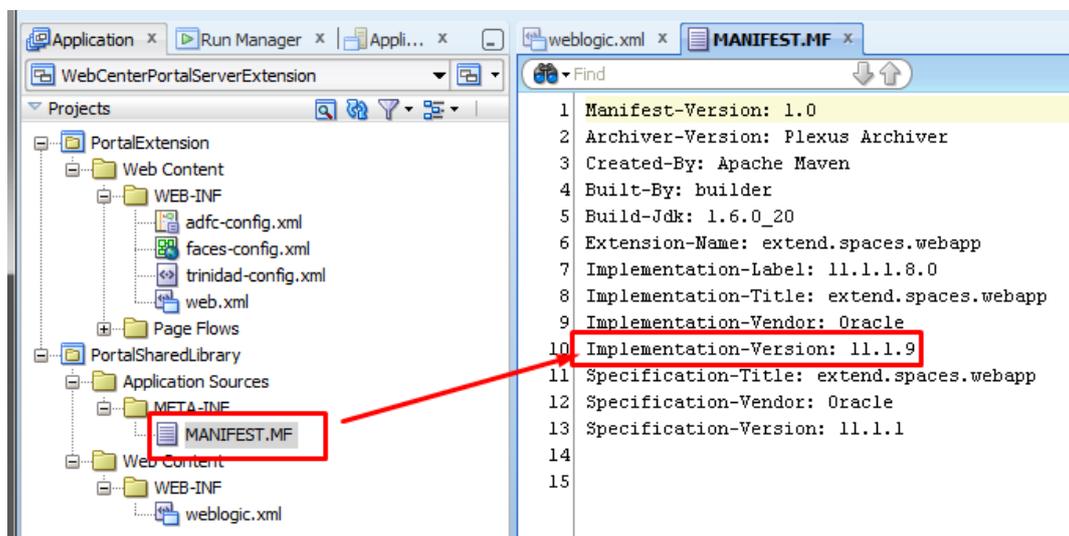


13. ライブラリ名とバージョン情報 (`extend.portal.training.generic.taskflow`) を追加して、先ほどデプロイしたOracle ADF共有ライブラリのWARを登録します (下図を参照)。ここに指定する情報がDeployerのマニフェスト・ファイルに指定したバージョン情報と一致するようにしてください。



注：複数のOracle ADF共有ライブラリを登録することは可能ですが、コンポーネントごとにアセット名とリソース名が異なる必要があります。たとえば、2番目のOracle ADF共有ライブラリを登録する場合、プロジェクトには別のデータ・バインディング名が含まれる必要があります。

14. デプロイ時の例外を防止するには、MANIFEST.MFファイルの実装バージョンを上げる必要があります。**MANIFEST.MF**ファイルを開き、`extend.spaces.webapp.war`の実装バージョンを変更します (下図を参照)。

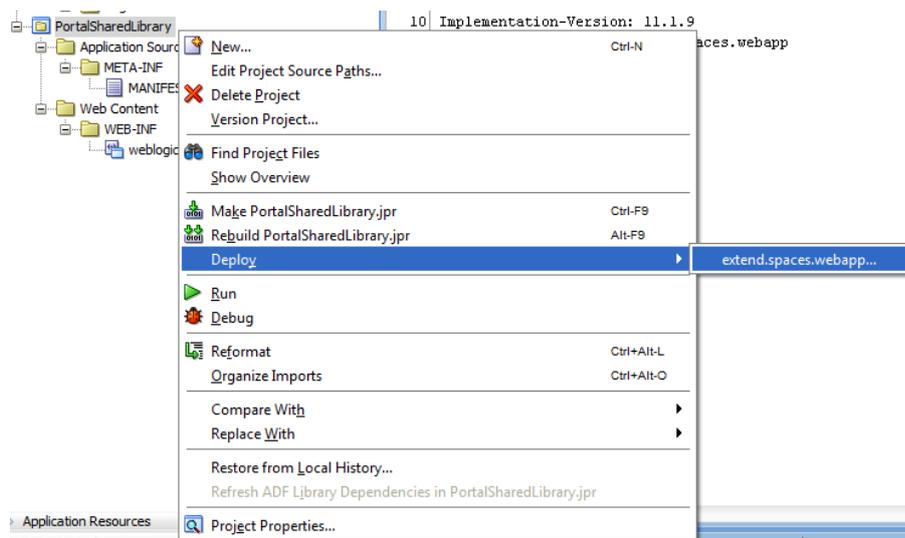


これで、PortalSharedLibraryプロジェクトをWebCenter Portal管理対象サーバーにデプロイできるようになりました。

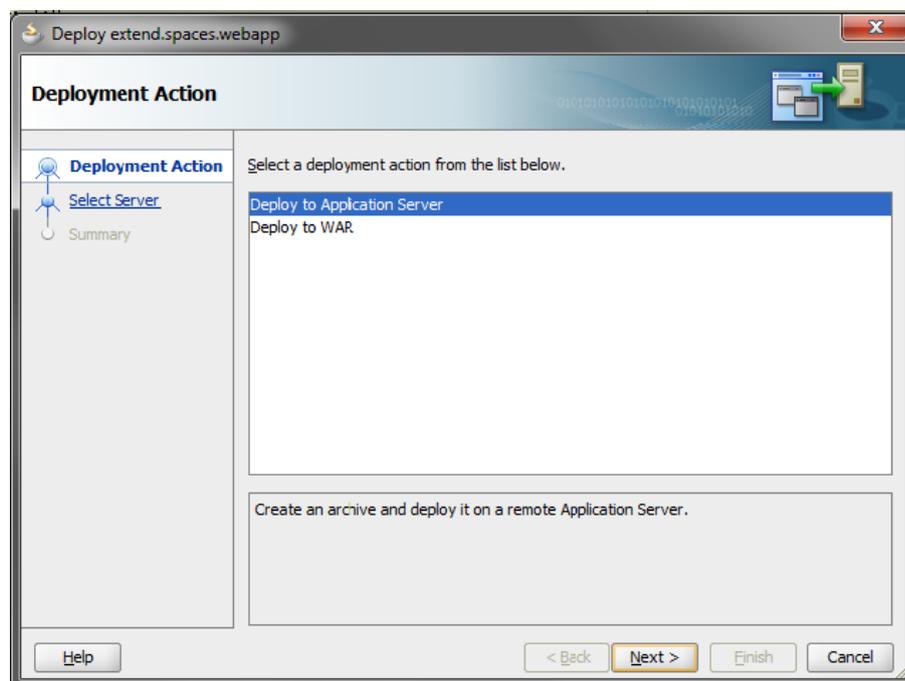
注：上のスクリーンショットでは、実装バージョンを11.1.9という高い値に上げています。これは、何度かテストを行ったためです。再デプロイするたびに、新しいバージョンがインストールされるようにこの数字を増やす必要があります。

ステップ11：WebCenter PortalへのPortalSharedLibraryのデプロイ

1. 「PortalSharedLibrary」を右クリックして「Deploy」を選択し、デフォルトのデプロイメント・プロファイルであるextend.spaces.webappを選択します。

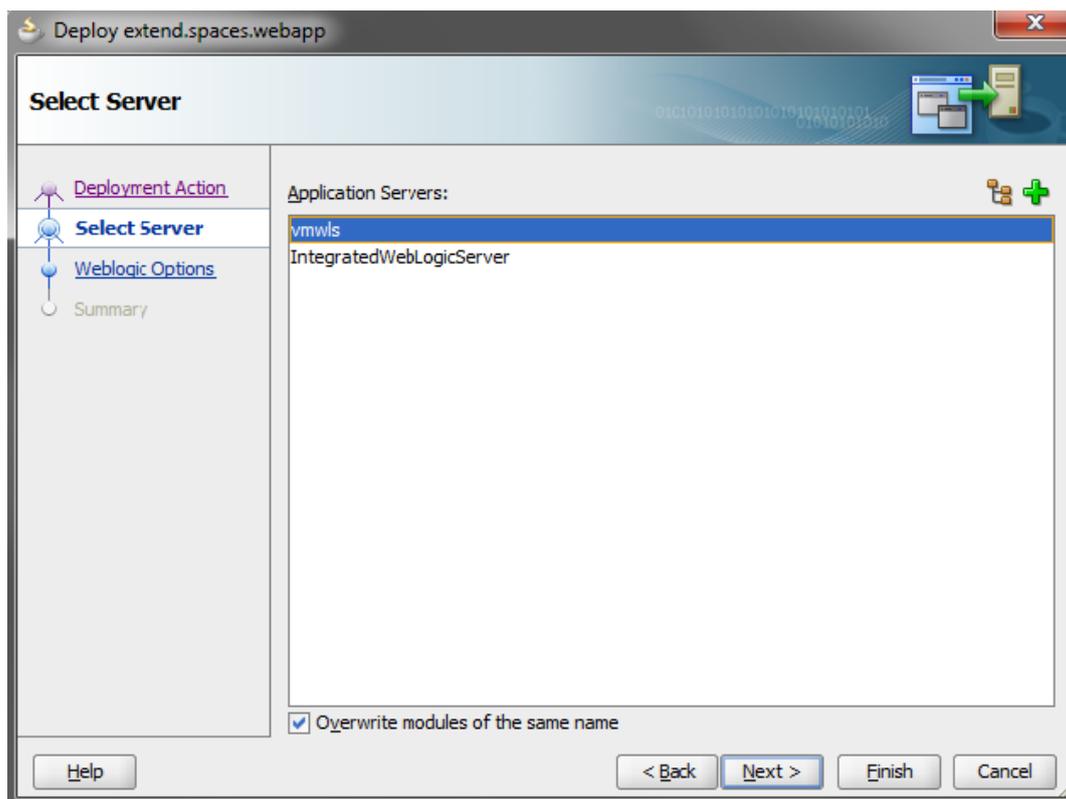


2. 「Deploy to Application Server」を選択して「Next」をクリックします。

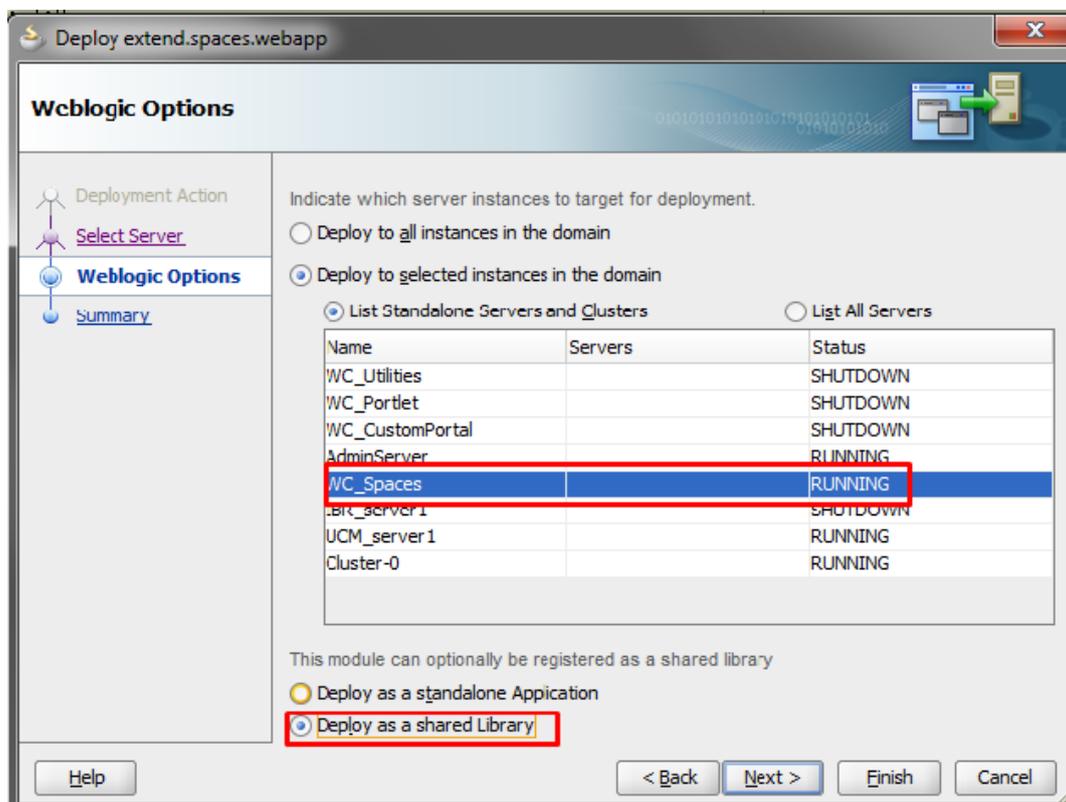


- 共有ライブラリのデプロイ先となるWebLogic Serverを指す接続を選択します。この例では、WebCenter PortalはVirtualBox (vmwls) 上で稼働しています。

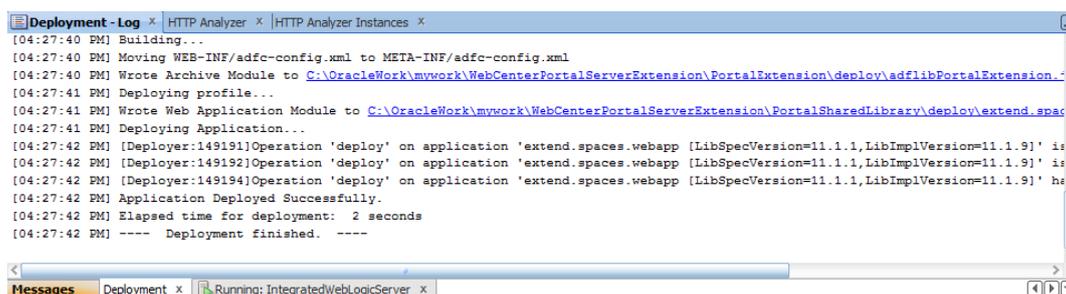
注：この例で示しているのは、JDeveloperからの直接デプロイです。必要に応じて、PortalSharedLibraryプロジェクトをWARファイル (extend.spaces.webapp.war) にデプロイしてから、WLSTスクリプトを使用してこのWARファイルを環境にデプロイできます。



4. WebLogic Options画面で、WebCenter Portalサーバーをデプロイする管理対象サーバーまたはクラスタ（この例では「WC_Spaces」）を選択し、「Deploy as a shared library option」を選択します。



5. デプロイメント・ログにエラー・メッセージが表示されていない場合、デプロイは正しく完了しています。



6. WebLogic Server管理コンソールにログインし、プロジェクトがデプロイされていることを確認します。

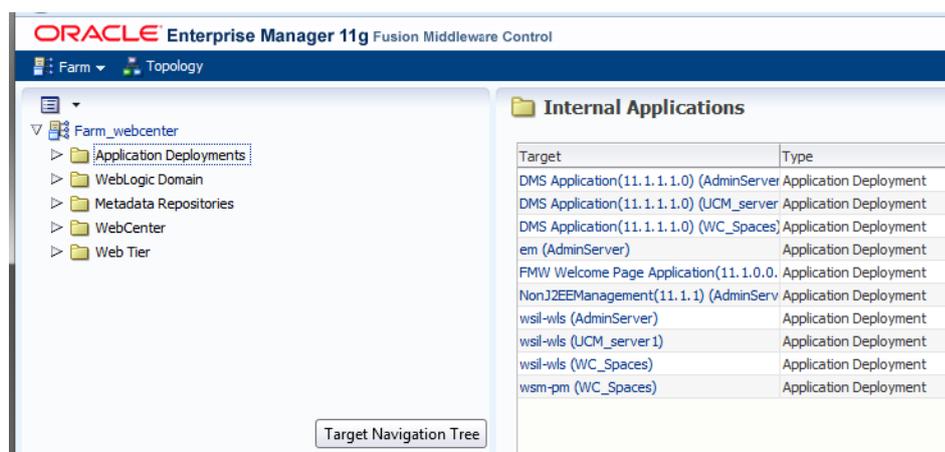
 emcore	Active		Library	100
 extend.portal.training.generic.taskflow(1.0.0,1.0.0)	Active		Library	100
 extend.spaces.shared.lib.training.taskflow(1.0.0,1.0.0)	Active		Library	100
 extend.spaces.shared.lib.training.taskflow(1.0.0,1.0.1)	Active		Library	100
 extend.spaces.shared.lib.training.taskflow(1.0.0,1.0.2)	Active		Library	100
 extend.spaces.webapp(11.1.1,11.1.2)	Active		Library	100
 extend.spaces.webapp(11.1.1,11.1.6)	Active		Library	100
 extend.spaces.webapp(11.1.1,11.1.7)	Active		Library	100
 extend.spaces.webapp(11.1.1,11.1.8)	Active		Library	100
 extend.spaces.webapp(11.1.1,11.1.9)	Active		Library	100
 FMW Welcome Page Application (11.1.0.0.0)	Active	✔ OK	Enterprise Application	5
 GenericWebService-UserWebService-context-root	Active	✔ OK	Web Application	100

7. WebCenter Portalが稼働している管理対象サーバー（この例ではWC_Spaces）を再起動します。サーバーを再起動したら、WebCenter Portalでサンプル・タスク・フローにアクセスできます。

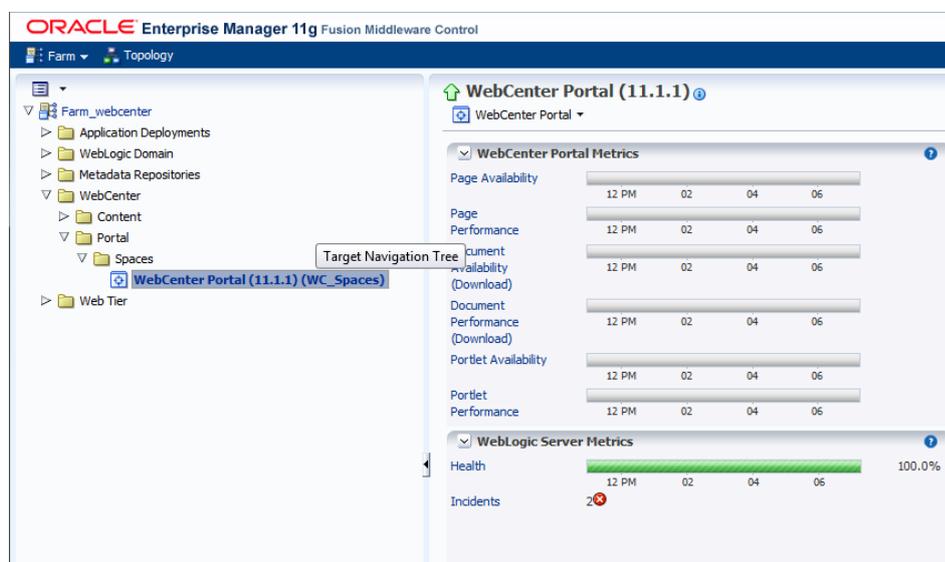
ステップ12：WebCenter PortalへのWebサービス接続の登録

カスタムのサンプル・タスク・フローにはWebサービス接続が必要です。この接続はWebCenter Portalインスタンスで構成する必要があります。WebCenter Portal用のWebサービス接続を構成するには、次のステップを実行します。

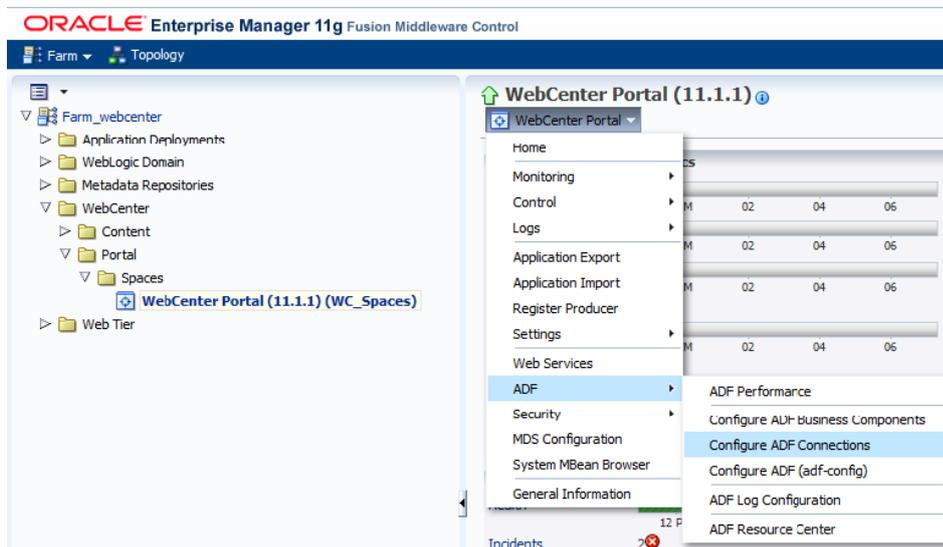
1. 構成の変更と追加を実行するには、Oracle Enterprise Manager (Oracle EM) を使用する必要があります。十分な権限を持ったユーザーを使用してOracle EMにログインします。



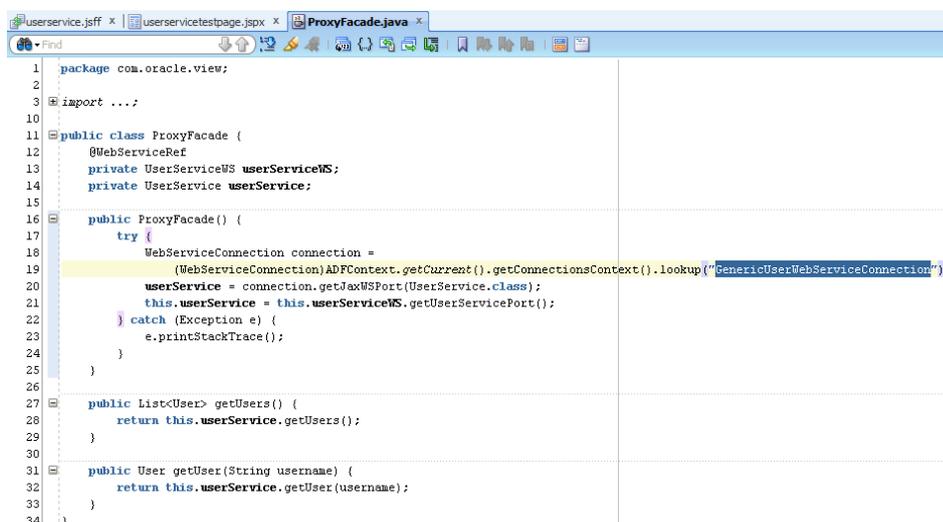
2. 「WebCenter」→「Portal」→「Spaces」へ移動して、「WebCenter Portal」アプリケーションをクリックします。

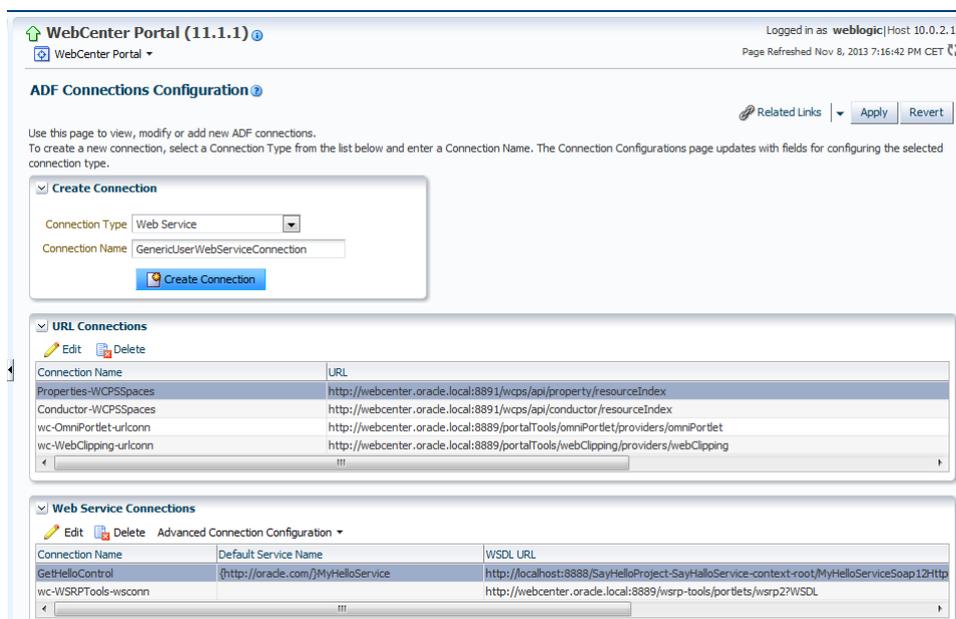


3. WebCenter Portalメニューから「ADF」→「Configure ADF Connections」を選択します。

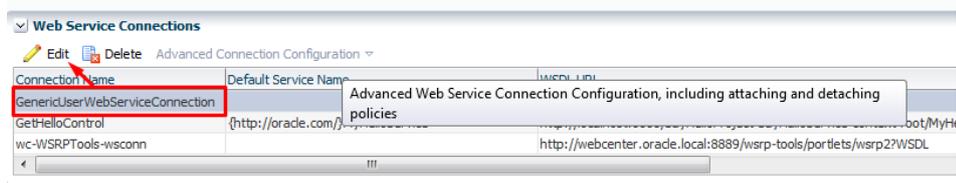


4. ProxyFacade.javaクラス（次のスクリーンショットを参照）で使用されたものと同じ名前を持つ新しいWebサービス接続を追加します。「Create Connection」ボタンをクリックして接続を作成します。

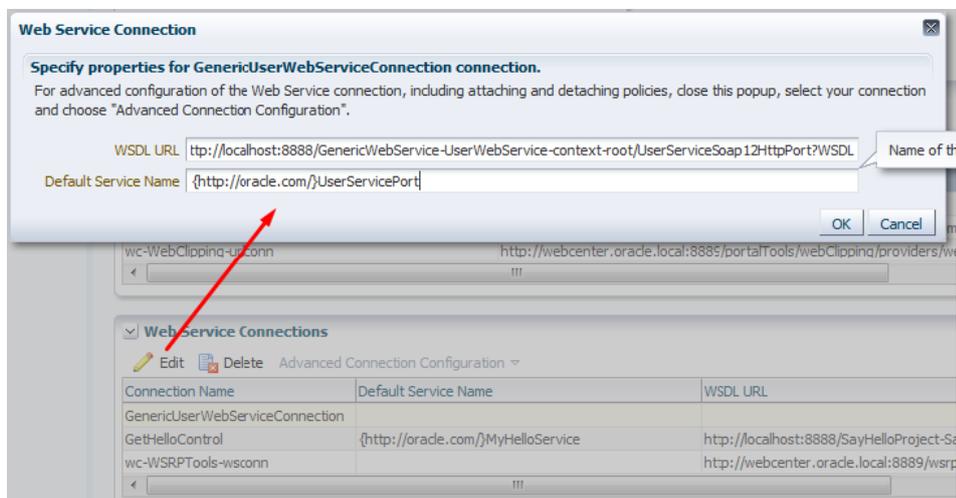




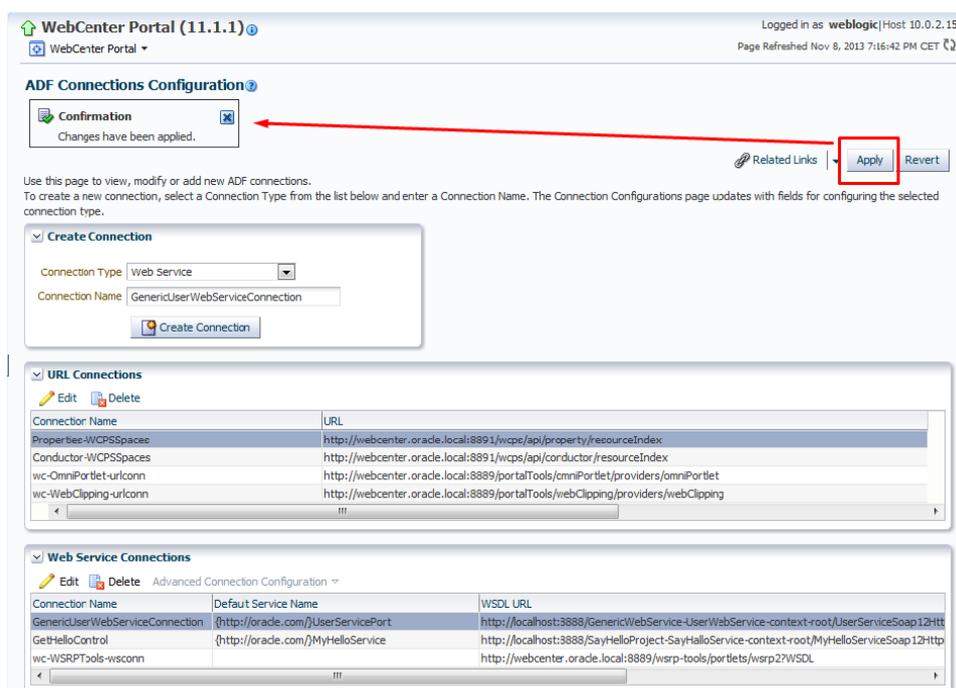
5. 新しい接続がWeb Service Connectionsセクションに表示されます。接続名をクリックしてから「Edit」アイコンをクリックします。



6. 次に示すように接続プロパティを指定し、「OK」をクリックして確定します。



7. 「Apply」ボタンをクリックしてMDSにこの設定を適用します。



この接続を作成した後にサーバーを再起動する必要はない点に注意してください。

注：複数の環境間で接続を移動するには、WLSTコマンドの`importWebCenterPortalConnections`を使用します。このコマンドはWebCenter Portal 11.1.1.8リリースで導入されています。接続プロパティ・ファイルについて、詳しくは[こちら](#)を参照してください。

ステップ13：WebCenter Portalリソース・カタログへのサンプル・タスク・フローの追加

サンプル・タスク・フローをポータル・ページにドラッグ・アンド・ドロップできるようにするには、リソース・カタログにこれを追加する必要があります。

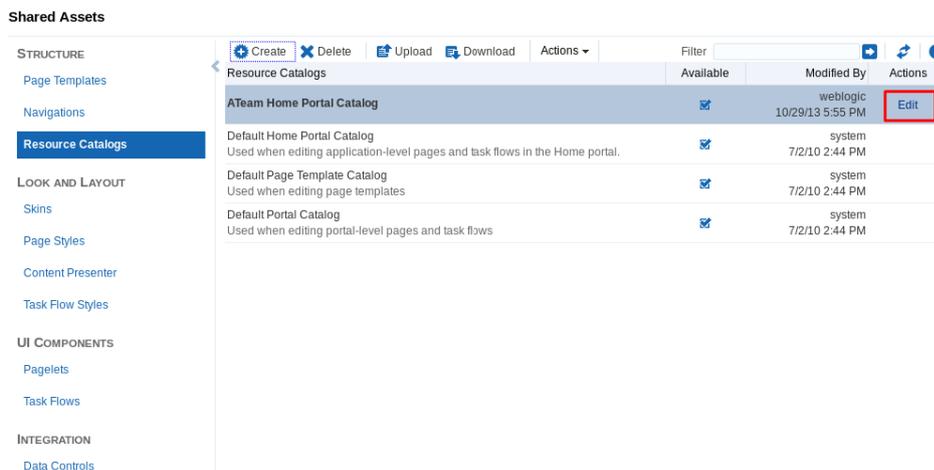
注：この例では、タスク・フローを共有リソース・カタログ（アプリケーション・レベルのリソース・カタログとも呼ばれる）に追加しますが、ポータル・レベルのリソース・カタログにタスク・フローを追加することもできます。

1. WebCenter Portalに管理者またはアプリケーション・スペシャリストとしてログインします（共有リソース・カタログの編集権限が必要です）。「Administration」→「Shared Assets」→「Resource Catalogs」を選択します。

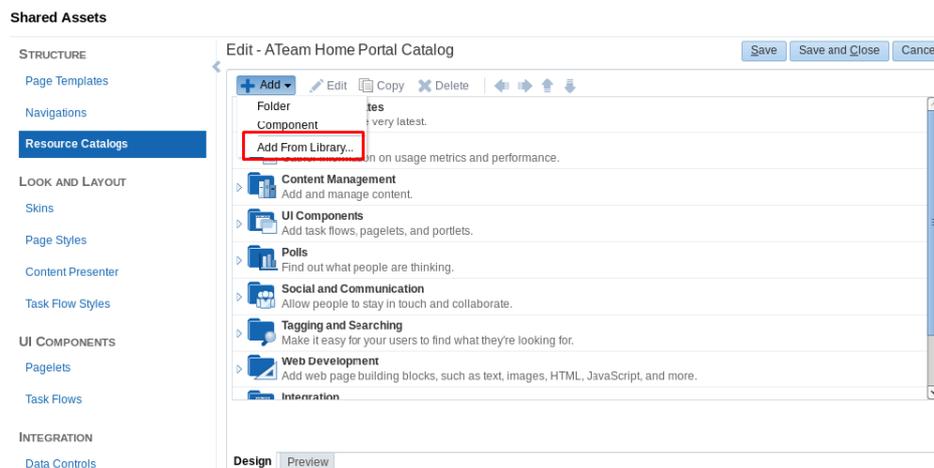
Resource Catalogs	Available	Modified By	Actions
ATeam Home Portal Catalog	<input checked="" type="checkbox"/>	weblogic 10/29/13 5:55 PM	Edit
Default Home Portal Catalog Used when editing application-level pages and task flows in the Home portal.	<input checked="" type="checkbox"/>	system 7/2/10 2:44 PM	
Default Page Template Catalog Used when editing page templates	<input checked="" type="checkbox"/>	system 7/2/10 2:44 PM	
Default Portal Catalog Used when editing portal-level pages and task flows	<input checked="" type="checkbox"/>	system 7/2/10 2:44 PM	

2. いずれかのデフォルト・カタログをコピーし、これを変更します。この例では、Default Home Portal CatalogのコピーにATeam Home Portal Catalogという名前を付けています。

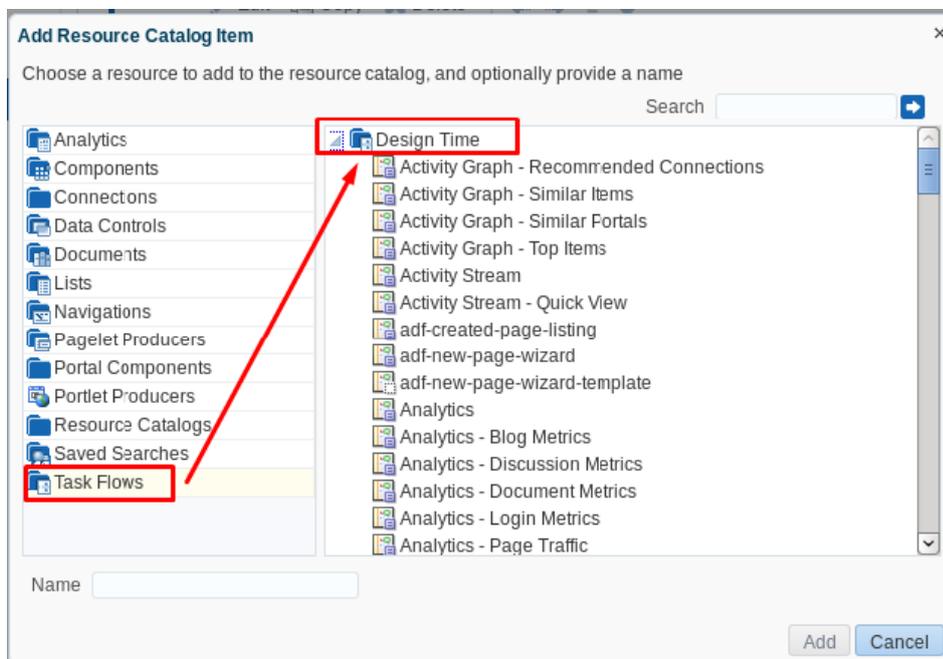
3. 「Edit」をクリックします。



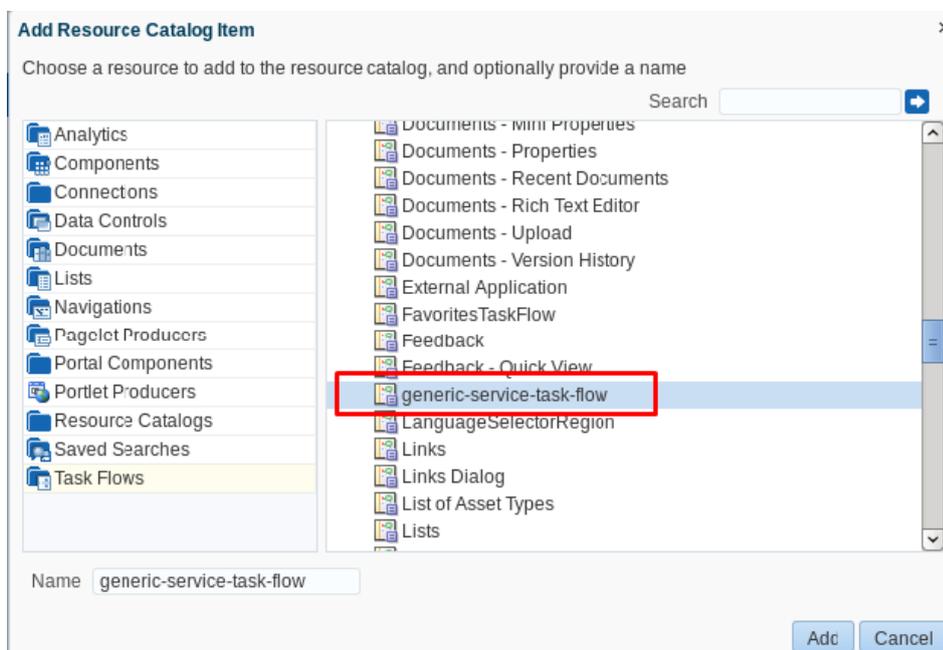
4. Edit Catalogウィンドウで「Add」をクリックし、「Add From Library」を選択します。



5. リソース・リストで「Task Flows」をクリックし、「Design Time」フォルダを開きます。

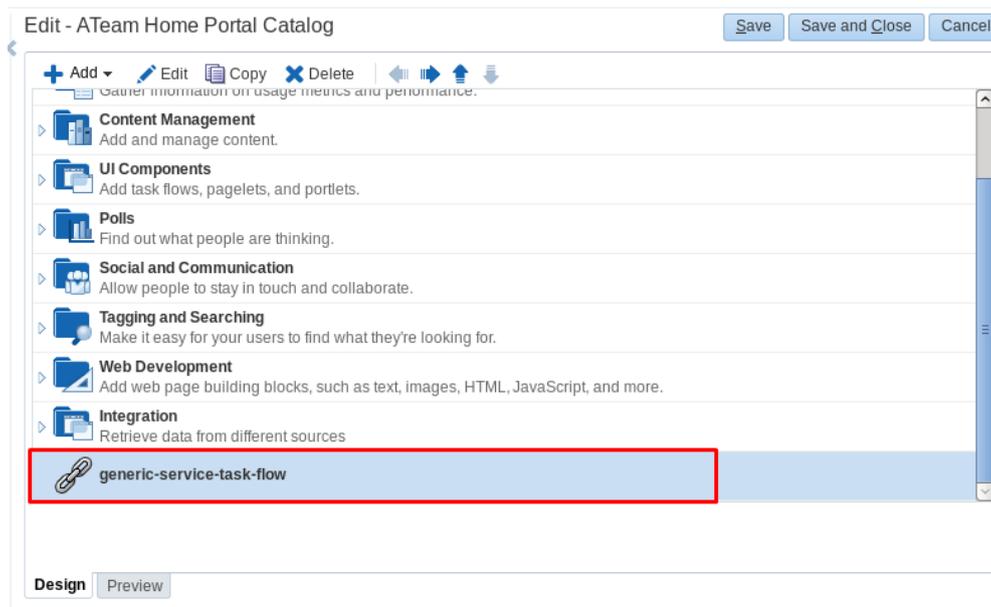


6. ツリー内のサンプル・タスク・フローを探します。タスク・フローの名前は、タスク・フローの作成時に使用したものと同じです（例：**generic-service-task-flow**）。これを選択し、「Add」をクリックしてリソース・カタログに追加します。



注：Edit Resource Catalogウィンドウでは、リソースをソートしたり、順序付けしたりできます。たとえば、新しいフォルダを作成してこの中にタスク・フローを入れることもできます。

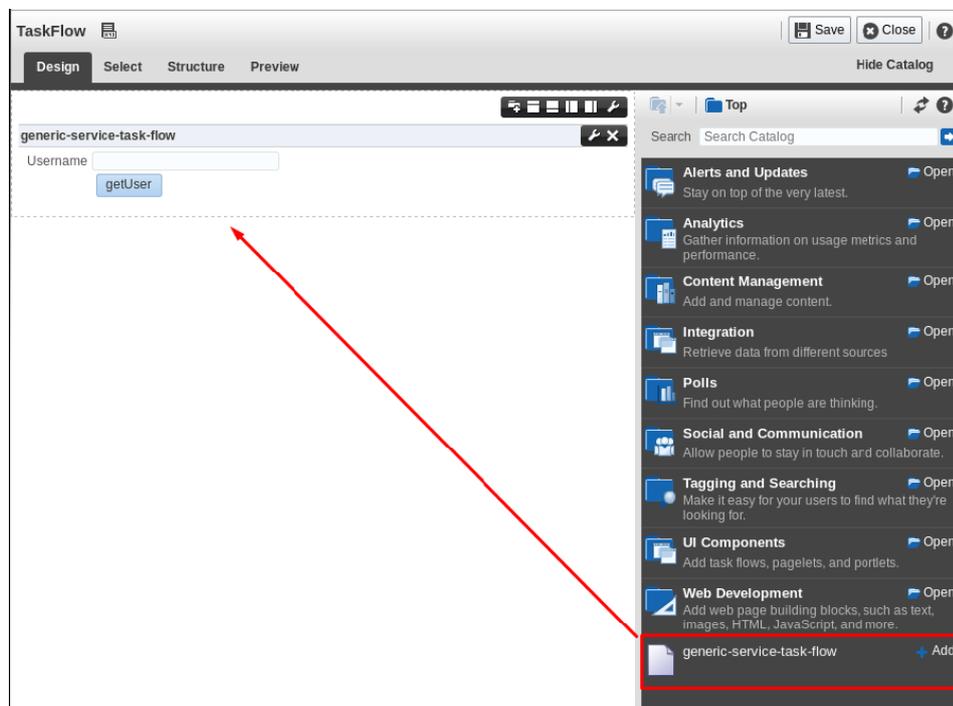
タスク・フローがリストに表示されます（上図を参照）。



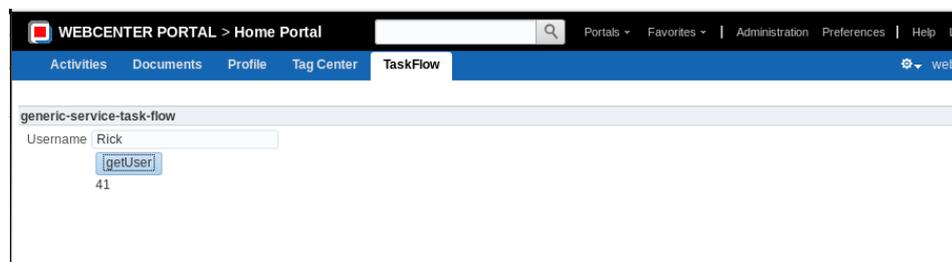
これで、同じリソース・カタログを使用しているポータル上でタスク・フローを消費できるようになりました。

注：ポータルから新しいタスク・フローにアクセスできるようにするには、このリソース・カタログをポータルのデフォルト・リソース・カタログとして指定する必要があります。

- 新しいページを作成して編集モードに切り替え、リソース・カタログからページ上にタスク・フローをドラッグ・アンド・ドロップします。



- ページを保存してタスク・フローをテストします。



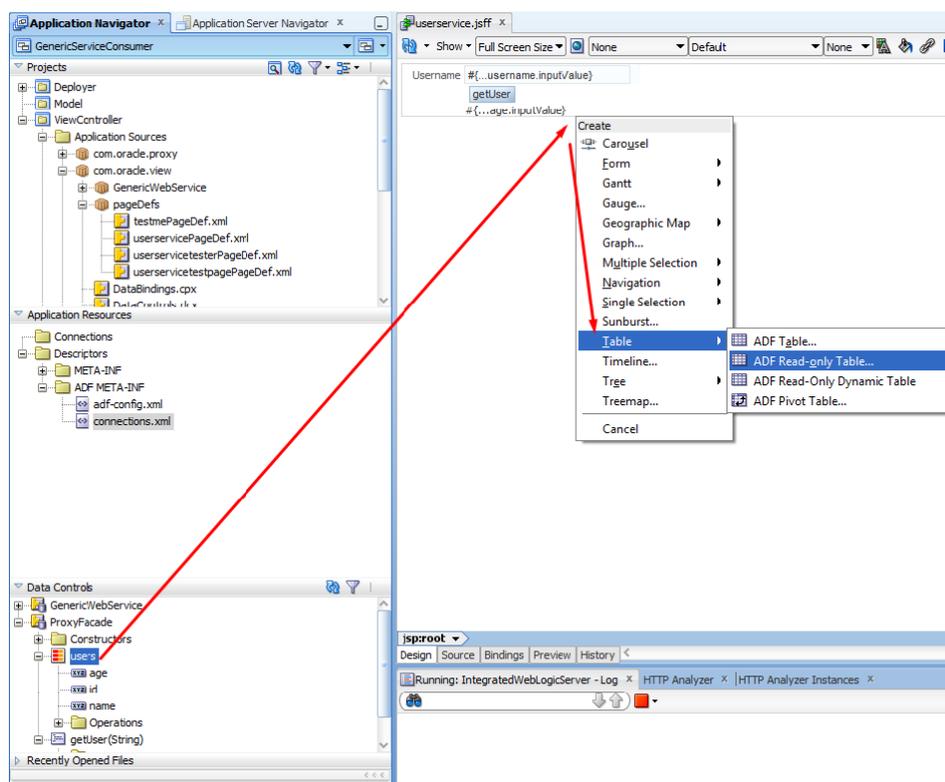
次に示すように、ページのプレビュー機能を使用してタスク・フローをテストすることもできます。



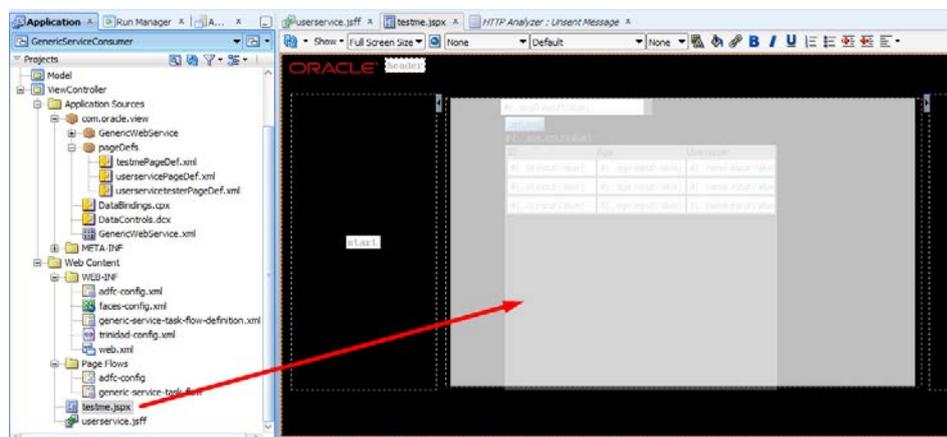
ステップ14：サンプル・タスク・フローの変更と再デプロイ

タスク・フローに変更を加えたら、Oracle ADFライブラリのJARファイルを再デプロイしてタスク・フローの更新を反映させる必要があります。これを行うには、次のステップを実行します。

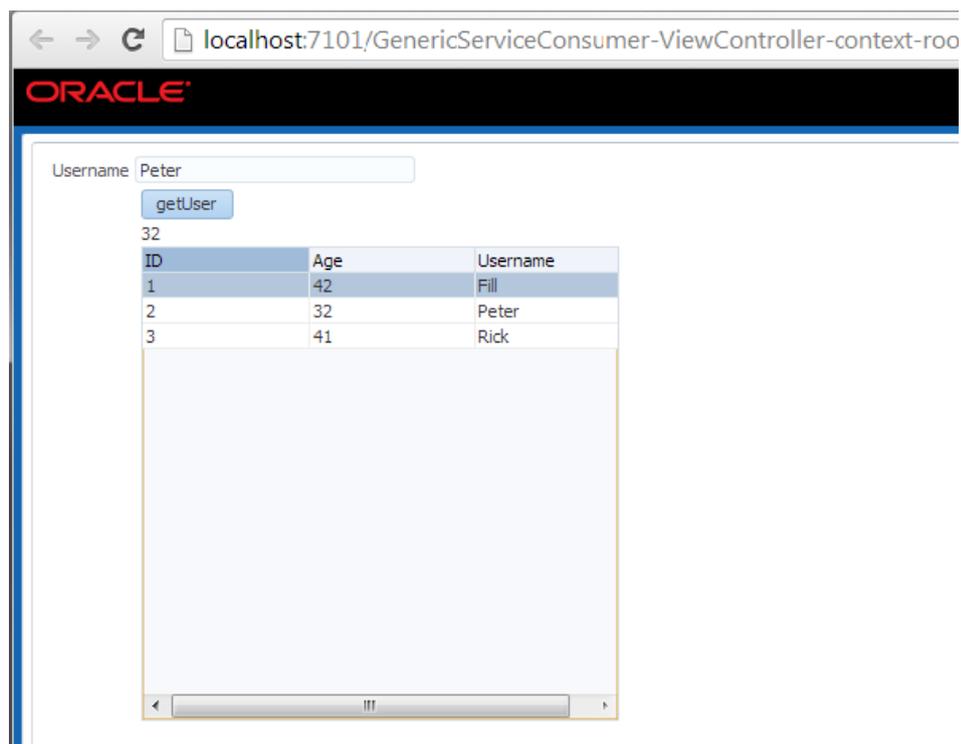
1. サンプル・タスク・フローを更新します。例として、Webサービスから取得したユーザーを表示する表を追加します。



2. 前述のとおり、テスト・ページを作成してタスク・フローにリージョンをバインドする方法で、タスク・フローをローカル・テストします。

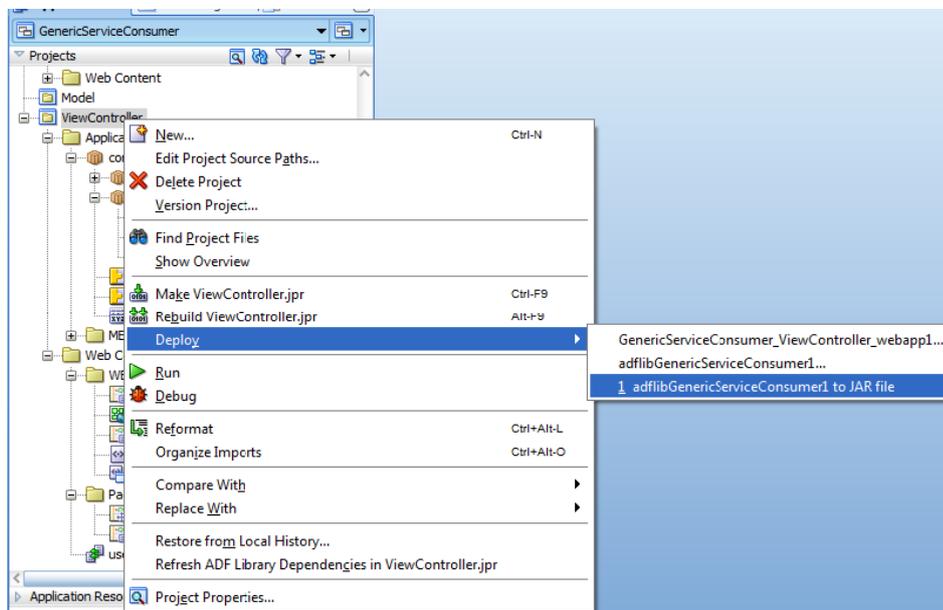


3. テスト・ページを実行してタスク・フローの変更をローカルでテストします。

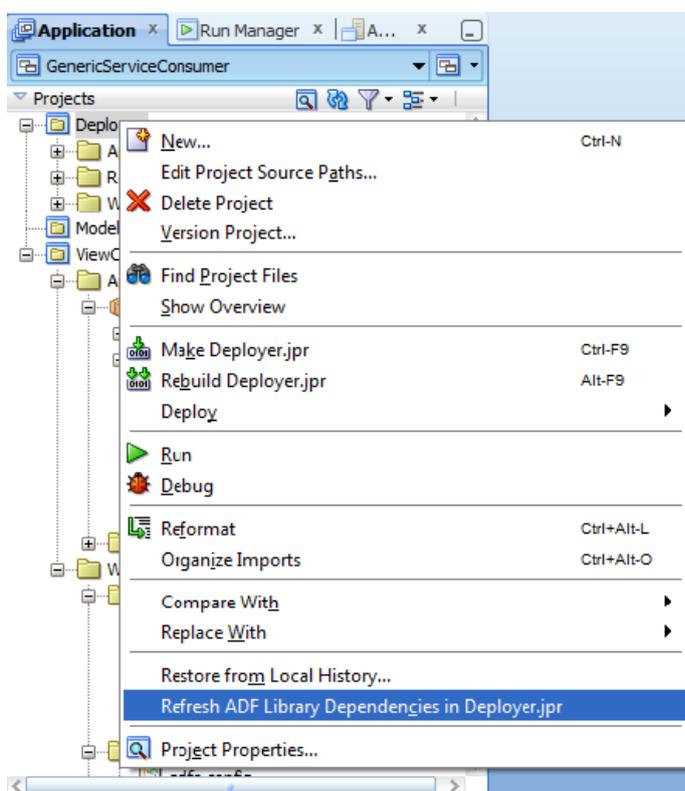


テストが正しく完了した場合は、更新したタスク・フローをポータル・サーバーにデプロイできます。

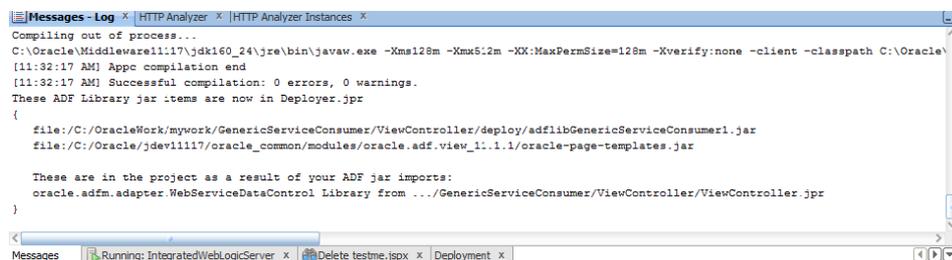
4. **ViewController**プロジェクトをOracle ADFライブラリのJARファイルにデプロイします (下図を参照)。デプロイメント・プロファイルを変更する必要はありません。



5. Deployerプロジェクトで、Oracle ADF共有ライブラリの依存関係を更新します。

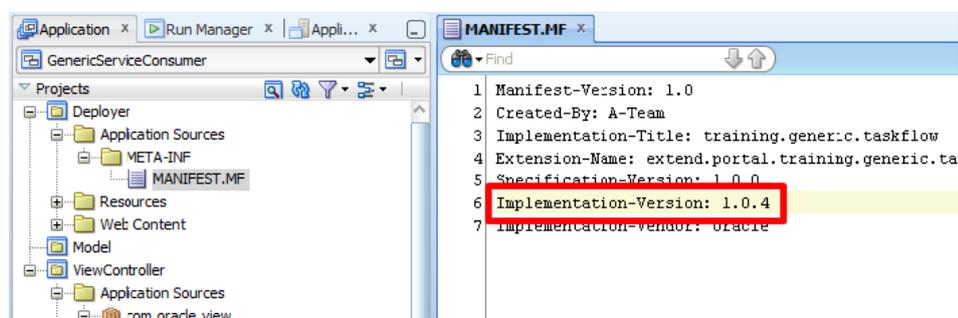


6. GenericServiceConsumerライブラリも更新されていることを確認します。



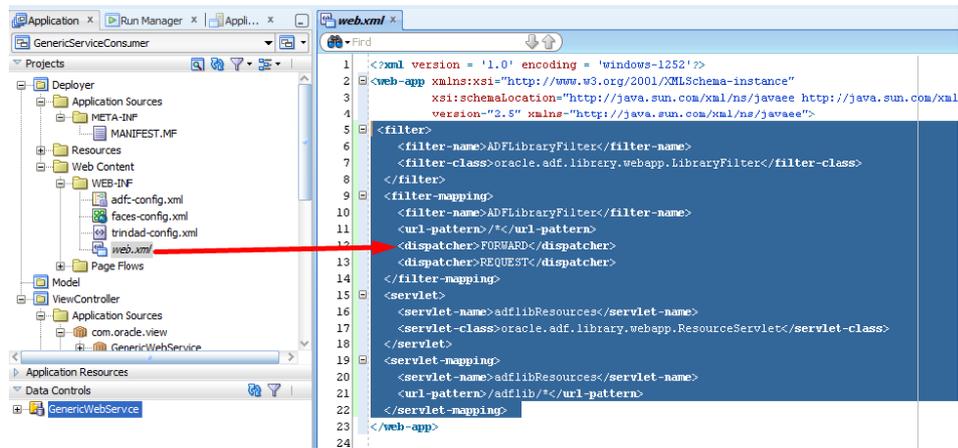
```
Messages-Log x [HTTP Analyzer x | HTTP Analyzer Instances x
Compiling out of process...
C:\Oracle\Middleware\1117\jdk160_24\jre\bin\javaw.exe -Xms128m -Xmx512m -XX:MaxPermSize=128m -Xverify:none -client -classpath C:\Oracle\
[11:32:17 AM] Appc compilation end
[11:32:17 AM] Successful compilation: 0 errors, 0 warnings.
These ADF Library jar items are now in Deployer.jar:
{
  file:/C:/OracleWork/mywork/GenericServiceConsumer/ViewController/deploy/adflibGenericServiceConsumer1.jar
  file:/C:/Oracle/jdev11117/oracle_common/modules/oracle.adf.view_11.1.1/oracle-page-templates.jar
}
These are in the project as a result of your ADF jar imports:
oracle.adfm.adapter.WebServiceDataControl Library from .../GenericServiceConsumer/ViewController/ViewController.jar
}
```

7. デプロイ時の競合を防止するため、Deployerのマニフェスト・ファイルで実装バージョンを上げます。



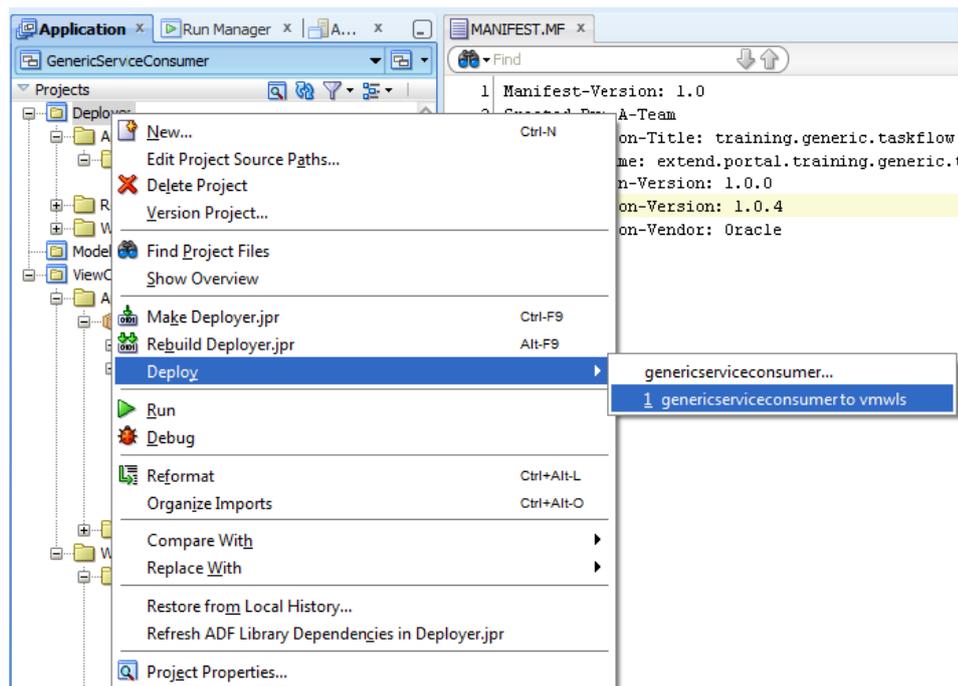
```
MANIFEST.MF x
Find
1 Manifest-Version: 1.0
2 Created-By: A-Team
3 Implementation-Title: training.generic.taskflow
4 Extension-Name: extend.portal.training.generic.ta
5 Specification-Version: 1.0.0
6 Implementation-Version: 1.0.4
7 Implementation-Vendor: Oracle
```

重要： DeployerプロジェクトのWEB.XMLが変更されていないことを確認してください。何らかの変更がある場合はこれを削除する必要があります。削除しない場合、再デプロイ後にWebCenter Portalサーバーを起動できず、エラー・メッセージが表示されます。ここで表示されるメッセージは巻末のトラブルシューティングの項に記載しています。

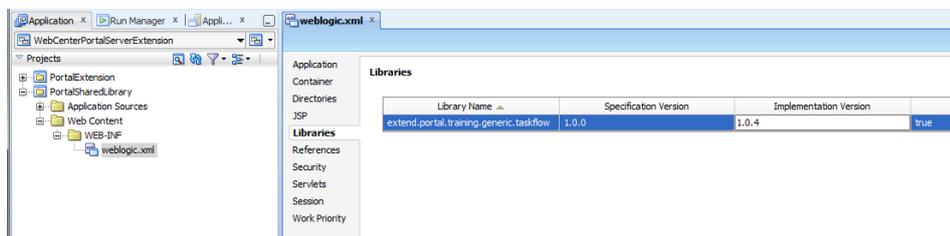


WEB.XMLファイルにコンテンツが含まれる場合はこれを削除します（上図を参照）。

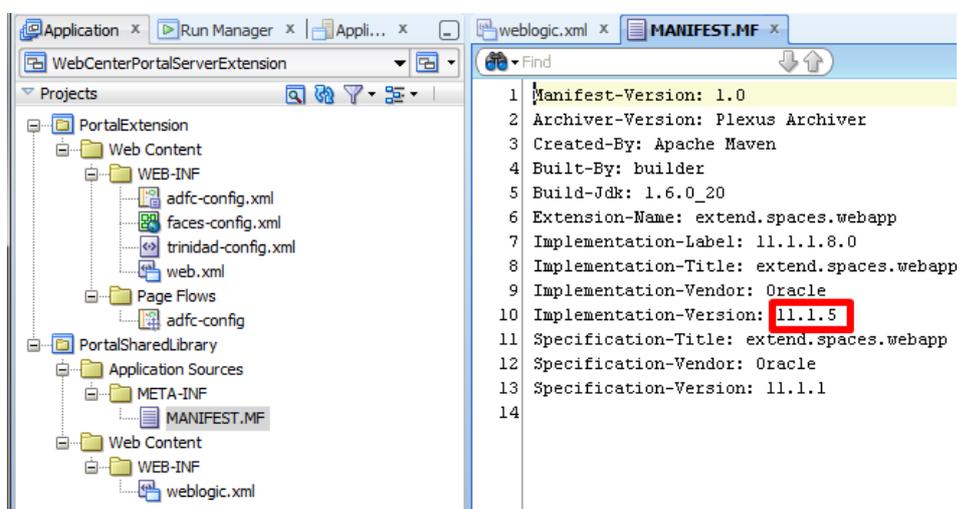
- Oracle ADF共有ライブラリのWARファイルをWebCenter Portalの管理対象サーバーまたはクラスタに再デプロイします。先ほど作成したサンプルのデプロイメント・プロファイルを使用します。



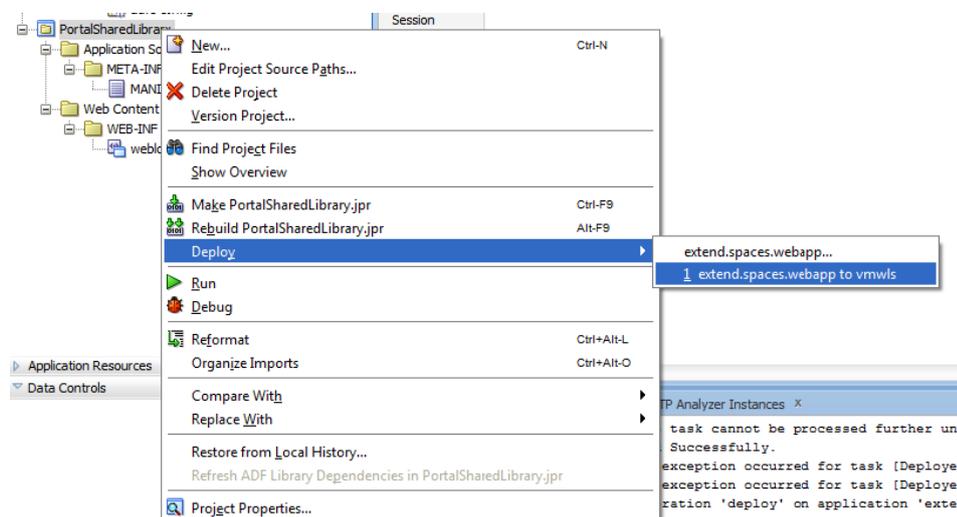
9. WebCenter Portal Server Extensionプロジェクトのweblogic.xmlで、新しい共有ライブラリ・バージョンを参照します（下図を参照）。



10. manifest.mfファイルで、WebCenter Portal Server Extensionのバージョンを上げます。



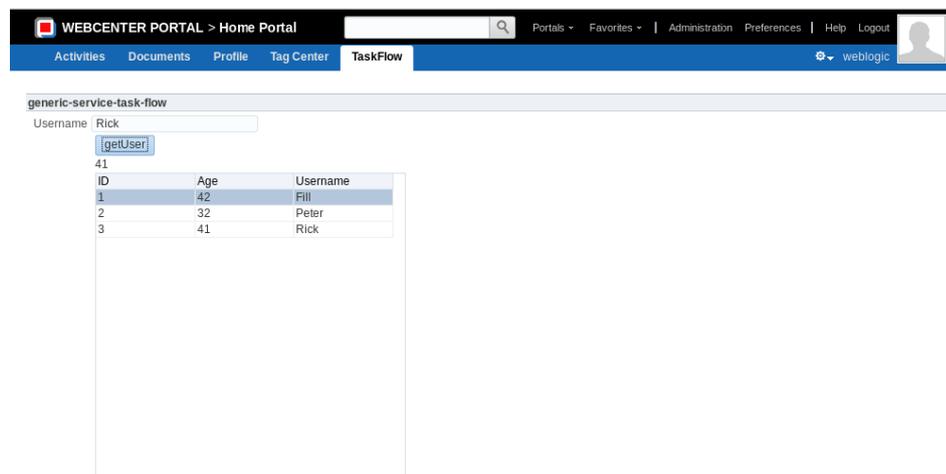
11. PortalSharedLibraryプロジェクトに対する更新を、WebCenter Portalの管理対象サーバーまたはクラスタにデプロイします。



12. WebCenter Portalの管理対象サーバーまたはクラスタを再起動して、この変更を有効にします。

注: 更新済みのタスク・フローは自動的にWebCenter Portalで有効になります。もう一度リソース・カタログにタスク・フローを追加する必要はありません。

13. サンプル・タスク・フローを含むページを開いて、更新済みの最新タスク・フローが表示されることを確認します。



トラブルシューティング

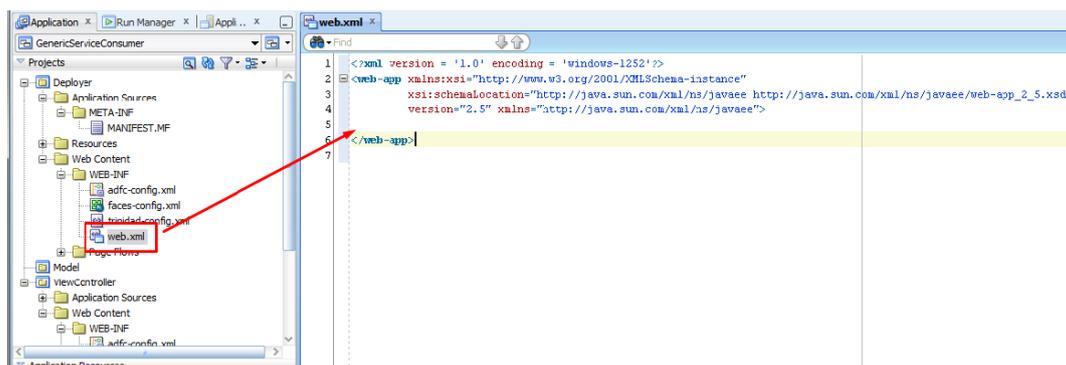
問題: Deployerプロジェクトのweb.xmlファイルに無効な構成が含まれる場合、次のエラー・メッセージが表示されます。

```
[WC_Spaces] [NOTIFICATION] [J2EE JSP-00008] [oracle.j2ee.jsp] [tid: [ACTIVE].ExecuteThread:
'1' for queue: 'weblogic.kernel.Default (self-tuning)'] [userId: anonymous] [ecid:
004uUEj1iPaFw000jzwno00023J0000Zw,0:1] [APP: webcenter#11.1.1.4.0] [URI: /webcenter/]
unable to dispatch JSP page: The following exception occurred:.[[
oracle.jsp.parse.JavaCodeException: Line # 13, oracle.jsp.parse.JspParseTagScriptlet@1d1311fe
Error: Java code in jsp source files is not allowed in ojsp.next mode.
    at oracle.jsp.parse.JspParseTagCore.createNode(JspParseTagCore.java:263)
    at oracle.jsp.parse.JspUtils.createChildNodes(JspUtils.java:2511)
    at oracle.jsp.parse.JspParseTagFile.createTree(JspParseTagFile.java:475)
    at oracle.jsp.parse.OracleJsp2Java.transformImpl(OracleJsp2Java.java:535)
    at oracle.jsp.parse.OracleJsp2Java.transform(OracleJsp2Java.java:593)
    at
oracle.jsp.runtimev2.JspPageCompiler.attemptCompilePage(JspPageCompiler.java:691)
    at oracle.jsp.runtimev2.JspPageCompiler.compileBothModes(JspPageCompiler.java:490)
    at
oracle.jsp.runtimev2.JspPageCompiler.parseAndGetTreeNode(JspPageCompiler.java:457)
    at oracle.jsp.runtimev2.JspPageInfo.compileAndLoad(JspPageInfo.java:624)
    at oracle.jsp.runtimev2.JspPageTable.compileAndServe(JspPageTable.java:645)
    at oracle.jsp.runtimev2.JspPageTable.service(JspPageTable.java:389)
    at oracle.jsp.runtimev2.JspServlet.internalService(JspServlet.java:842)
    at oracle.jsp.runtimev2.JspServlet.service(JspServlet.java:766)
    at javax.servlet.http.HttpServlet.service(HttpServlet.java:821)
    at
weblogic.servlet.internal.StubSecurityHelper$ServletServiceAction.run(StubSecurityHelper.java:
227)
    at
weblogic.servlet.internal.StubSecurityHelper.invokeServlet(StubSecurityHelper.java:125)
    at weblogic.servlet.internal.ServletStubImpl.execute(ServletStubImpl.java:301)
    at weblogic.servlet.internal.ServletStubImpl.execute(ServletStubImpl.java:185)
    at
weblogic.servlet.internal.RequestDispatcherImpl.invokeServlet(RequestDispatcherImpl.java:52
6)
    at
weblogic.servlet.internal.RequestDispatcherImpl.forward(RequestDispatcherImpl.java:253)
    at
weblogic.servlet.internal.ServletResponseImpl.sendError(ServletResponseImpl.java:731)
    at oracle.jsp.runtimev2.JspReportUtil.sendError(JspReportUtil.java:144)
    at oracle.jsp.runtimev2.JspReportUtil.reportException(JspReportUtil.java:201)
```

```
at oracle.jsp.runtimev2.JspPageTable.compileAndServe(JspPageTable.java:694)
at oracle.jsp.runtimev2.JspPageTable.service(JspPageTable.java:389)
at oracle.jsp.runtimev2.JspServlet.internalService(JspServlet.java:842)
at oracle.jsp.runtimev2.JspServlet.service(JspServlet.java:766)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:821)
at
weblogic.servlet.internal.StubSecurityHelper$ServletServiceAction.run(StubSecurityHelper.java:
227)
at
weblogic.servlet.internal.StubSecurityHelper.invokeServlet(StubSecurityHelper.java:125)
at weblogic.servlet.internal.ServletStubImpl.execute(ServletStubImpl.java:301)
at weblogic.servlet.internal.TailFilter.doFilter(TailFilter.java:27)
at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:57)
at oracle.adf.library.webapp.LibraryFilter.doFilter(LibraryFilter.java:181)
at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:57)
at oracle.wcps.client.PersonalizationFilter.doFilter(PersonalizationFilter.java:74)
at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:57)
at
oracle.webcenter.content.integration.servlets.ContentServletFilter.doFilter(ContentServletFilt
er.java:168)
at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:57)
at oracle.security.jps.ee.http.JpsAbsFilter$1.run(JpsAbsFilter.java:119)
at oracle.security.jps.util.JpsSubject.doAsPrivileged(JpsSubject.java:324)
at oracle.security.jps.ee.util.JpsPlatformUtil.runJaasMode(JpsPlatformUtil.java:460)
at oracle.security.jps.ee.http.JpsAbsFilter.runJaasMode(JpsAbsFilter.java:103)
at oracle.security.jps.ee.http.JpsAbsFilter.doFilter(JpsAbsFilter.java:171)
at oracle.security.jps.ee.http.JpsFilter.doFilter(JpsFilter.java:71)
at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:57)
at oracle.dms.servlet.DMSServletFilter.doFilter(DMSServletFilter.java:163)
at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:57)
at oracle.security.jps.ee.http.JpsAbsFilter$1.run(JpsAbsFilter.java:119)
at oracle.security.jps.util.JpsSubject.doAsPrivileged(JpsSubject.java:324)
at oracle.security.jps.ee.util.JpsPlatformUtil.runJaasMode(JpsPlatformUtil.java:460)
at oracle.security.jps.ee.http.JpsAbsFilter.runJaasMode(JpsAbsFilter.java:103)
at oracle.security.jps.ee.http.JpsAbsFilter.doFilter(JpsAbsFilter.java:171)
at oracle.security.jps.ee.http.JpsFilter.doFilter(JpsFilter.java:71)
at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:57)
at weblogic.servlet.internal.RequestEventsFilter.doFilter(RequestEventsFilter.java:27)
at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:57)
at
weblogic.servlet.internal.WebAppServletContext$ServletInvocationAction.wrapRun(WebAppSe
rvletContext.java:3730)
at
```

```
weblogic.servlet.internal.WebAppServletContext$ServletInvocationAction.run(WebAppServletC
ontext.java:3696)
    at
weblogic.security.acl.internal.AuthenticatedSubject.doAs(AuthenticatedSubject.java:321)
    at weblogic.security.service.SecurityManager.runAs(SecurityManager.java:120)
    at
weblogic.servlet.internal.WebAppServletContext.securedExecute(WebAppServletContext.java:
2273)
    at
weblogic.servlet.internal.WebAppServletContext.execute(WebAppServletContext.java:2179)
    at weblogic.servlet.internal.ServletRequestImpl.run(ServletRequestImpl.java:1490)
    at weblogic.work.ExecuteThread.execute(ExecuteThread.java:256)
    at weblogic.work.ExecuteThread.run(ExecuteThread.java:221)
]]
```

解決方法：DeployerのWEB.XMLファイルから無効な構成を削除します。ファイルが次のようになっていれば問題ありません。





Oracle WebCenter Portal 11g Release 1
バージョン11.1.1.8.0を使用したタスク・
フロー開発ライフ・サイクル

2013年12月

著者：Lyudmil Pelov

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

海外からのお問い合わせ窓口：
電話：+1.650.506.7000
ファクシミリ：+1.650.506.7200

www.oracle.com



Oracle is committed to developing practices and products that help protect the environment

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

本文書は情報提供のみを目的として提供されており、ここに記載される内容は予告なく変更されることがあります。本文書は、その内容に誤りがないことを保証するものではなく、また、口頭による明示的保証や法律による黙示的保証を含め、商品性ないし特定目的適合性に関する黙示的保証および条件などのいかなる保証および条件も提供するものではありません。オラクル社は本文書に関するいかなる法的責任も明確に否認し、本文書によって直接的または間接的に確立される契約義務はないものとします。本文書はオラクル社の書面による許可を前もって得ることなく、いかなる目的のためにも、電子または印刷を含むいかなる形式や手段によっても再作成または送信することはできません。

OracleおよびJavaはOracleおよびその子会社、関連会社の登録商標です。その他の名称はそれぞれの会社の商標です。

Hardware and Software, Engineered to Work Together