WCF 自習書

# AWTS サンプル セットアップガイド

developer & platform evangelism

制作者: エディフィストラーニング株式会社

# 目次

**はじめに** ..... **4**

**セットアップ方法** ..... **5**

環境の前提条件 ..... 5

セットアップ手順 ..... 5

接続文字列の変更 ..... 7

証明書のインストール ..... 11

**サンプル操作方法ウォークスルー** ..... **13**

社内向けシステム ..... 15

顧客向け Web サイト ..... 23

# はじめに

このセットアップガイドでは、第 2 部、および、第 3 部で使用するサンプルプログラム「Adventure Works Transport Services」について、そのセットアップ方法を示しています。このセットアップ手順を行うには、予めサンプルを入手する必要があります。

また、セットアップを行うには、以下の基本的な操作知識が必要になります。

- SQL Server Management Studio または、sqlcmd コマンドなどを使用して、SQL Server データベースをアタッチすることができる。（参考として、このあとのセットアップ手順では sqlcmd コマンドを使用したアタッチ方法を掲載しています。）
- Visual Studio 2010 のプロジェクトを開き、テキストファイルの編集や、プロジェクトのプロパティを変更できる。また、ソリューションやプロジェクトなど、Visual Studio の基本的な用語が理解できる。

# セットアップ方法

この章では、当サンプルプログラム「Adventure Works Transport Services」のセットアップ手順について説明します。(以降、サンプルプログラムの名称は「AWTS サンプルプログラム」と表記します。) AWTS サンプルプログラム自体のセットアップは、10 分程度で完了しますが、いくつか、開発ツールのインストールなどの準備も必要です。

## 環境の前提条件

AWTS サンプルプログラムでは、以下が動作する環境を予め準備しておく必要があります。

- 次の 2 項目について動作可能なオペレーティングシステム (Windows Server 2008 R2 など)
- SQL Server 2008 R2 のいずれかのエディション
- Visual Studio 2010 のいずれかのエディション

対象のオペレーティングシステムについては、SQL Server 2008 R2、Visual Studio 2010、および、.NET Framework 4 が動作できるのであれば、理論的には問題はありませんが、当サンプルの作成過程においては、Windows Server 2008 R2 で検証しています。

使用する SQL Server 2008 R2 も特にエディションは問いませんが、当サンプルの検証では、SQL Server 2008 R2 Enterprise Edition を使用しています。また、既定オプションのインストールで構いません。インストール場所やインスタンス名も、特に制約はありませんが、当サンプルで使用している接続文字列を書き換えずに済むようにするため、SQL Server をインストールする際には、サンプルのインストール先と同じローカルマシン上に、既定のインスタンス名でインストールすると便利です。接続文字列の変更方法は、このあとのセットアップ手順にも記載されています。

なお、SQL Server Express を既定構成でインストールした場合、インスタンス名は「sqlexpress」となります。また、接続文字列を変更しない場合でも、このサンプルを理解するために、セットアップ手順について一通りお読みになることをお奨めします。

また、WCF サービスの署名や暗号化のために、新規作成した証明書を以下のストアに上書きインストールするので、既存の同名証明書が上書きされても問題ない環境である必要があります。

- 「証明書(ローカル コンピューター)」の「個人」の証明書ストアの
  「localhost」(CN=localhost) という名前の証明書
- 「証明書 - 現在のユーザー」の 「信頼されたユーザー」の証明書ストアの
  「localhost」(CN=localhost) という名前の証明書

## セットアップ手順

AWTS サンプルプログラムを利用できるようにするには、次の 4 つの作業を行う必要があります。続く説明も参考にして、これら 4 つの作業を行ってください。

① 当サンプルのフォルダー「SamplesWCF」を適当な場所にコピーする
② db ファイル内のデータベースファイルをアタッチする
③ 必要に応じて、サンプルプログラムの接続文字列を変更する
④ WCF サービスの署名や暗号化で使用する証明書をインストールする

まず、①のサンプル プログラムのフォルダーについては、階層構造を維持したまま、フォルダー全体をローカルディスク内の任意の場所にコピーしてください。（これ以降の説明では、C:¥SamplesWCF というパスになるようにコピーした上で、サンプル プログラムを使用しています。）

②のデータベースについては、環境の前提として、Visual Studio 2010 環境から実行されるサンプルプログラムが、Windows 認証を使用して SQL Server にログインでき、適切な権限のもとデータベースを操作できる必要があります。それを簡単に行うには、Visual Studio を使用するユーザーアカウントが、対象となる SQL Server に対して、管理者としてログインできるようにすることです。これを実現する一番簡単な方法は、Windows OS にログオン後に同一のデスクトップ環境から、Visual Studio と SQL Server を既定手順でインストールすることです。

②の具体的なデータベースは、AWTS サンプル プログラムに含まれる次のデータベースファイルです。SQL Server Management Studio や sqlcmd コマンドなどを使用して、これらのデータベースファイルをアタッチしてください。参考までに、sqlcmd コマンドを使用して、データベースファイルをアタッチする方法も示しておきます。

- AdwkMileage データベース ---- AdwkMileages.mdf、AdwkMileages_log.ldf
- AdwkReservations データベース ---- AdwkReservations.mdf、AdwkReservations_log.ldf
- aspnetdb データベース ---- aspnetdb.mdf、aspnetdb_log.ldf
- ExternalPoints データベース ---- ExternalPoints.mdf、ExternalPoints_log.ldf

（参考）sqlcmd を用いた場合の AdwkMileage データベースをアタッチする場合の手順

1. 「Visual Studio コマンド プロンプト」など、sqlcmd を実行できる環境のコマンド プロンプトを開きます。
2. コマンド プロンプト上で、次のコマンドを実行し、SQL Server にログインします。（オプション「-E」は Windows 認証を意味し、オプション「-S .」（末尾は半角スペース、ドット）はローカルマシンの既定インスタンスを意味します。既定構成の SQL Server Express のインスタンスを使用する場合は、「-S .¥SQLEXPRESS」となります。なお、「E」と「S」は大文字です。）

```
sqlcmd -E -S .
```

3. sqlcmd が起動したら、次のコマンドを順に実行して、データベースをアタッチします。

```
1> use master;
2> go
1> sp_attach_db 'AdwkMileages', 'C:¥SamplesWCF¥db¥AdwkMileages.mdf';
2> go
```

上記の sp_attach_db のプロシージャの引数として、データベース名のほか、既存のデータベースファイル（.mdf ファイル）のパスを指定します。

4. アタッチが済んだら、次のコマンドを実行し、sqlcmd を終了します。

```
1> exit
```

③については、SQL Server のインストール場所が、サンプルプログラムの存在するローカルマシンで、かつ、既定のインスタンス名ならば、接続文字列の変更の必要はありません。もし、そうでなければ、次の手順に沿って、接続文字列を変更してください。

## 接続文字列の変更

セットアップ手順の冒頭に挙げた③の接続文字列の情報は、当サンプルの以下の各プロジェクトに含まれます。サンプルの既定構成の接続を使用しない場合は、この後の説明も参考にして、接続文字列を修正してください。

- AdwkMService フォルダーの AdwkMService ソリューションに含まれる以下のプロジェクト
  - AdwkMService プロジェクト（※1）---- App.config を直接編集
  - AdwkMDac プロジェクト（※2）---- プロジェクト デザイナーで編集
  - DummyExternalService プロジェクト（※2）---- プロジェクト デザイナーで編集
- AdwkRService フォルダーの AdwkRService ソリューションに含まれる以下のプロジェクト
  - AdwkRService プロジェクト（※3）---- App.config を直接編集
  - AdwkRDac プロジェクト（※3）---- App.config を直接編集
- AdwkWebApp フォルダーの AdwkWebApp ソリューションに含まれる以下のプロジェクト
  - AdwkWebApp プロジェクト（※4）---- Web.config を直接編集
  - AdwkRDac プロジェクト（※3）---- App.config を直接編集

上記のプロジェクトの中で「※1」と「※3」は、App.config ファイルをテキストエディターで開いて、接続文字列を編集する必要があります。以下に「AdwkMService」の場合の手順を示します。

1. AdwkMService フォルダーの中にある AdwkMService.sln という名前のソリューションを開きます。
2. AdwkMService プロジェクト配下にある App.config ファイルをテキストエディターで開き、次の接続文字列を見つけて、インスタンス名の部分(黄色の部分)を、実際に使用する名前に変更します。(紙面の都合で、途中改行しています。)

```
<connectionStrings>
    <add name="Adwk.Mileage.Dac.Properties.Settings.AdwkMileagesConString"
        connectionString="Data Source=.;Initial Catalog=AdwkMileages;
                          Integrated Security=True"
        providerName="System.Data.SqlClient" />
</connectionStrings>
```

上記の例のドット「.」は、ローカルマシンの既定インスタンスを意味します。もし、他のサーバーへアクセスするならば、「サーバー名¥インスタンス名」となります。

3. 設定が済んだら、[ファイル] メニューの [すべてを保存] などをクリックして、変更内容を反映させます。
4. 他の作業で AdwkMService.sln ソリューションは使用するので、そのまま開いておきます。

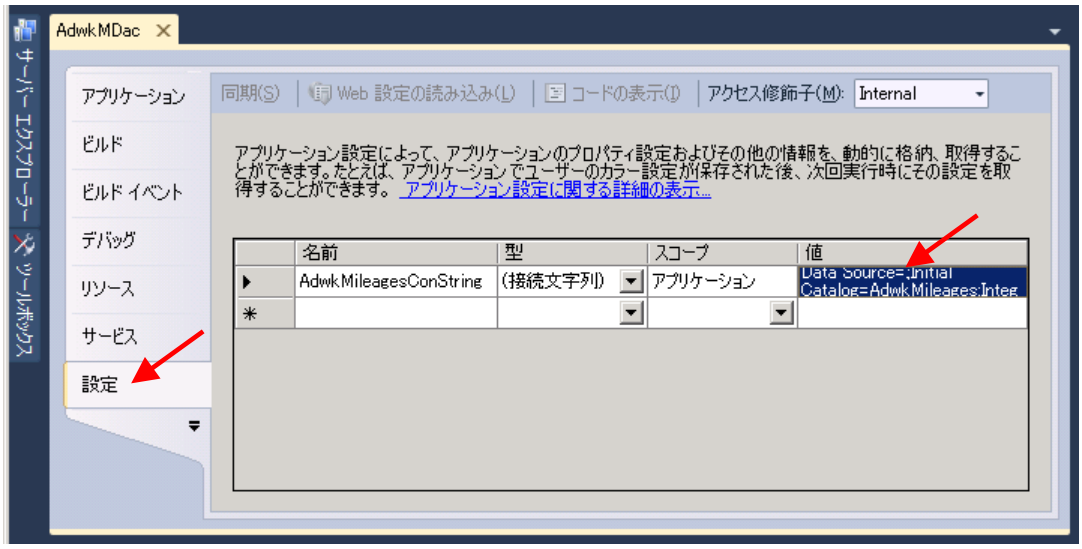
なお、「※3」のプロジェクト（AdwkRService プロジェクト 1 つと、AdwkRDac プロジェクト 2 つ）も、前述と同様に App.config ファイルをテキストエディターで開いて修正しますが、修正すべき文字列が Entity Framework 向けの文字列であるので、少し異なります。以下の部分（黄色の部分）を同様に正しいインスタンス名に変更してください。(紙面の都合で、途中改行しています。)

```
<connectionStrings>
  <add name="AdwkReservationsEntities"
     connectionString="metadata=res://*/AdwkReservations.csdl|
     res://*/AdwkReservations.ssdl|res://*/AdwkReservations.msl;
     provider=System.Data.SqlClient;provider connection string=&quot;
     Data Source=.;Initial Catalog=AdwkReservations;IntegratedSecurity=True;
     MultipleActiveResultSets=True&quot;"
     providerName="System.Data.EntityClient" />
</connectionStrings>
```

次に、「※2」のプロジェクトも同様に変更しますが、テキストエディターで App.config ファイルを直接編集せずに、この後に示す手順で行うことをお奨めします。というのは、「※2」に該当するプロジェクト（AdwkMDac プロジェクト、および、DummyExternalService プロジェクト）では、接続文字列について、以下の操作手順に連動して、ソースコード上に既定値として接続文字列が自動的にコードコーディングされ埋め込まれているからです。(実際のところ、実行時に構成ファイルが存在すれば、この既定値の文字列は使用されません。)

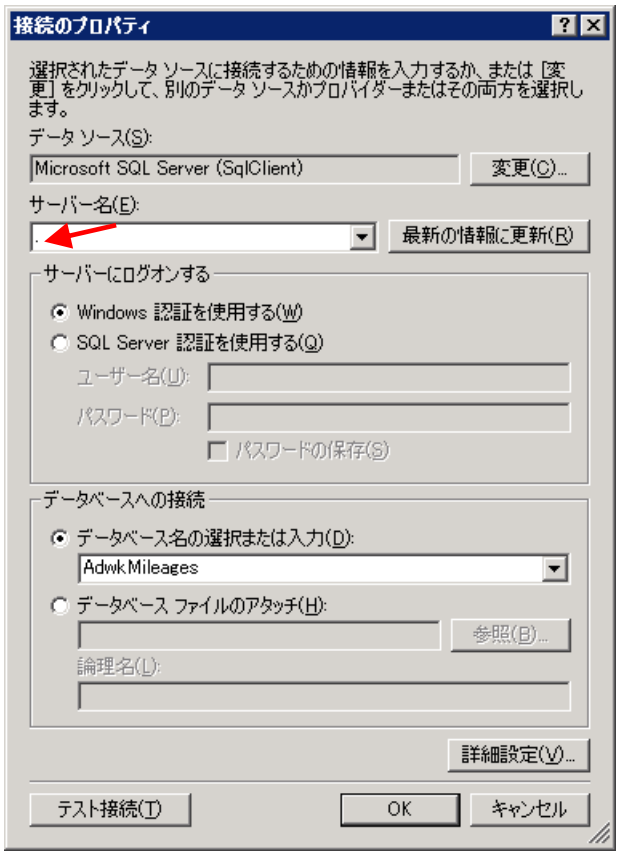
5. 現在開いている AdwkMService ソリューションの AdwkMDac プロジェクトについて、プロジェクトデザイナーを開きます。(ソリューションエクスプローラー上の AdwkMDac プロジェクトを右クリックして、[プロパティ] をクリックします。)
6. 次のようにプロジェクトデザイナーが開いたら、[設定] タブをクリックします。グリッド形式で接続文字列がリストアップされるので、右端の列にある接続文字列をクリックし選択し、[...] ボタンを押して、接続文字列の編集画面を開きます。

図 1. AdwkMDac プロジェクトのプロジェクト デザイナー




7. 次図のように、[接続のプロパティ] ダイアログボックスが開くので、インスタンス名を実際に使用する名前に変更します。(ここでも、「サーバー名¥インスタンス名」形式で入力します。ローカルマシンの既定インスタンスの場合は、ドット「.」です。)

図 2. 接続のプロパティ




8. [OK] ボタンをクリックして、上記のダイアログボックスを閉じた後、プロジェクトデザイナーに戻ります。

9. 設定が済んだら、[ファイル] メニューの [すべてを保存] などをクリックして、変更内容を反映させます。
10. 同様に、AdwkMService ソリューション内の DummyExternalServices プロジェクトについても、接続文字列のサーバー名を変更します。
11. 念のため、[ソリューション] メニューの [ソリューションのビルド] をクリックして、ソリューション全体をビルドし、複数のプロジェクト間で適切にファイルが参照できるようにしておきます。
12. ソリューションを閉じます。

最後に、「※4」の AdwkWebApp プロジェクトについて、Web.config ファイル内の接続文字列を次の要領で変更します。

13. AdwkWebApp フォルダーの中にある AdwkWebApp.sln という名前のソリューションを開きます。
14. AdwkWebApp プロジェクトの中の web.config ファイルをテキストエディターで開き、次の 3 つの接続文字列を見つけ、実際に使用するインスタンス名に変更します。(紙面の都合で、途中改行しています。)

```
<connectionStrings>
  <!-- マイレージ データベース (staffページのサンプル専用) -->
  <add name="AdwkDac.Properties.Settings.AdwkMileagesConnectionString"
      connectionString="Data Source=.;Initial Catalog=AdwkMileages;
      Integrated Security=True"
      providerName="System.Data.SqlClient" />
  <!-- ASP.NETがメンバーシップなどで利用するデータベース -->
  <remove name="LocalSqlServer"/>
  <add name="LocalSqlServer"
       connectionString="data source=.;Integrated Security=SSPI;
       Initial Catalog=aspnetdb"
       providerName="System.Data.SqlClient"/>
  <!-- 予約管理用のデータベース -->
  <add name="AdwkReservationsEntities"
       connectionString="metadata=res://*/AdwkReservations.csdl|
       res://*/AdwkReservations.ssdl|res://*/AdwkReservations.msl;
       provider=System.Data.SqlClient;provider connection string=&quot;
       Data Source=.;Initial Catalog=AdwkReservations;
       Integrated Security=True;MultipleActiveResultSets=True&quot;"
       providerName="System.Data.EntityClient" />
</connectionStrings>
```

**註:**このうち 2 番目の文字列は、ASP.NET フォーム認証用のユーザー情報が含まれる、aspnetdb データベースへ接続するために使用します。

15. 設定が済んだら、[ファイル] メニューの [すべてを保存] などをクリックして、変更内容を反映させます。
16. ソリューションを閉じておきます。

## 証明書のインストール

セットアップ手順の冒頭に挙げた④について、証明書のインストール方法を示します。ここでは、予め用意したバッチファイルを使用して、証明書を作成しインストールします。

1. 「Visual Studio コマンド プロンプト」など、certmgr.exe や makecert.exe を実行できる環境のコマンド プロンプトを開きます。（Administorator 以外のユーザーでログインしている場合は管理者権限付きでコマンドプロンプトを開きます）
2. サンプルに含まれる証明書インストール用のバッチファイルが存在する「_setup」サブフォルダーに移動します。（以下は、サンプルのフォルダーが C:¥SamplesWCF の場合の例）

```
C: [Enter]
CD C:¥SamplesWCF¥_setup [Enter]
```

3. 以下のコマンドを実行し、証明書を作成しインストールします。

```
setup_certs.bat [Enter]
```

4. すると、以下の確認メッセージが表示され、既存の同名の証明書が上書きされる旨が表示されます。このサンプルでは、確認メッセージに表示された 2 つの証明書が必要です。既存のものがあって、上書きされて問題ない場合は、いずれかのキーを押して、処理を進めてください。

```
.
以下の既存の証明書を削除した上で再作成します。
  (1)「証明書(ローカル コンピューター)」の「個人」の証明書ストアの
     「localhost」(CN=localhost) という名前の証明書
  (2)「証明書 - 現在のユーザー」の 「信頼されたユーザー」の証明書ストアの
     「localhost」(CN=localhost) という名前の証明書
中止する場合は、[Ctrl]+[C]キーを押したのち、
「バッチ ジョブを終了しますか (Y/N)?」という問いに[Y]、[Enter]を押してください。
.
続行するには何かキーを押してください . . .
```

5. バッチ処理が開始され、次のようにメッセージが表示されることを確認します。

```
... 削除します。
Error: Failed to find a certificate to delete        ←[1]
CertMgr Failed
Error: Failed to find a certificate to delete        ←[2]
CertMgr Failed
... 作成します。
Succeeded                          ←[3]
CertMgr Succeeded                  ←[4]
... バッチジョンが終了しました。
.
続行するには何かキーを押してください . . .
```

上記の[1]と[2]は、既存の証明書の削除を試みた際に、見つからなかった場合のメッセージです。削除に失敗（Failed to ...）と表示されますが、初回であれば、このように表示されて問題ありません。

なお、このバッチファイルは何回も実行でき、2 回目からは既存のものが正常に削除されるので、次のように表示されます。

```
.
続行するには何かキーを押してください . . .
... 削除します。
CertMgr Succeeded
CertMgr Succeeded
... 作成します。
Succeeded
CertMgr Succeeded
... バッチジョンが終了しました。
.
```

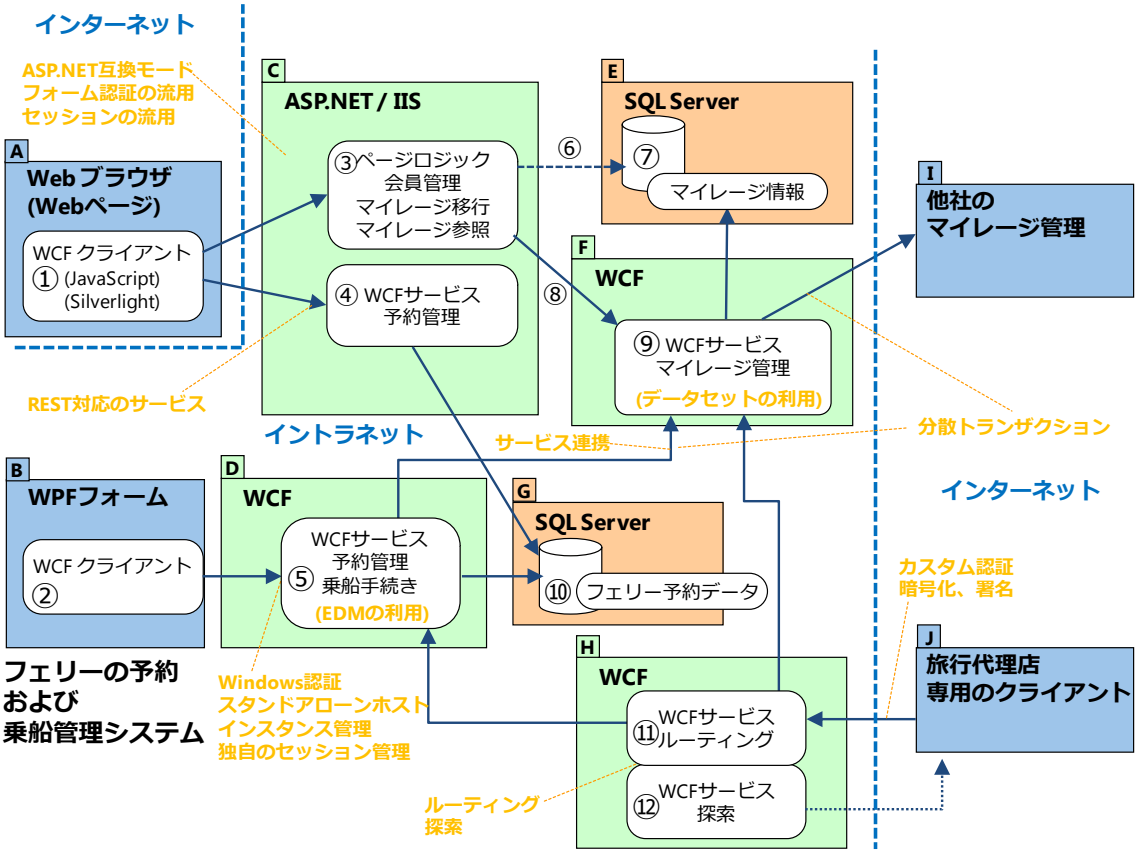
これで証明書もインストールされ、セットアップ手順の冒頭に表示された、①から④までの作業が終了しました。

# サンプル操作方法ウォークスルー

この章では、セットアップを行った AWTS サンプルプログラムが動作するか確認するため、また、サンプルの操作に慣れて頂くため、操作手順を示します。サンプルプログラムにおいて考えうる操作手順すべてを扱うわけではありませんが、ここで基本的な操作手順に慣れておけば、第 2 部以降の本編を読みながらの動作確認も行いやすくなるでしょう。

ここでのサンプルプログラムは、フェリーやバスなどの交通機関を運営する架空の企業である Adventure Works Transport Services において、フェリーの予約管理や乗船手続き、関連するマイレージの処理を行うシステムです。顧客向けのインターネットに公開される ASP.NET Web サイトと、WPF アプリケーションをクライアントとするイントラネット向けの多層アプリケーションから構成されています。以下にシステムの全体図を掲載します（この図を含め、システム構成や実装に関する説明は、第 2 部以降で改めて扱っています）。

図 3. AWTS サンプル プログラムのシステム構成




このサンプルは、大きく分けて 2 種類あります。

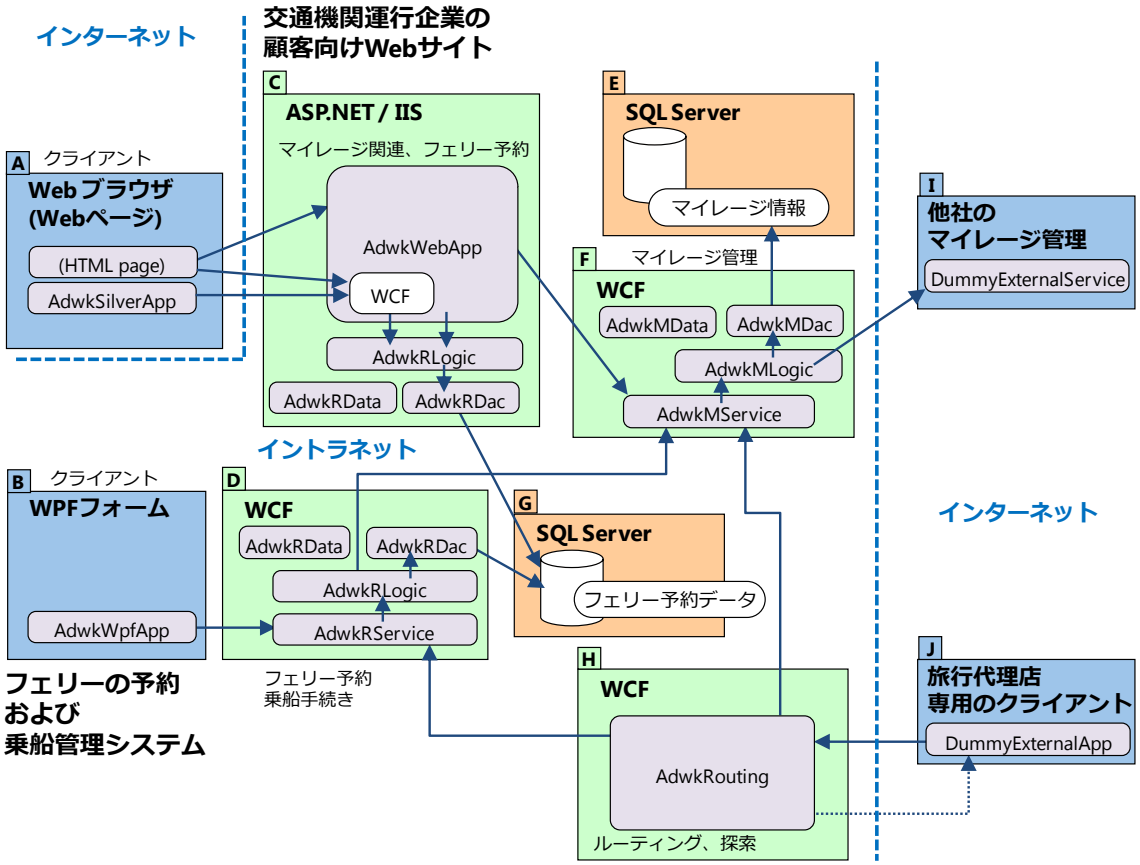
1 つは、図の上半分の[A]、[C]、[F]などのブロックを使用する、Web ベースのアプリケーションです。これは、顧客がインターネット経由で利用するフェリー予約システムです。この Web アプリケーションでは、フェリーの予約のほか、フェリー乗船によって獲得できるマイレージの管理も行っています。[F]のブロックは、マイレージ管理を行う WCF サービスであり、[I]のブロックにある社外のバー

トナー企業のマイレージ管理システム（ポイント管理システム）と、WCF サービスを介して連携しています。

もう 1 つは、図の下半分の[B]、[D]、[G]などからなるシステムであり、架空の Adventure Works Transport Services の企業内において、社員が社内 LAN 上で使用するシステムです。顧客から電話などで受け付けたフェリー予約の登録のほか、乗船手続きやマイレージの加算などができます。また、[J]のブロックにある外部パートナー企業（旅行代理店）からも、WCF サービスを介して、フェリーの予約を行うことができます。

なお、このシステムを Visual Studio のプロジェクト単位に分割して、依存関係を示すと次図のようになります。

**図 4. AWTS サンプル プログラムのプロジェクト構成**



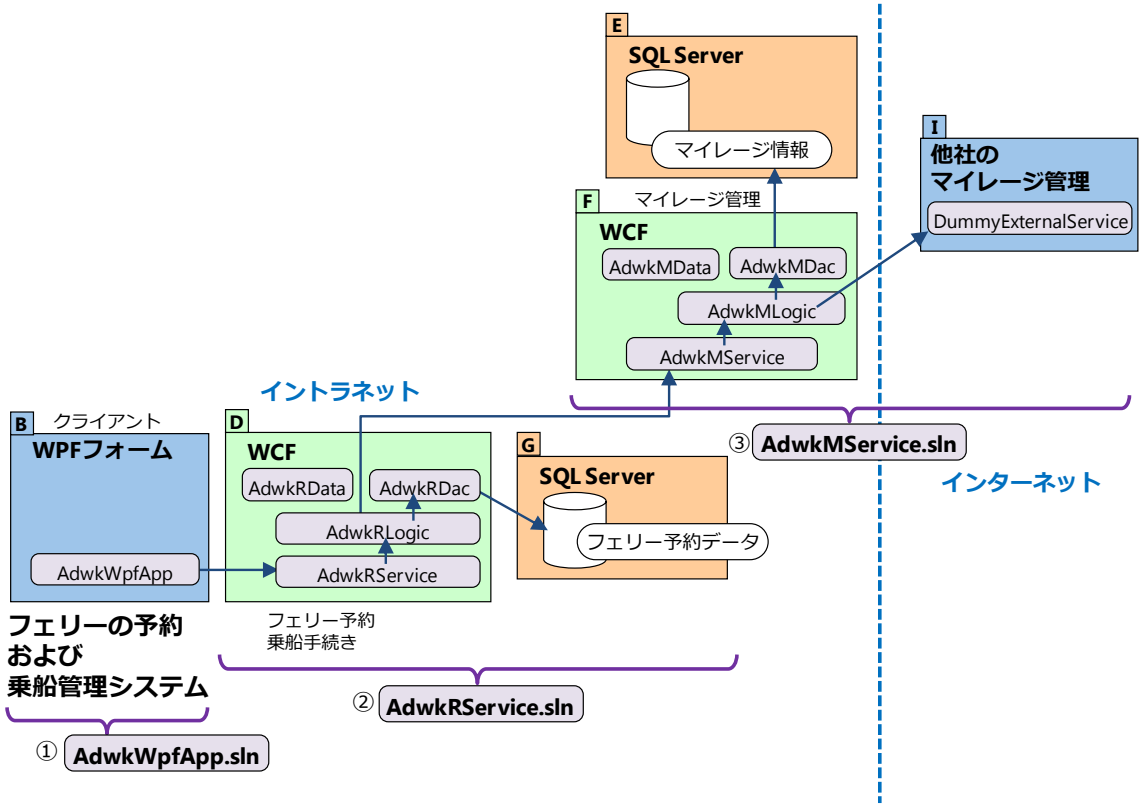

**註:** 図中のさらに詳しい解説は、第 2 部で取り上げています。

このあとは、大きく分けた 2 つのシステム構成について、それぞれの主な操作を紹介します。本資料では、図中の下半分の社内システム（[B]、[D]、[G]など）を「社内向けシステム」と呼ぶことにします。また、上半分の顧客向けの Web ベースのシステム（[A]、[C]、[F]など）を「顧客向け Web サイト」と呼ぶことにします。

## 社内向けシステム

このシステムでは、リッチクライアントの 1 つである WPF アプリケーションをフロントエンドとして使用し、その背後では、主にフェリーの予約や乗船手続き、マイレージ加算などを行う WCF サービスが動作しています。そして、バックエンドには SQL Server を使用しています。この社内システムに関わるリューションとプロジェクトを示すと、次図のようになります。

図 5. 社内向けシステム実行時の構成

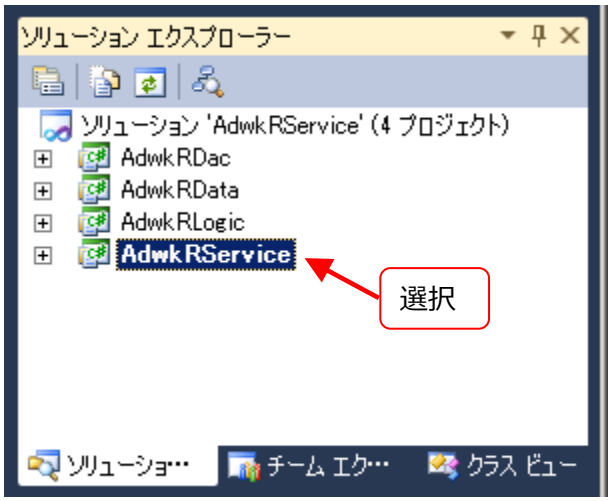



使用するソリューションとしては、AdwkWpfApp.sln、AdwkRService.sln、および、AdwkMService.sln の 3 つです。これらのソリューションは、SamplesWCF フォルダー直下の各サブフォルダーに保管されています。

以下に、この社内システムを使用して、フェリーの予約と乗船手続きを行う操作手順を示します。

1. AdwkWpfApp.sln、AdwkRService.sln、および、AdwkMService.sln の 3 つのソリューションをそれぞれエクスプローラー上でダブルクリックなどして、Visual Studio 2010 で開きます。つまり、Visual Studio 2010 のインスタンスは 3 つになります。(Administorator 以外のユーザーでログインしている場合は管理者権限付きで Visual Studio 2010 を起動します)
2. 予約管理サービス（AdwkRService プロジェクト）を実行するため、まずは、ソリューション AdwkRService.sln が開いた Visual Studio において、ソリューション エクスプローラー上で、「AdwkRService」プロジェクトをクリックして選択します。（選択する場所は、次図のように、ルートノードの下にある「AdwkRService」プロジェクトノードです。ルートノード自体の「ソリューション 'AdwkRService'」ではないので、注意してください。）

図 6. AdwkRService プロジェクトをクリックして選択

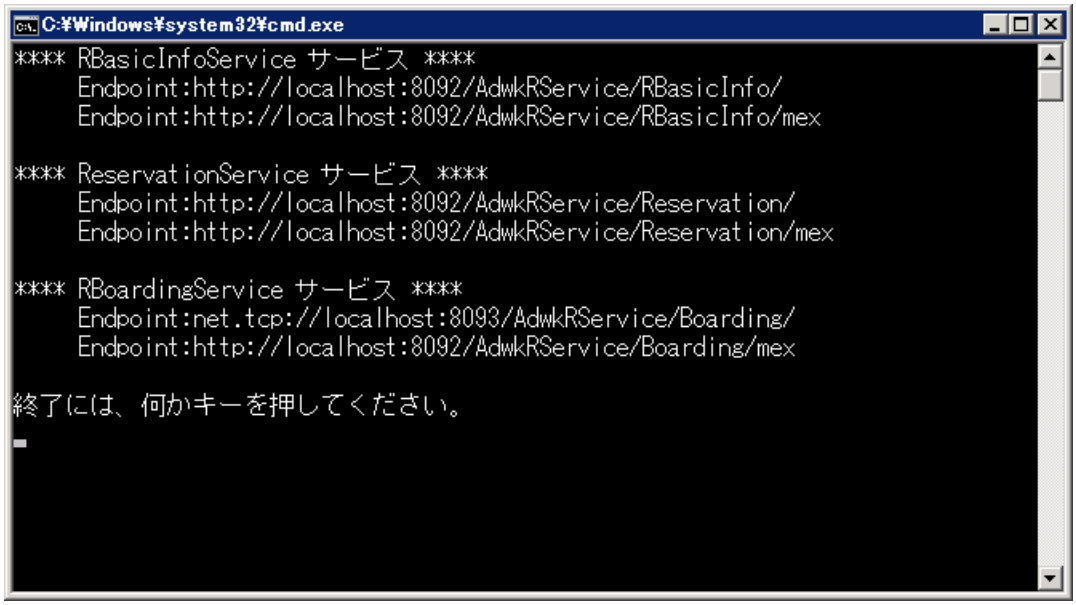



3. AdwkRService プロジェクトを選択した状態にしたまま、Visual Studio のメニュー バーで、［デバッグ］、［デバッグなしで開始］の順にクリックして、このプロジェクトを実行します。

**註:** このサンプルの各ソリューションには、複数のプロジェクトが含まれています。このようなソリューションにおいて、上記の手順 2 と手順 3 のように、ソリューション エクスプローラー上で、特定のプロジェクトを選択した後、メニューから実行コマンドを選択すると、そのプロジェクトが実行できるように、このサンプルでは予め構成してあります。たびたび、サンプルを実行する際には、特定のプロジェクトを実行することがあるので、この方法を利用してみてください。

4. 実行すると、コマンドプロンプトが開き、次のように表示されことを確認します。このまま実行した状態にしておきます。

図 7. 予約管理サービス（AdwkRService プロジェクト）が実行した状態



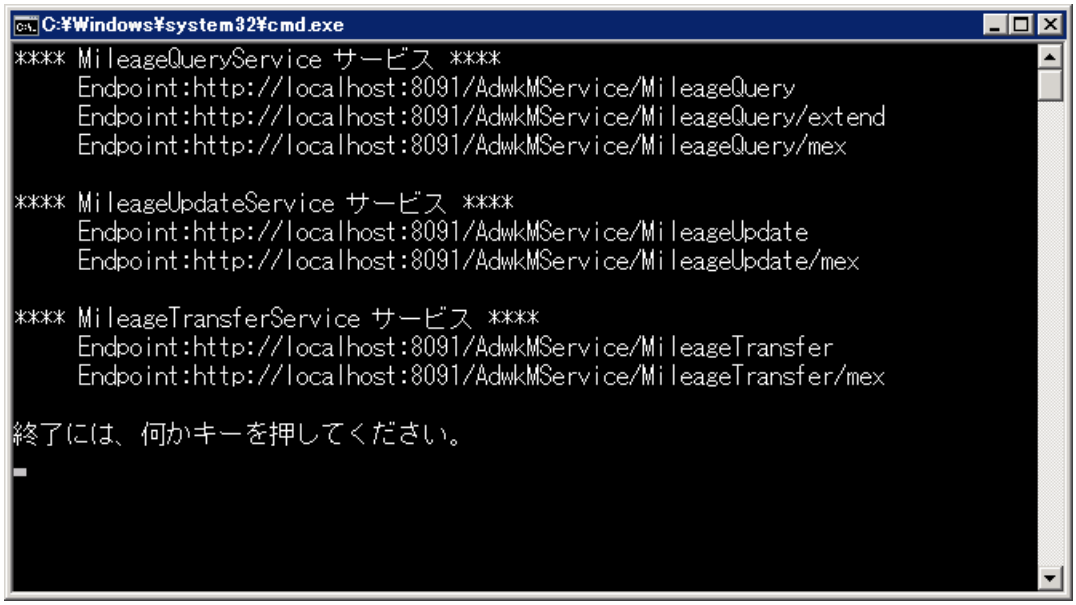

5. マイレージ管理サービス（AdwkMService プロジェクト）を実行するため、まずは、ソリューション AdwkMService.sln が開いた Visual Studio において、ソリューション エクスプロー

ラー上で、「AdwkMService」プロジェクトをクリックして選択します。（選択する場所は、ルートノードの下にある「AdwkMService」プロジェクトノードです。ルートノード自体の「ソリューション 'AdwkMService'」ではないので、注意してください。）

6. AdwkMServiceプロジェクトを選択した状態にしたまま、Visual Studioのメニュー バーで、［デバッグ］、［デバッグなしで開始］の順にクリックして、このプロジェクトを実行します。

7. 実行すると、コマンドプロンプトが開き、次のように表示されことを確認します。このまま実行した状態にしておきます。

図 8. マイレージ管理サービス（AdwkMService プロジェクト）が実行した状態




これで、社内システムに必要な 2 つの WCF サービス（図 5 の[D]と[F]）が起動しました。なお、ここでの手順では、マイレージを外部のパートナー企業に移行しないので、DummyExternalService（図 5 の[I]）は使用しません。このあとの、顧客向け Web サイトで使用します。

次にクライアントを起動しましょう。

8. クライアント（AdwkWpfApp プロジェクト）を実行するため、まずは、ソリューション AdwkWpfApp.sln が開いた Visual Studio において、ソリューション エクスプローラー上で、「AdwkWpfApp」プロジェクトをクリックして選択します。（選択する場所は、ルートノードの下にある「AdwkWpfApp」プロジェクトノードです。ルートノード自体の「ソリューション 'AdwkWpfApp'」ではないので、注意してください。）

9. AdwkWpfApp プロジェクトを選択した状態にしたまま、Visual Studio のメニュー バーで、［デバッグ］、［デバッグなしで開始］の順にクリックして、このプロジェクトを実行します。

10. 実行すると、次のように［予約管理・乗船手続］メインウィンドウが表示されることを確認します。

図 9. AdwkWpfApp プロジェクトのメインウィンドウ


ここまでが、このサンプルの社内システムの起動に必要な手順です。これで、社内システムが利用できるようになりました。ここで、フェリーの予約を行ってみましょう。

11. ウィンドウ左側の中間あたりにある［運行予定照会/予約］ボタンをクリックします。すると、次図のように［運行予定照会/予約］ウィンドウが表示されます。（表示直前に WCF サービスを呼び出して応答を得るまで表示しないので、表示まで多少時間が掛るかもしれません。この点は、第 2 部の「4.2 クライアントからの同期呼び出し」で改善させます。）

図 10. ［運行予定照会/予約］ウィンドウ

12. 経路やクラス、照会期間は、前図のように既定のままにして、［運行予定照会］ボタンをクリックし、運行予定データを WCF サービス（AdwkRService）から取得します。

13. すると、次図のように運行予定が一覧表示されます（表示までに時間が掛るかもしれません）。ここで、一覧から先頭行（運行日が 2011/07/01 の 0001 便、クラスは一等）を選択します。姓と名の欄には、それぞれ「山田」、「太郎」、会員コードには「0000100001」と入力して、［予約］ボタンをクリックします。

**図 11. 運行予定を選択して予約データを入力**


14. すると予約が完了して、次図のように予約番号が表示されるので、その番号を姓名とともに控えておきます。（予約番号は、このサンプルの使用状況によって異なります。）

**図 12. 予約完了（予約番号を控えること）**

15. ［運行予定照会/予約］ウィンドウの閉じるボタン［×］をクリックして、このウィンドウを閉じて、元の［予約管理・乗船手続］メインウィンドウに戻ります。

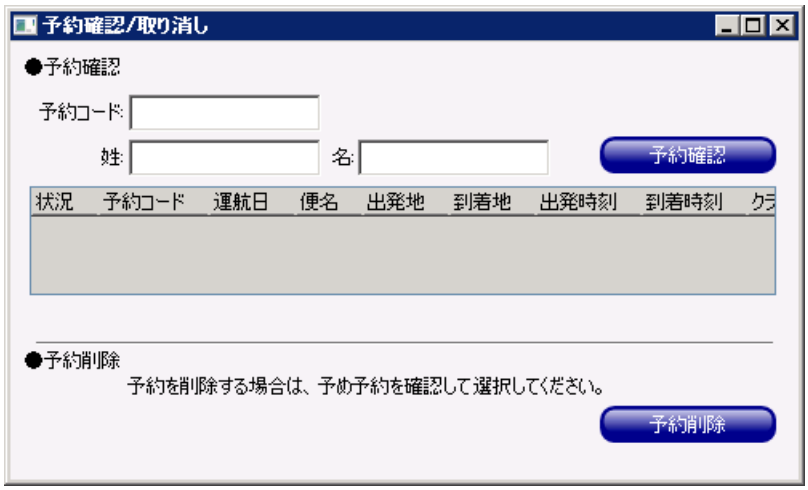
これで、フェリーの予約ができました。余力があれば、手順 11 から手順 15 を繰り返して、予約を複数回行ってみましょう。その際、経路や期間の指定も変えてみましょう。なお、後から予約の照会や削除を行うためにも、予約した際の「姓名」と「予約番号」を控えておいてください。

> **註:** 運航予定は 2011/07/01 から 2011/07/07 までの分が用意されています。また、他の顧客として、以下の姓名、会員コードでの登録が可能です。
> 鈴木 花子、0000100002
> 佐藤 一郎、0000100003
> なお、会員コード無しの顧客（非会員）も入力可能であり、会員コードを空欄にする場合は、任意の姓名で予約できます。

次に予約の確認をしてみましょう。

16. 次に予約の確認をするため、［予約管理・乗船手続］メイン ウィンドウ右側の中間あたりにある［予約確認/取り消し］ボタンをクリックします。すると、次図のように［運行予定照会/予約］ウィンドウが表示されます。

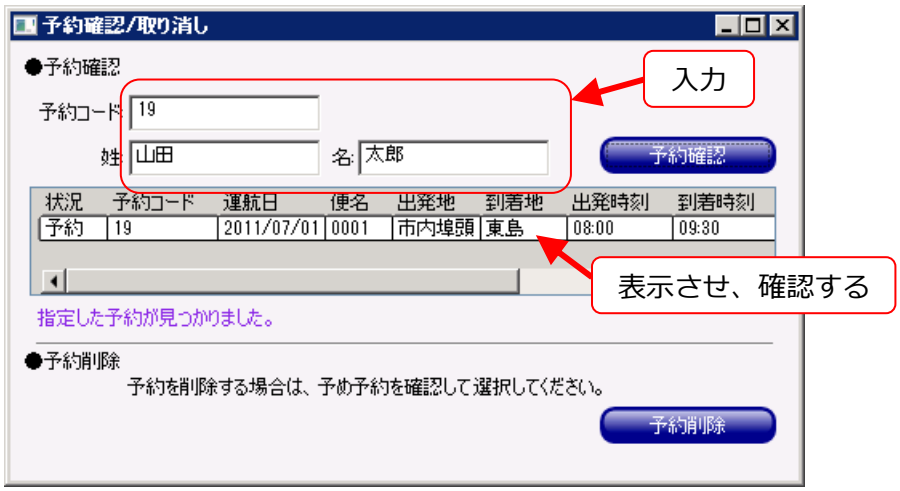
**図 13. ［予約確認/取り消し］ウィンドウ**




> **註:** このサンプルのテーマは WCF サービスであるので、ユーザー インターフェイスは比較的簡単なものになっています。上記の予約確認のウィンドウでは、予約の有無を確認するには、姓名と予約番号（予約コード）を正確に指定する必要があります。本来であれば、特定の顧客の全予約を表示するなど、予約の検索条件の指定に、様々なバリエーションがあるのが望ましいですが、この例では省略しています。

17. 次図のように「予約コード」欄には、前の手順で控えた予約番号を入力し、姓に「山田」、名に「太郎」と入力して、［予約確認］ボタンをクリックし、予約データを表示させて確認します。

図 14. 予約データの検索




ここで、仮に予約の行を選択した後、［予約削除］ボタンをクリックすると、この予約データを削除できます。ここでは、このあとで乗船手続きを行うので、削除せずに、このウィンドウを閉じることにします。

18. ［予約確認/取り消し］ウィンドウの閉じるボタン［×］をクリックして、このウィンドウを閉じて、元の［予約管理・乗船手続］メインウィンドウに戻ります。

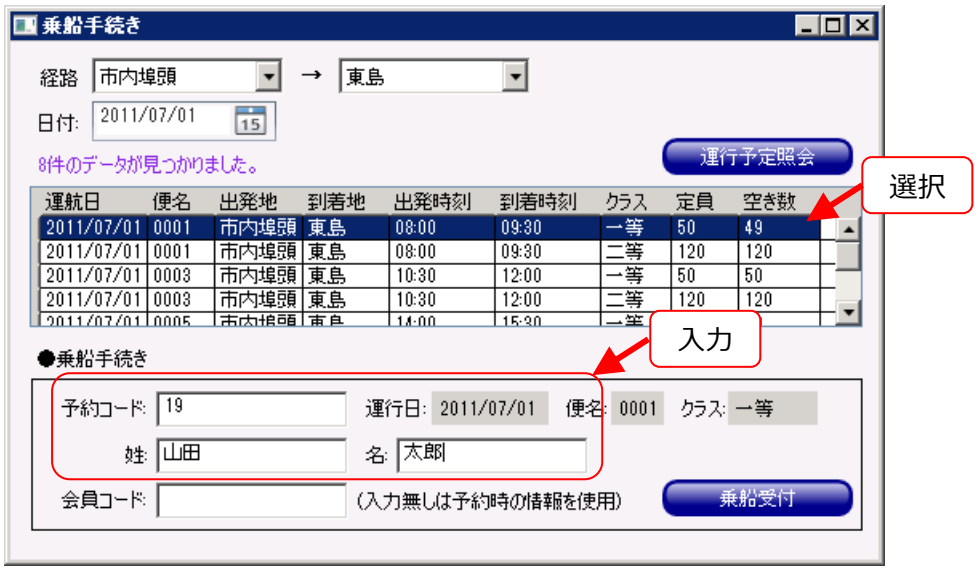
この社内システムの運用時の流れとしては、予約担当の社員によって予約が済んだ後、実際の乗船日に、別の担当者が乗船手続きを行うことになります。ここでは引き続き、［予約管理・乗船手続］メインウィンドウから乗船手続きを続けます。

19. 次に乗船手続きを行うため、［予約管理・乗船手続き］メイン ウィンドウ左側下部の［乗船手続き］ボタンをクリックします。すると、次図のように［乗船手続き］ウィンドウが表示されます。

図 15. ［乗船手続き］ウィンドウ

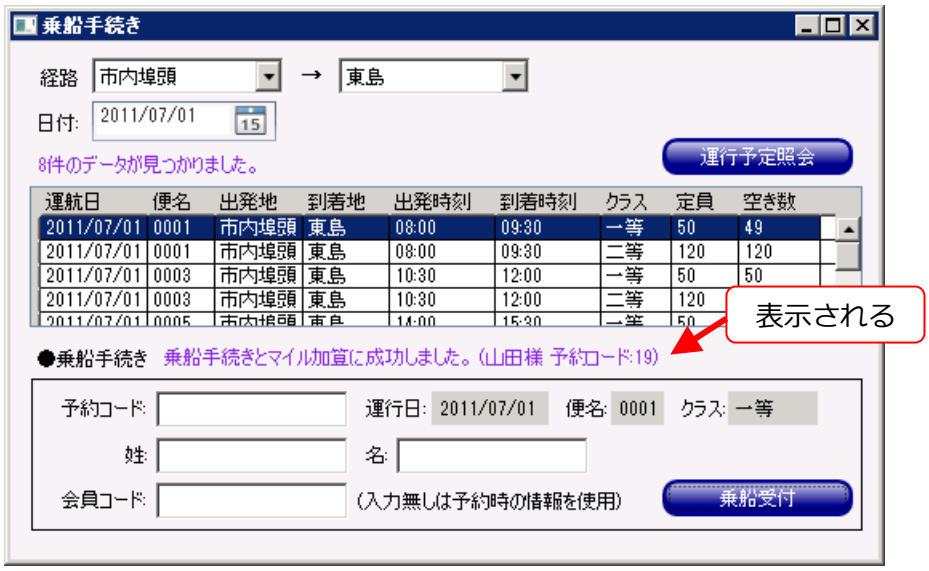

20. 経路や日付は既定のままにして、［運行予定照会］ボタンをクリックします。次図のように運行予定の一覧が表示されたら、先頭の行（運航日 2011/07/01、便名 0001、クラスは一等）を選択した後、控えていた同じ予約番号（予約コード）を入力し、姓と名にそれぞれ、「山田」、「太郎」と入力し、［乗船受付］ボタンをクリックします。

図 16. 乗船データを入力




21. すると、予約管理サービス（AdwkRService）を呼び出して乗船手続きを行い、さらに、マイレージ管理サービス（AdwkMService）を呼び出してマイレージ加算も行い、次図のように手続きが正常に成功した旨のメッセージが表示されることを確認します。

図 17. 乗船手続き成功




22. 確認が済んだら、［乗船手続き］ウィンドウの閉じるボタン［×］をクリックして、このウィンドウを閉じて、元の［予約管理・乗船手続］メインウィンドウに戻ります。

これで、社内システムでの基本的な操作の確認が済みました。

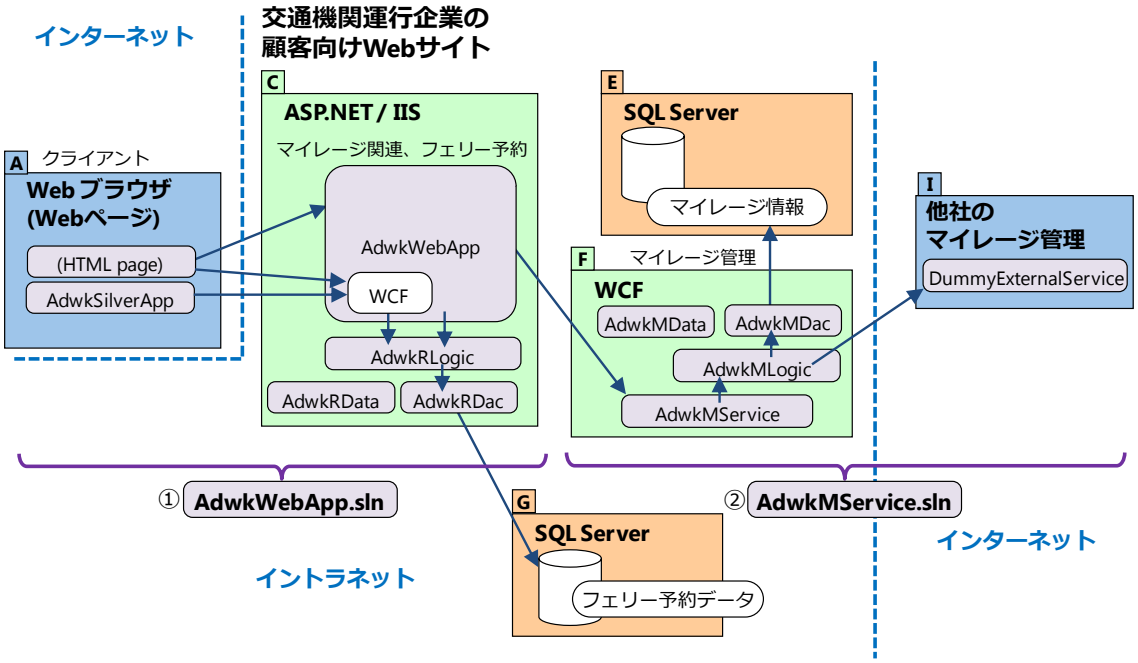
23. ［予約管理・乗船手続］メインウィンドウの閉じるボタン［×］をクリックして、このウィンドウを閉じて、AdwkWpfApp アプリケーションを閉じます。
24. 2 つの WCF サービス（AdwkRService、AdwkMService）のそれぞれのコマンドプロンプトで、いずれかのキーを押して、WCF サービスを終了させ、コマンド プロンプトも閉じておきます。
25. AdwkWpfApp.sln、AdwkRService.sln、および、AdwkMService.sln の 3 つのソリューションを開いている Visual Studio を終了します。

## 顧客向け Web サイト

この Web サイトでは、ASP.NET Web アプリケーションを実行しており、クライアントはブラウザーを使用します。また、Web サイト上では、ASP.NET Web アプリケーションと同じホスティング環境上で、フェリー予約関連の WCF サービスも稼働しており、クライアントの Web ページから予約管理の WCF サービスへ直接アクセスします。このフェリー予約関連のサービスは、第 3 部で主に扱います。また、Web サイト上の Web アプリケーション自体が WCF サービスのクライアントとして、マイレージ管理サービスを利用しています。

この Web サイトのシステムに関わる部分のソリューションとプロジェクトを示すと、次図のようになります。

図 18. 顧客向け Web サイト




ソリューションとしては、AdwkWebApp.sln、および、AdwkMService.sln の 2 つを使用します。これらのソリューションは、SamplesWCF フォルダー直下の各サブフォルダーに保管されています。

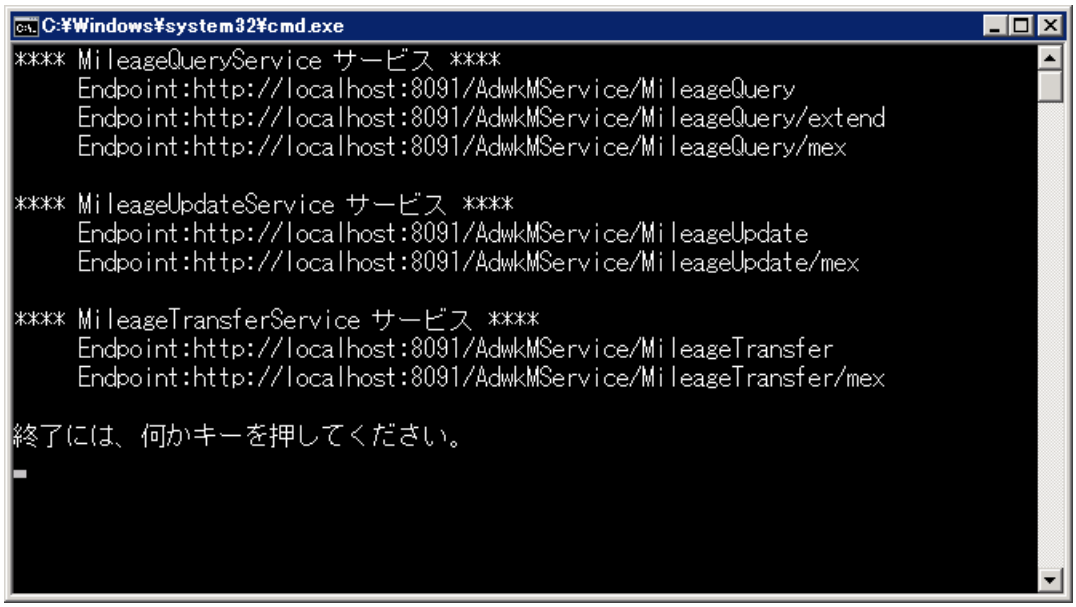
以下に、このシステムを使用して、マイレージ確認などの行う操作手順を示します。

1. AdwkWebApp.sln、および、AdwkMService.sln の 2 つのソリューションをそれぞれエクスプローラー上でダブルクリックなどして、Visual Studio 2010 で開きます（つまり、Visual Studio 2010 のインスタンスは 2 つになります）。

> **註:** この Web サイトで稼働する予約関連の WCF サービスは、ASP.NET のホスティング環境に組み込まれているため、AdwkWebApp プロジェクトに含まれています。よって、前述の社内システムで使用した AdwkRService ソリューションは使用しません。

2. マイレージ管理サービス（AdwkMService プロジェクト）を実行するため、まずは、ソリューション AdwkMService.sln が開いた Visual Studio において、ソリューション エクスプローラー上で、「AdwkMService」プロジェクトをクリックして選択します。（選択する場所は、ルートノードの配下にある「AdwkMService」プロジェクトノードです。ルートノードの「ソリューション 'AdwkMService'」ではないので、注意してください。）
3. AdwkMService プロジェクトを選択した状態にしたまま、Visual Studio のメニュー バーで、［デバッグ］、［デバッグなしで開始］の順にクリックして、このプロジェクトを実行します。
4. 実行すると、コマンドプロンプトが開き、次のように表示されることを確認します。このまま実行した状態にしておきます。

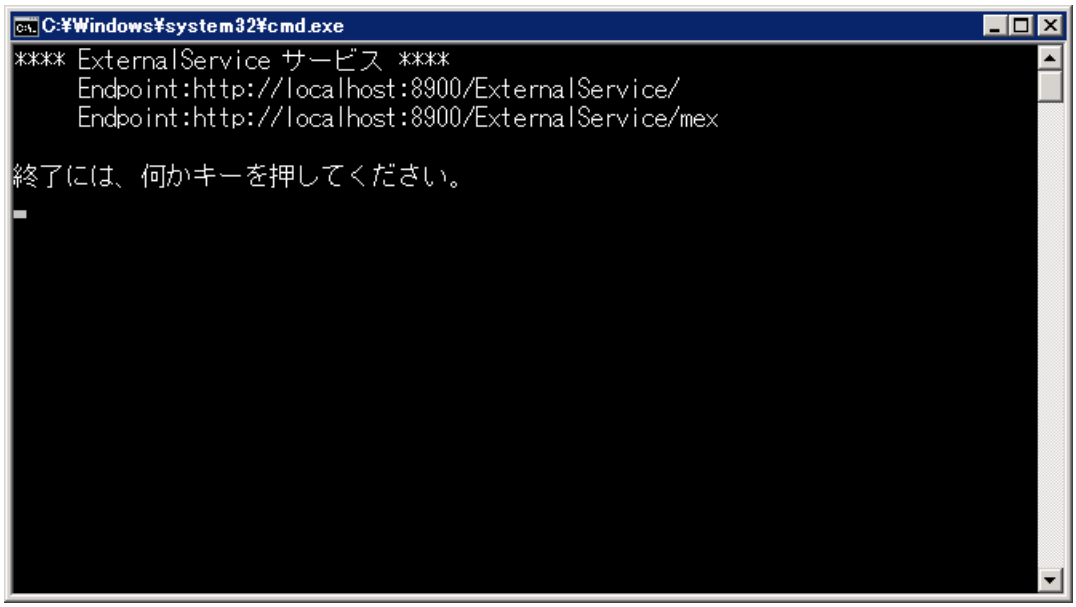
図 19. マイレージ管理サービス（AdwkMService プロジェクト）が実行した状態




このほか、マイル移行のために、外部のパートナー企業のマイレージ管理サービスを起動する必要があります。そのサービスは、DummyExternalService プロジェクトに実装されています。

5. 外部のパートナー企業のマイレージ管理サービス（DummyExternalService プロジェクト）を実行するため、まずは、ソリューション AdwkMService.sln が開いた Visual Studio において、ソリューション エクスプローラー上で、「DummyExternalService」プロジェクトをクリックして選択します。
6. DummyExternalService プロジェクトを選択した状態にしたまま、Visual Studio のメニューバーで、［デバッグ］、［デバッグなしで開始］の順にクリックして、このプロジェクトを実行します。
7. 実行すると、コマンドプロンプトが開き、次のように表示されことを確認します。このまま実行した状態にしておきます。

図 20. 外部企業のマイレージ管理サービス（DummyExternalService プロジェクト）を実行した状態




次に、以下の手順に従って AdwkWebApp プロジェクトを実行して、ASP.NET のホスティング環境と、クライアントである Internet Explorer を起動します。

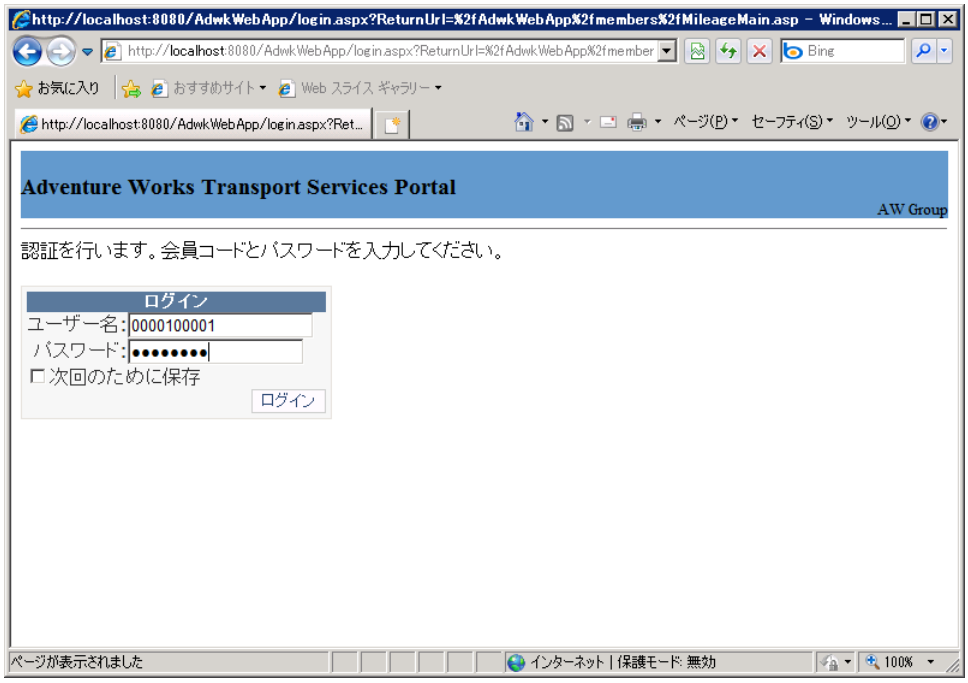
8. ソリューション AdwkWebApp.sln を開いた Visual Studio において、ソリューション エクスプローラー上で、「AdwkWebApp」プロジェクトをクリックして選択します。（選択する場所は、ルートノードの下にある「AdwkWebApp」プロジェクトノードです。ルートノード自体の「ソリューション 'AdwkWebApp'」ではないので、注意してください。）
9. AdwkWebApp プロジェクトを選択した状態にしたまま、Visual Studio のメニュー バーで、［デバッグ］、［デバッグなしで開始］の順にクリックして、このプロジェクトを実行します。
10. 実行すると、ホスティング環境である「ASP.NET 開発サーバー」が起動するほか、次のように Web アプリケーションの Default.aspx ページが表示されことを確認します。

図 21. 顧客向け Web サイトの初期ページ


ここでは、マイルの履歴と移行を行ってみましょう。

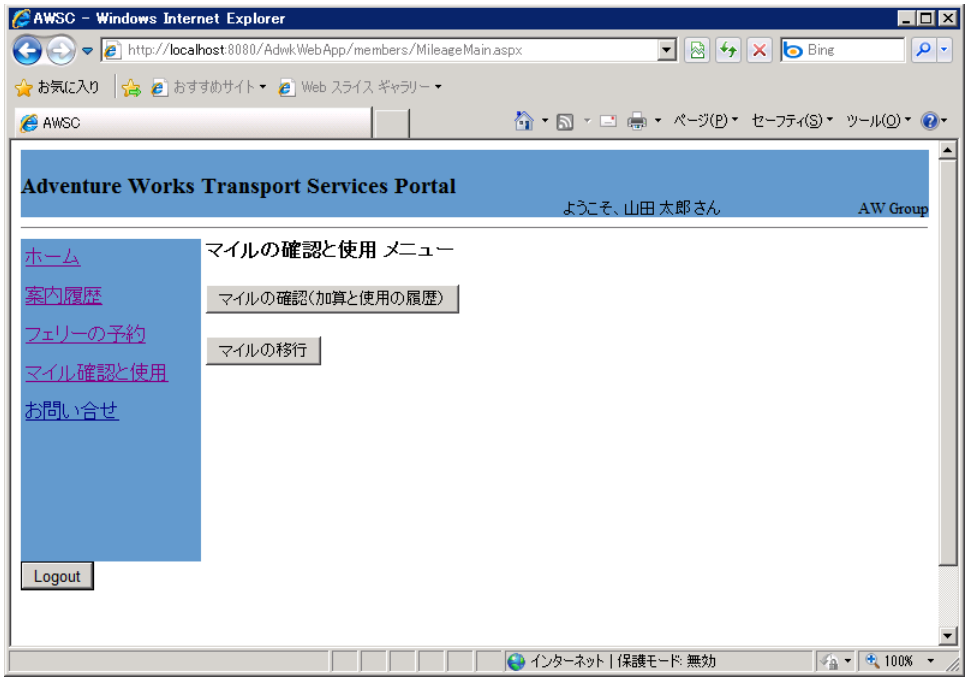
11. ページ上の左側の［マイル確認と使用］リンクをクリックします。すると、現時点では未認証のため、次図のようにログインページが表示するので、ユーザー名に「0000100001」、パスワードに「password」と入力し、［ログイン］ボタンをクリックします。

図 22. ログインページ



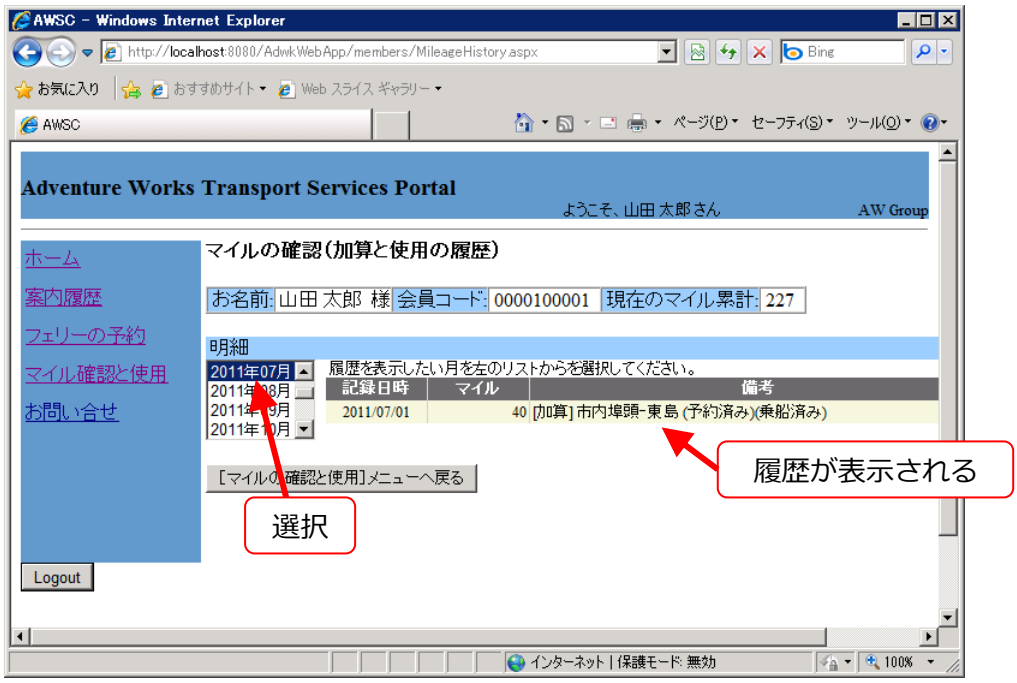
12. すると、「山田太郎」さんとしてログインした後、［マイル確認と使用］リンクから本来表示すべきリンク先に移動し、次のように「マイルの確認と使用 メニュー」が表示されます。

図 23. マイルの確認と使用 メニュー




13. メニューに列挙された［マイルの確認（加算と使用の履歴）］ボタンをクリックします。すると、次図のようにマイルの確認ページが表示されるので、「年月」のリストボックスの中から「2011 年 07 月」をクリックして選択し、その月の履歴を表示させます。

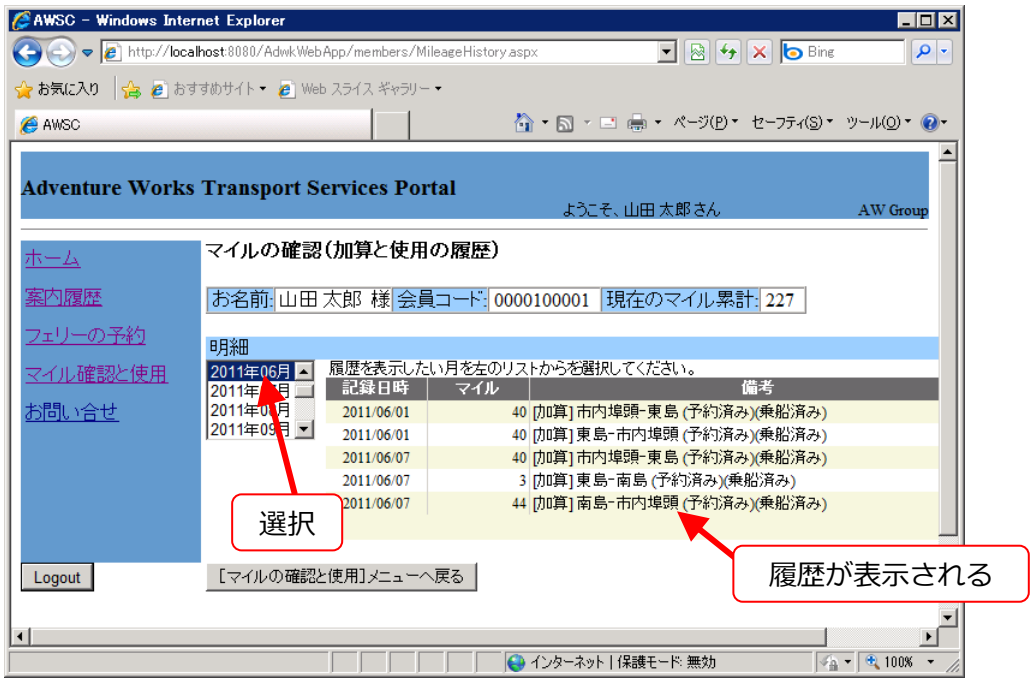
図 24. 履歴確認



前図に表示された履歴は、前の手順で乗船手続きを行って、マイル加算されたものです。

14. 同様に「年月」のリストボックスから「2011 年 06 月」を選択し、該当する履歴を表示させます（次図参照）。

図 25. 別の履歴の確認



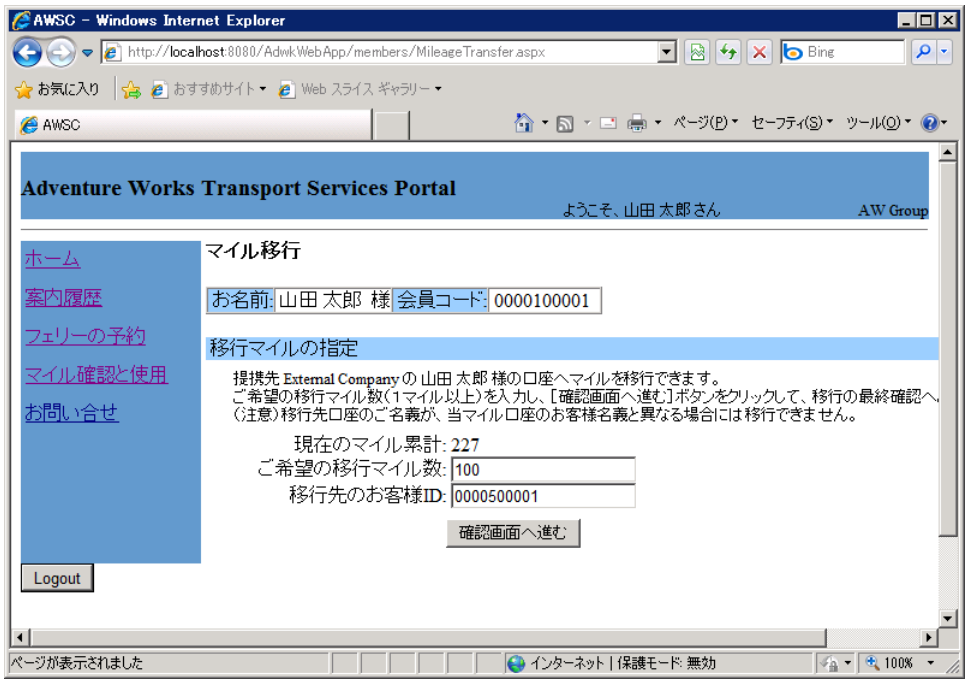

この履歴は、既にサンプルのデータベースに用意してあったものです。

**註:** この履歴確認のページの「年月」のリストボックスは、簡単にするため、サンプルデータの年月を含む、2011 年 04 月から 2013 年 03 月までの 24 カ月が表示できるよう、このページのプログラムコード（MileageHistory.aspx.cs）にハードコーディングしています。

15. 確認が済んだら、ページ下部の［［マイルの確認と使用］メニューへ戻る］ボタンをクリックして、図 23 のメニューに戻ります。

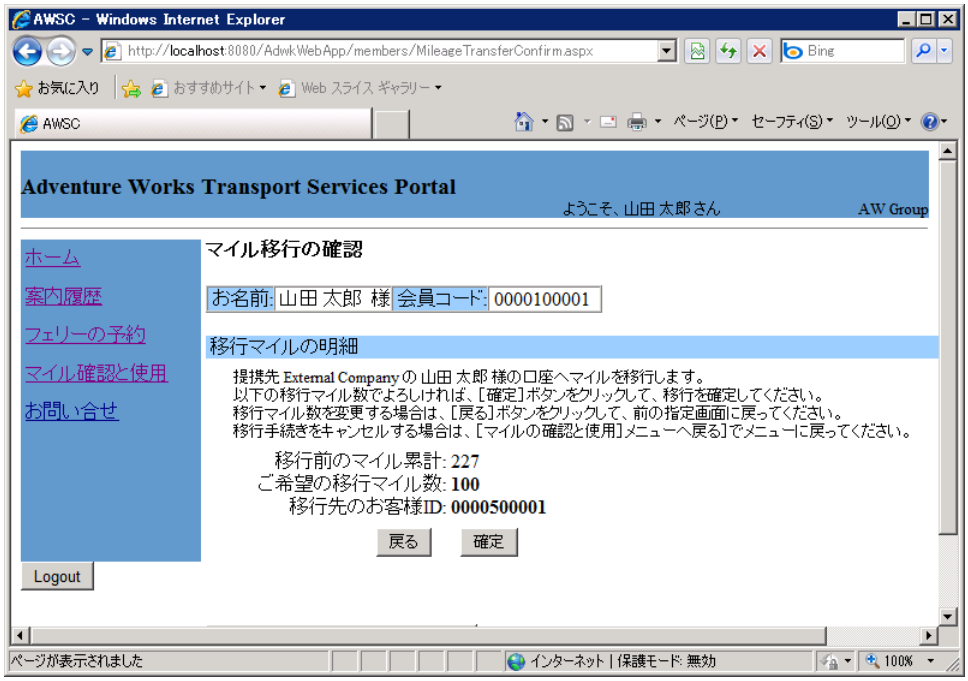
16. 今度はマイルを移行するため、メニューに列挙された［マイルの移行］ボタンをクリックします。すると、次図のように表示されるので、「ご希望の移行マイル数」欄には「100」と入力し、「移行先のお客様 ID」には「0000500001」と入力します。（この ID は、山田太郎さんの移行先の ID です。正しく入力しないと、マイルの移行処理に失敗します。）入力が済んだら、［確認画面へ進む］ボタンをクリックします。

図 26. マイルの移行



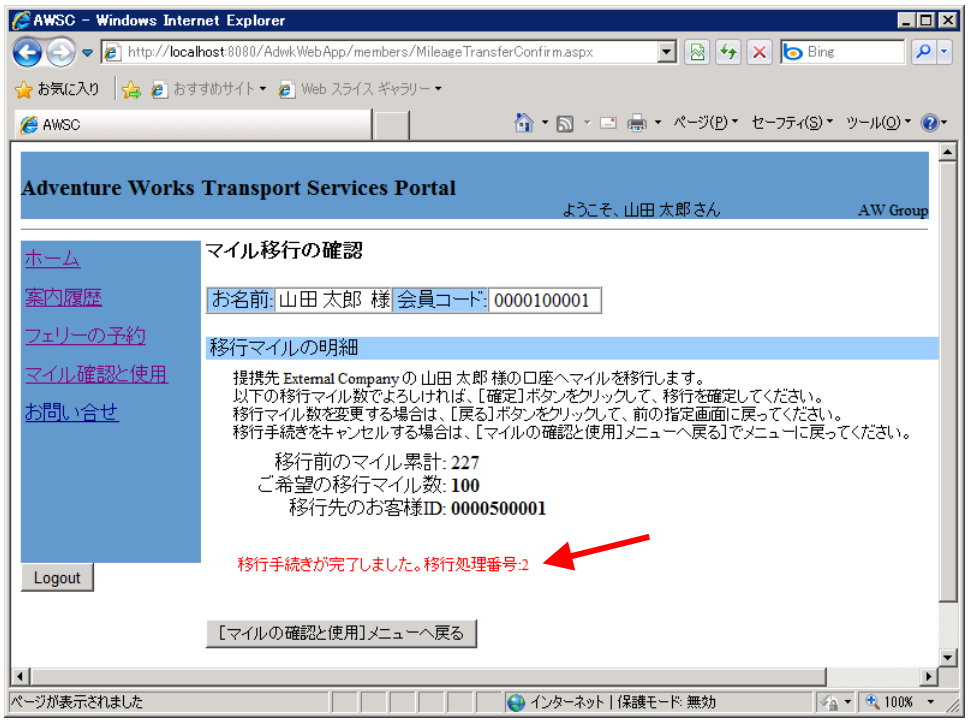

17. すると次図のように、マイル移行の確認ページになるので、［確定］ボタンをクリックします。

図 27. マイル移行の確認




18. すると、次図のように移行完了のメッセージが表示されるので、確認します。

図 28. マイル移行の完了




この移行手続きによって、DummyExternalService の WCF サービスが呼び出され、指定されたマイルが移行されました。

> **註:** このほか、以下のユーザーが Web サイトにログインできるほか、「鈴木花子」さんは、同一名義の移行先の ID を持っているので、マイルの移行処理も可能です。
>
> | ［氏名］ | ［ユーザー名］ | ［パスワード］ | ［移行先 ID］ |
> |---|---|---|---|
> | 鈴木 花子 | 0000100002 | password | 0000500002 |
> | 佐藤 一郎 | 0000100003 | password | |

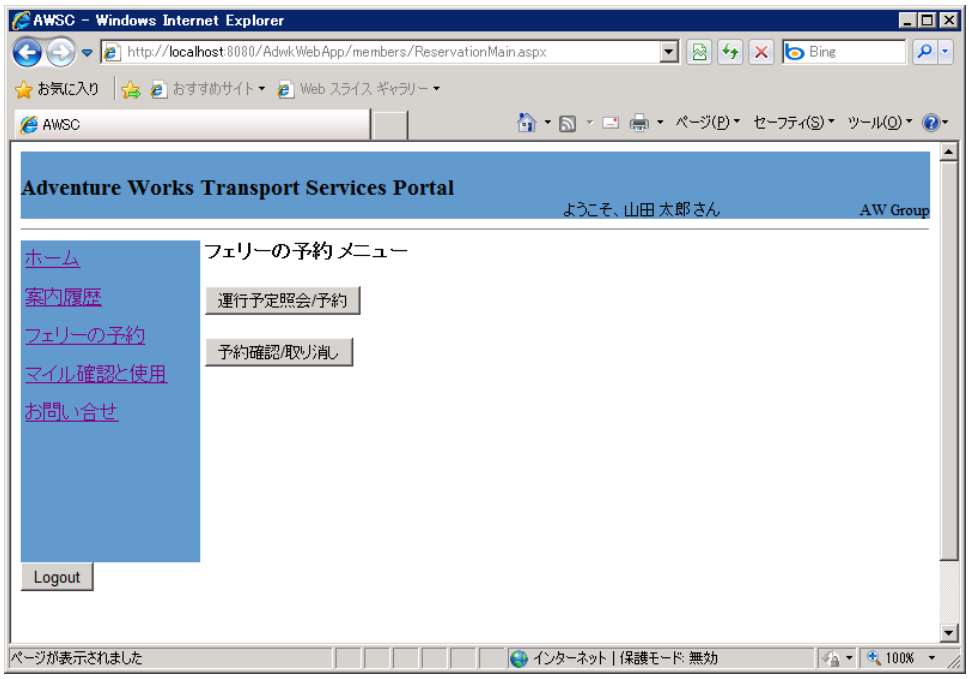
このあと、［［マイルの確認と使用］メニューへ戻る］ボタンをクリックすれば、図 23 のメニューへ戻ることができるほか、ページの左辺のリンク一覧からリンクをクリックして、他の処理を確認しても構いません。

今度は、この Web アプリケーションを使用して、予約の登録と確認、削除を行ってみましょう。

引き続き、顧客向け Web サイト（AdwkWebApp）を使用します。既にすべて終了してしまった場合は、手順 1 から手順 12 を行って、図 21 のように、顧客向け Web サイトの Web ページが利用できる状態にしてください。

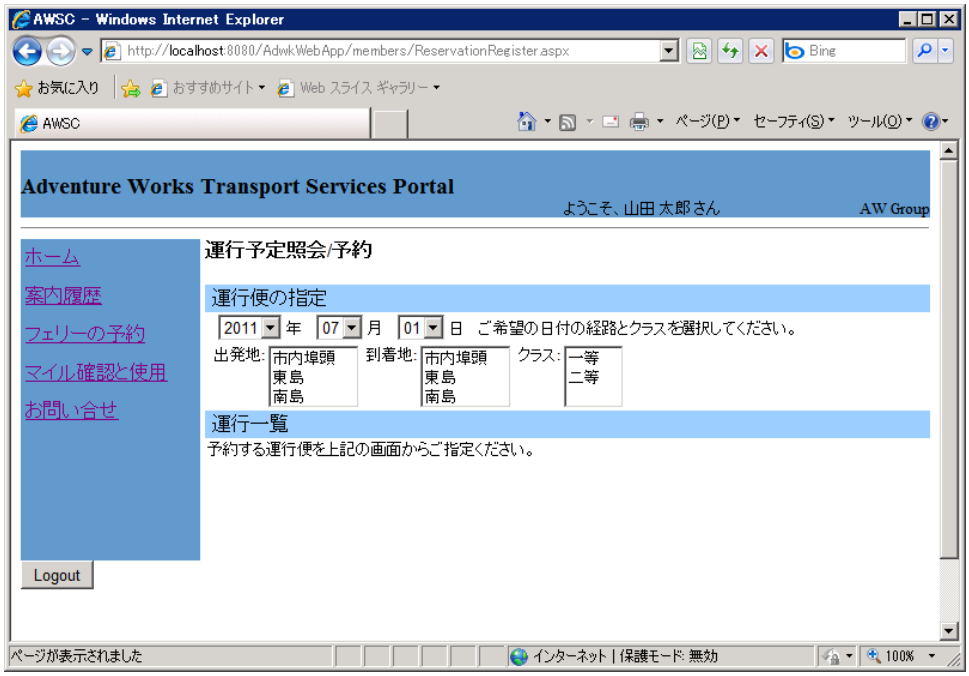
19. Web ページ上の左側の［フェリーの予約］リンクをクリックします。もし、ログインページが表示された場合は、ユーザー名に「0000100001」、パスワードに「password」と入力し、［ログイン］ボタンをクリックします。［フェリーの予約］リンクからの移動が完了すると、次のように「フェリーの予約」メニューが表示されます。

図 29. フェリーの予約 メニュー




20. メニューに列挙された［運行予定照会/予約］ボタンをクリックします。次図のように「運行予定照会/予約」ページが表示されることを確認します。

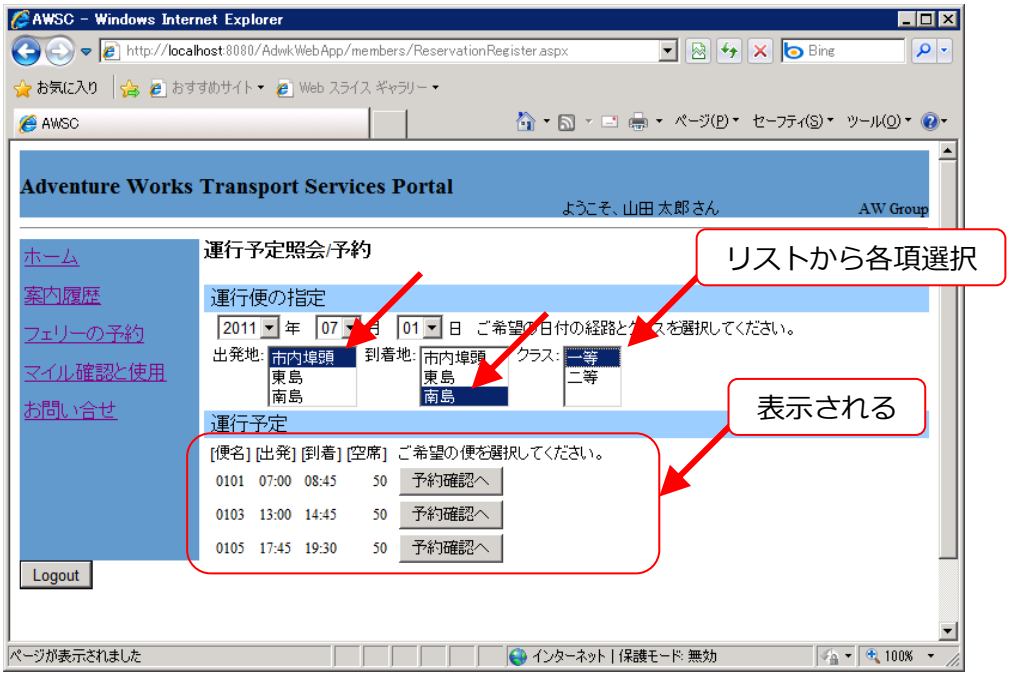
図 30. 運行予定照会/予約 ページ




**註:** このサンプルでは、この Web ページから WCF サービスを呼び出すことに焦点を当てているため、ユーザー インターフェイスは簡素化しており、ここでは、運行便の検索指定の方法としては、日単位で指定する方法のみ提供しています。

21. 前述のページが表示されたら、「運行便の指定」セクションにおいて、日付は既定のままにして、出発地、到着地、および、クラスのリストボックスの中から、それぞれ、「市内埠頭」、「南島」、「一等」の順にクリックして選択します。すべての選択が済むと、次図のように、該当する運行便の一覧が表示されることを確認します。

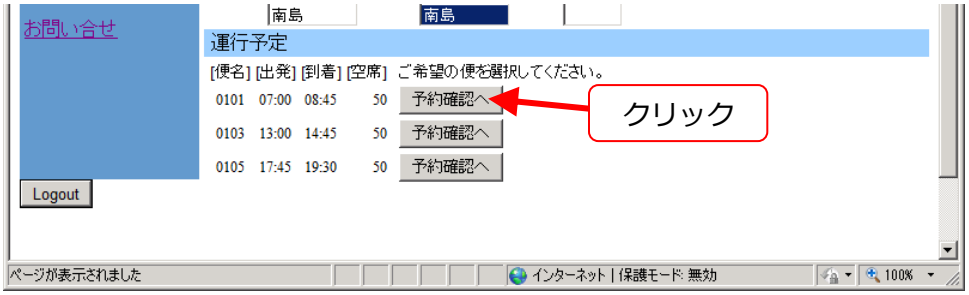
**図 31. 運行便の指定と該当データの一覧表示**




**註:** 前述の Web ページでは、日付、出発日、到着地、また、クラスなどの指定条件を変えると、それに呼応して、該当データが一覧表示されます。余力があれば、指定条件を変更して、表示の変化を確認してみてください。
これらの動作は、この Web ページに記述されたクライアント側の JavaScript によるものであり、この JavaScript から Web サイト側の WCF サービスを呼び出し、その結果を一覧として表示しています。
確認が済んだら、この後の手順に合わせるため、手順 21 で行った指定条件に戻してください。

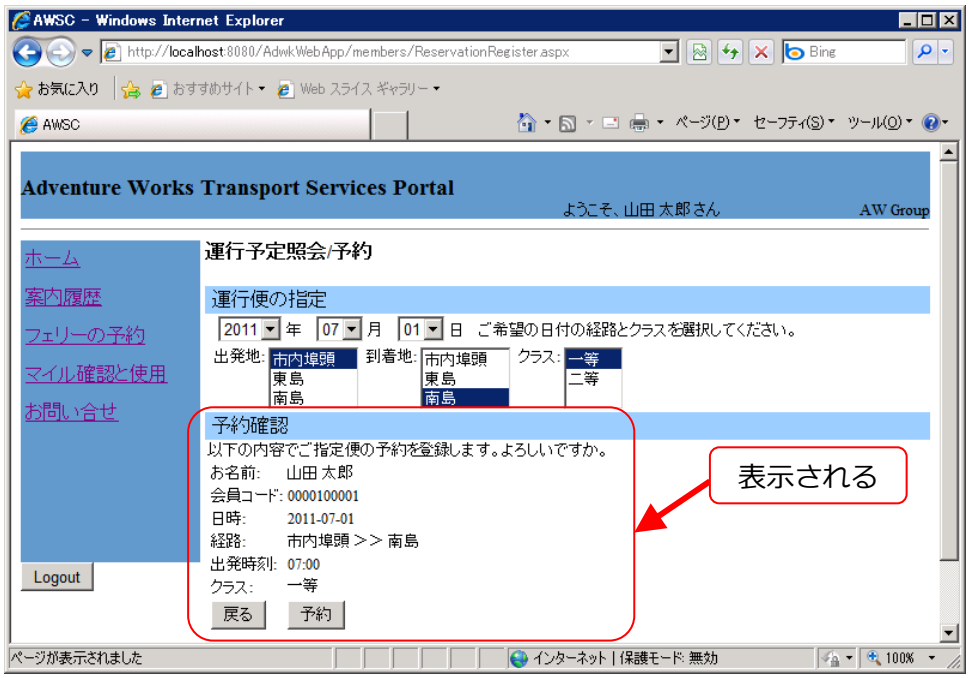
22. 図 31 の Web ページのように表示された「運行予定」セクションの一覧の中から、1 行目の「0101 便」を予約するため、次図のように同じ行の右端の［予約確認へ］ボタンをクリックします。

**図 32. ［予約確認へ］ボタンをクリック**



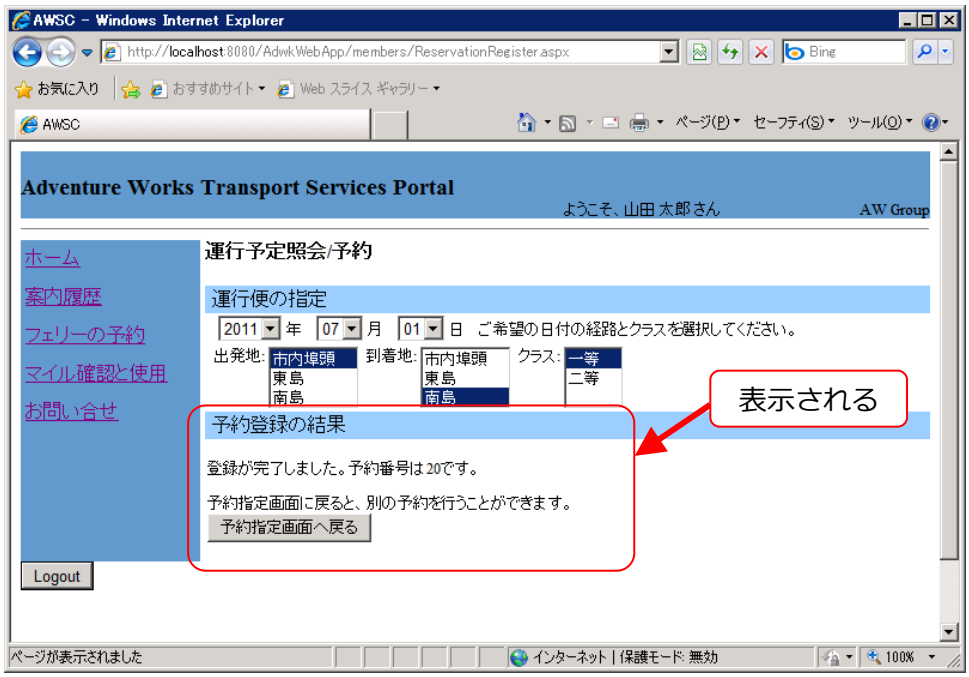
23. すると、予約の確認を行うために表示内容が切り替わるので、次図のように表示されることを確認します。

図 32. 予約の確認



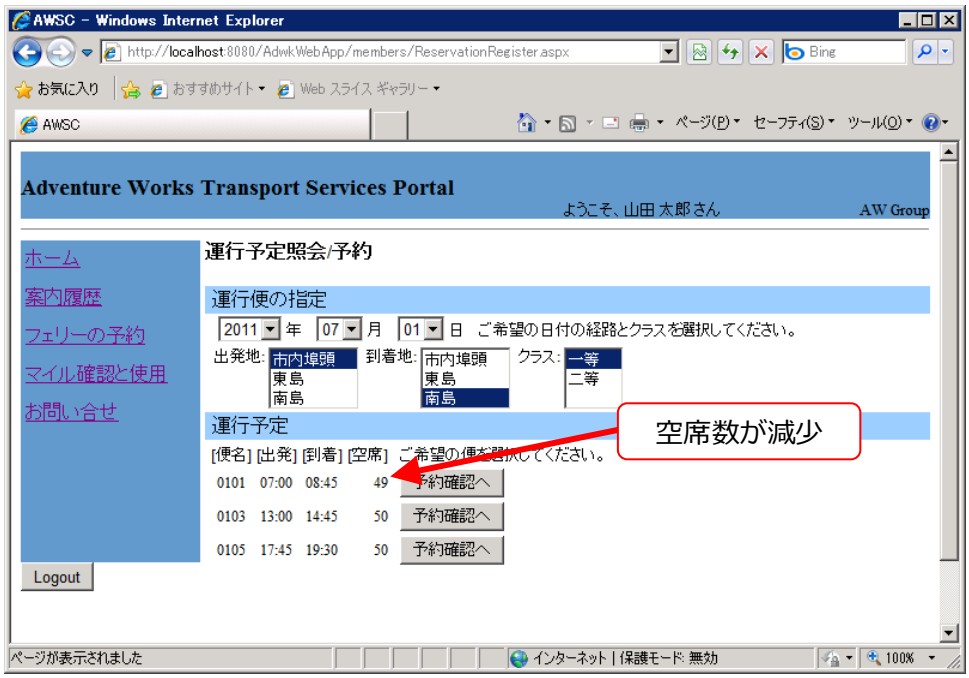

24. ここで、前図のページ下部にある［予約］ボタンをクリックして、予約を確定させます。次図のように予約の登録が完了した旨のメッセージが表示されることを確認します。ここでも、後で予約を確認できるようにするため、予約番号を控えておいてください。（実際の予約番号は、このサンプルの使用状況によって異なります。）

図 34. 予約登録の完了



25. 前図のページ下部にある［予約指定画面へ戻る］をクリックして、運行一覧の最新状態を表示させ、予約した運行便の空席数が減っていることを確認します。

図 35. 元の運行便の一覧（空席数は減少）



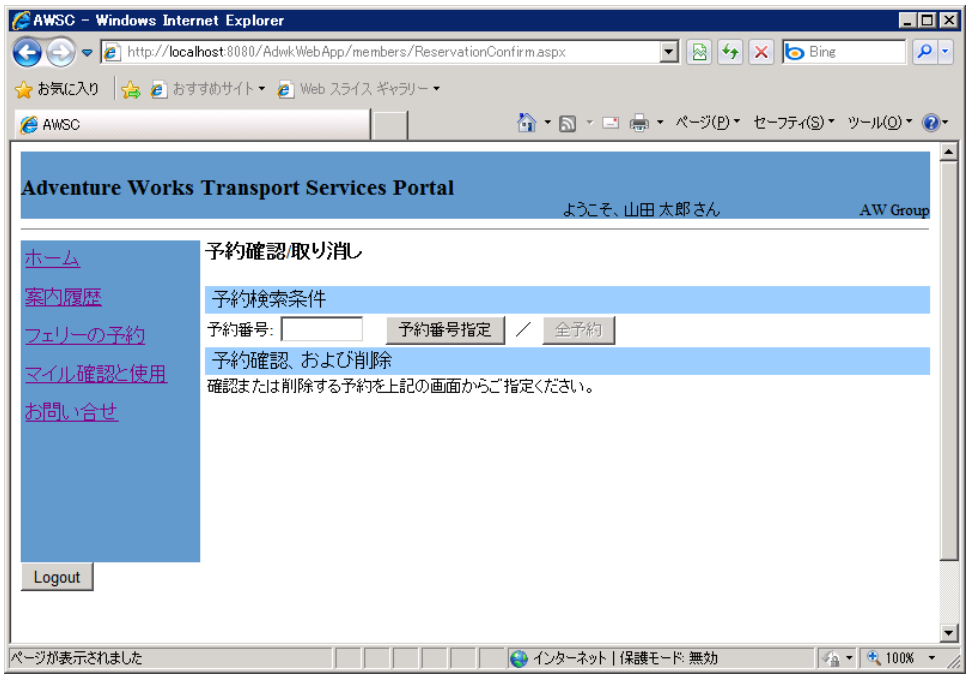

最後に予約の確認と削除を行ってみましょう。

26. Web ページの左側にある［フェリーの予約］リンクをクリックします。次図のように「フェリーの予約」メニューが表示されることを確認します。

図 36. 運行予定照会/予約 ページ

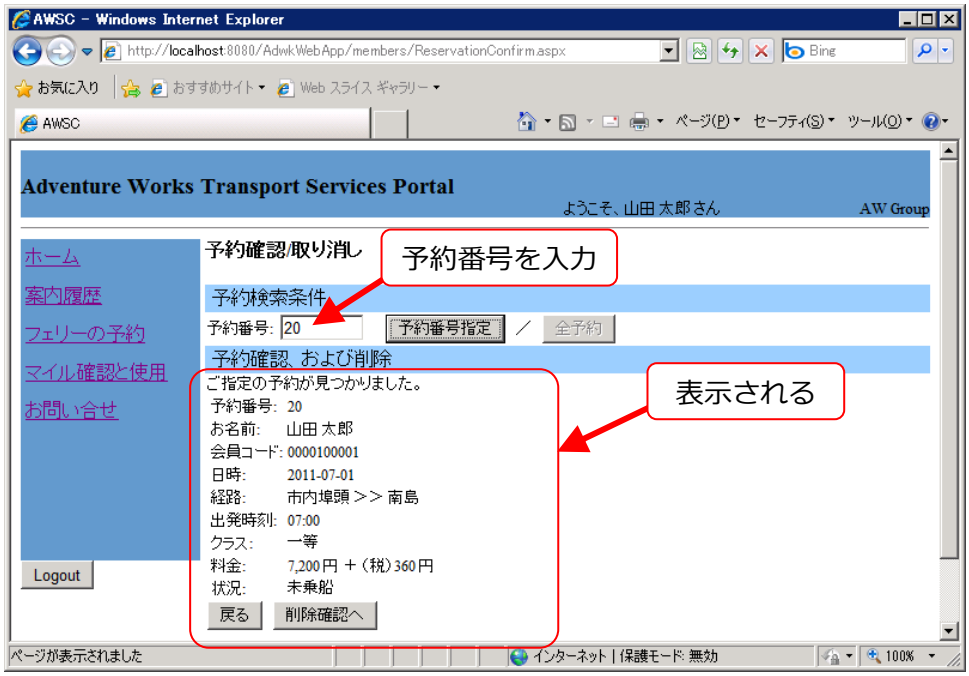
27. 前図のメニューに列挙された［予約確認/取り消し］ボタンをクリックします。次図のように「予約確認/取り消し」ページが表示されることを確認します。

**図 37. 予約確認/取り消し ページ**



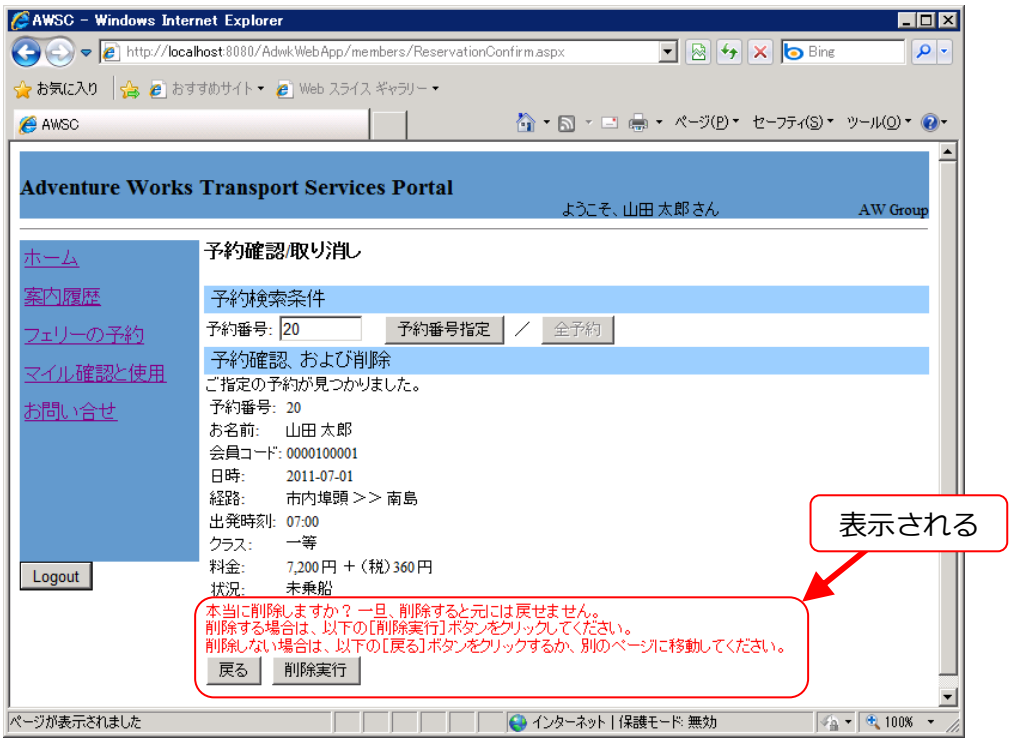

28. 予約時に控えておいた予約番号を、次図のように予約番号欄に入力して、［予約番号指定］ボタンをクリックします。該当する予約データが表示されることを確認します。

**図 38. 該当する予約データの表示**



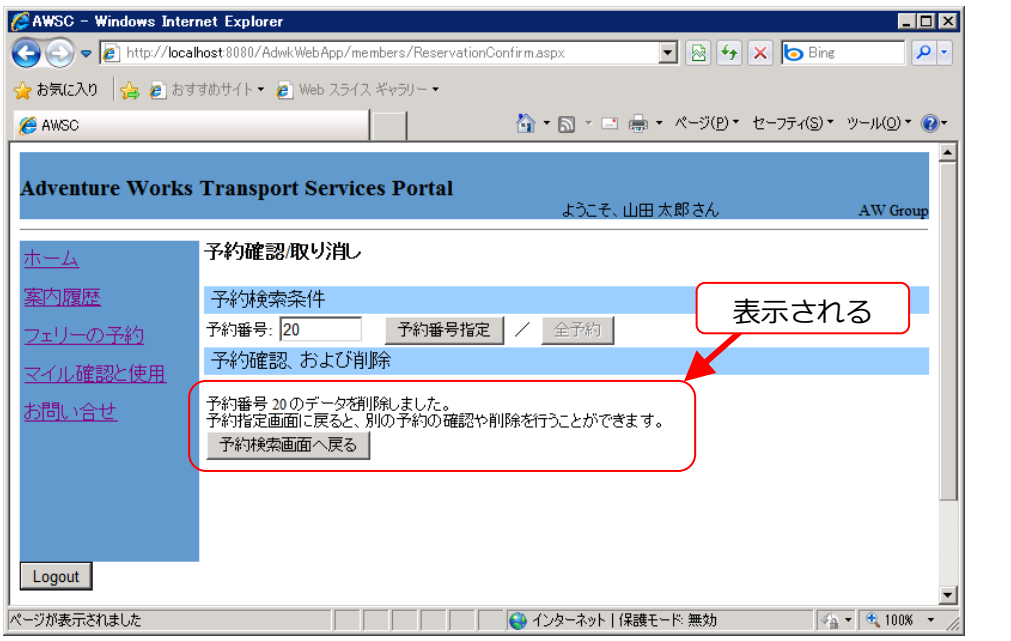

29. 前図のページ下部にある［削除確認へ］ボタンをクリックします。次図のように、削除を確認するメッセージが表示されることを確認します。

図 39. 削除の確認



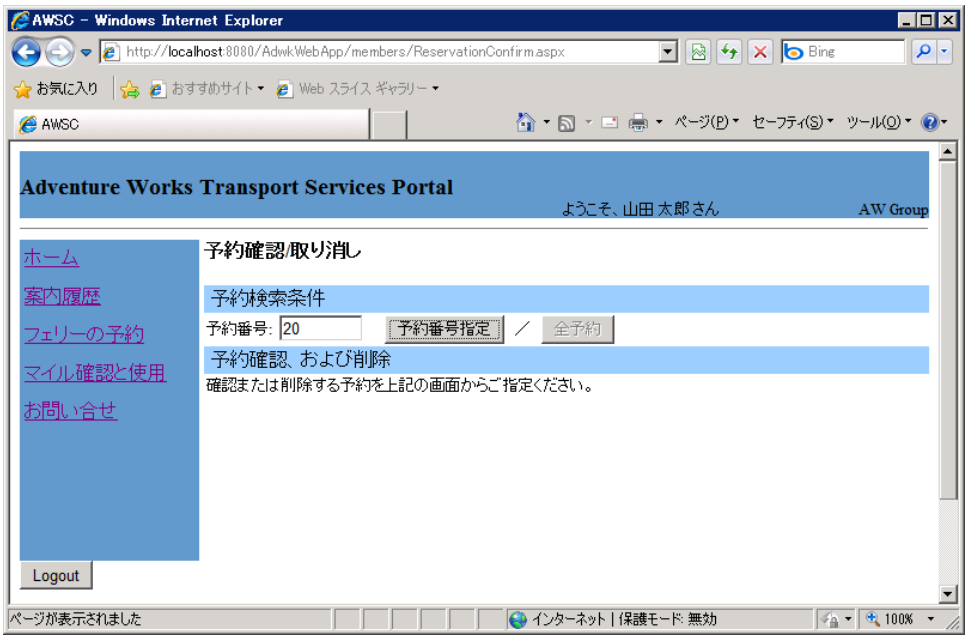

30. 削除を確定するため、前図のページ下部にある［削除実行］ボタンをクリックします。次図のように、予約が削除された旨のメッセージが表示されることを確認します。

図 40. 削除の完了



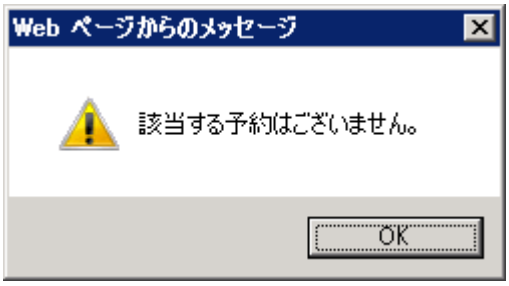

31. 前図のページ下部の［予約検索画面へ戻る］ボタンをクリックします。次図のように、検索前の「予約確認/取り消し」ページに戻ることを確認します（同じ予約番号が入力されたままの状態です）。

図 41. 元の予約検索画面




32. ここで、再び同じ予約を検索するため、同じ予約番号を指定したまま、［予約番号指定］ボタンをクリックします。すると、次図のように、予約が見つからない旨のメッセージボックスが表示されることを確認します。

図 42. 該当する予約がない場合の表示




33. 確認が済んだら、［OK］をクリックして、メッセージボックスを閉じます。
34. 確認が済んだら、Web ページ（Internet Explorer）、および、AdwkMService プロジェクトと DummyExternalService プロジェクトの各サービスを終了します。

以上、AWTS サンプル プログラムの中の主要部分である「社内向けシステム」と「顧客向け Web サイト」について、基本的な操作手順を確認しました。本編では、ここで取り上げた手順以外にも、必要に応じて補足説明を行う予定です。