

序文

Caché Web サービス・クイックスタート・チュートリアルへようこそ

Cachéは、Web サービス、およびシンプル・オブジェクト・アクセス・プロトコル (SOAP)やWeb サービス記述言語(WSDL)などの Web サービス標準を完全にサポートしています。さらに、Web サービスのサポートが Caché に組み込まれているため、追加のミドルウェアは必要ありません。

Cachéを使用すると、任意のプログラミング言語を使用して構築可能で、任意のプラットフォームで実行できる SOAP 対応アプリケーションで使用できる Web サービスを短時間で容易に作成し、発行できます。また、Cachéを使用すると、任意の言語を使用して構築可能で、任意のプラットフォームで実行できる Web サービスを利用するクライアントを短時間で容易に作成することもできます。

このチュートリアルでは、Web サービスの発行および外部 Web サービスの利用に Cachéを使用する方法について簡単に説明します。また、Caché Web サービスのプロデューサおよびコンシューマの両方を対象とした、標準的なエラー処理についても説明します。

このチュートリアルは3つの章に分かれています。第I章と第II章はそれぞれ独立しているため、どのような順序でも学習を行うことができます。ただし、チュートリアルの第III章は、既に第I章と第II章を学習していることが前提となっています。

1. "[第I章: Cachéを使用したWebサービスの発行](#)" Cachéを使用して Web サービスを発行する方法を学習します。
2. "[第II章: Cachéを使用したWebサービスの利用](#)" Cachéを使用して、Webサービスを利用するクライアントを作成する方法を学習します。
3. "[第III章: CachéでのWebサービスのエラー処理](#)" 標準的なエラー処理をCaché Web プロデューサとコンシューマの両方に提供する方法を学習します。

Note:

このチュートリアルの演習と例では、ユーザがCachéの基本的な作業に習熟していることを前提にしています。Cachéについてさらに学習するには、"[Cachéクイックスタート・チュートリアル](#)"を参照してください。

はじめに

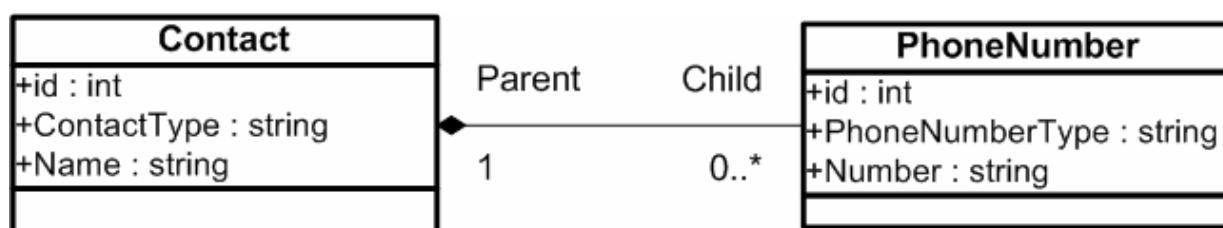
このチュートリアルの第 I 章では、Caché を使用した Web サービスの開発と発行に焦点をあてて説明します。この章の学習を終えると、以下のことを実行できるようになります。

- 単純な Caché Web サービス・プロデューサ・アプリケーションの作成
- Caché Web サービス・プロデューサ・アプリケーションの WSDL ドキュメントの表示
- Caché Web サービス・テスト・ページと Caché ターミナルの両方を使用した、Caché Web サービス・プロデューサ・アプリケーションのテスト
- Caché Web サービス・プロデューサ・アプリケーションにおけるクエリ、複雑なオブジェクト、およびオブジェクトのコレクションの使用

サンプル・アプリケーションのオブジェクト・モデル

このチュートリアルの例と演習では、単純な Caché Contact Management アプリケーションで Web サービス・プロデューサを作成します。アプリケーションのファイルは、Caché インストールの一部として提供されています。作業する Caché ネームスペースにそれらのファイルをインストールする必要があります。すべての例と演習用ファイルのインストール情報へのリンクは、下記のメモを参照してください。

アプリケーションは、**Contact** と **PhoneNumber** という 2 つのクラスで構成されています。これらのクラスは、連絡先とその電話番号を表します。次の図は、アプリケーションのオブジェクト・モデルを表しています。



Contact には、以下のプロパティがあります。

- *id* — オブジェクト ID。 **Contact** の各インスタンスに対して一意です。値は、Caché によって自動的に割り当てられます。Caché スタジオのクラス・エディタやインスペクタには、このプロパティは表示されません。
- *ContactType* — 連絡先のタイプを表します。有効な値は、“Business”と“Personal”のみです。
- *Name* — 連絡先の名前を表します。値は、任意の **%String** です。

PhoneNumber には、以下のプロパティがあります。

- *id* — オブジェクト ID。 **PhoneNumber** の各インスタンスに対して一意です。値は、Caché によって自動的に割り当てられます。Caché スタジオのクラス・エディタやインスペクタには、このプロパティは表示されません。
- *PhoneNumberType* — 電話番号のタイプを表します。有効な値は、“Business”、“Home”、“Fax”および“Mobile”のみです。
- *Number* — 電話番号を表します。値は、任意の **%String** です。形式に関する制限はありません。

これらのプロパティのサポートに加えて、2 つのクラス間には親子リレーションシップが確立されています。つまり、各 **Contact** (親クラス) オブジェクトに任意の数の **PhoneNumber** (子クラス) オブジェクトを含めることができます。ただし、各 **PhoneNumber** オブジェクトは 1 つの **Contact** のみに含める必要があります。**PhoneNumber** オブジェクトは、**Contact** と独立して存在することはできません。また、**PhoneNumber** オブジェクトを複数の **Contact** に含めることはできません。このリレーションシップは、次の 2 つのプロパティによって表されます (図には示されていません)。

- *PhoneNumbers* — **Contact** のプロパティ。 **Contact** に含まれる **PhoneNumber** (子) オブジェクトのコレクションを表します。
- *Contact* — **PhoneNumber** のプロパティ。 **PhoneNumber** の親である **Contact** オブジェクトへの参照を含みます。

Note:

Caché アプリケーションの配置とインストールの詳細は、[AppendixA: 例と演習ファイルのインストール]を参照してください。 Caché アプリケーションをインストールしたら、サンプル・データを入力します。 アプリケーションへのデータの入力の詳細は、[AppendixB: サンプル・アプリケーションのコンパイルと生成]を参照してください。

Web サービスのバックグラウンド

Web サービスは関連する一連のメソッドで、インターネットなどのネットワークを介して呼び出すことができます。Web サービスとそのクライアントは、単純な非占有プロトコルのみを使用して情報を交換します。そのため、場所、プラットフォーム、またはプログラミング言語に関係なく、相互に通信することができます。Web サービス・プロトコルは基本的に、リモート・プロシージャ・コール (RPC) を実行するためのプラットフォーム独立の方法を提供します。

次のテーブルは主な Web サービス・プロトコルとその目的の一覧で、各プロトコルについて簡単に説明しています。

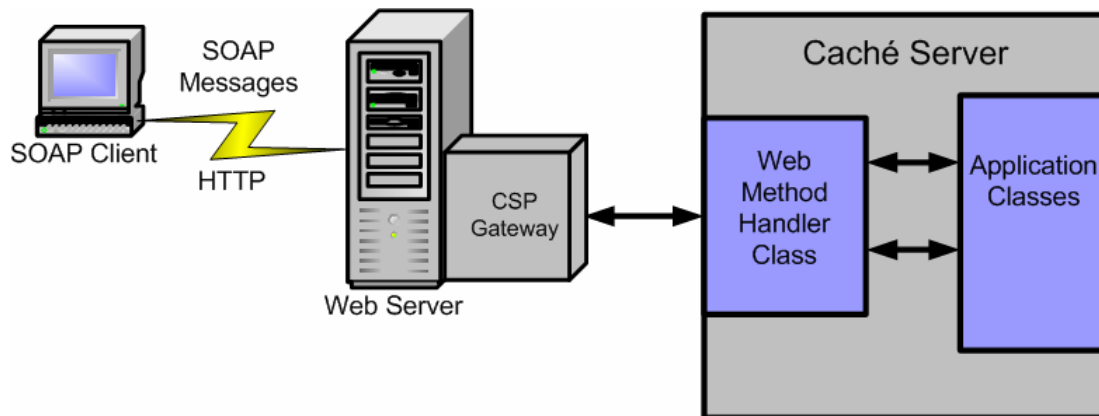
	目的	説明
HTTP	転送	ハイパーテキスト転送プロトコル (HyperText Transfer Protocol)。インターネットで使用される基本的なネットワーク・プロトコル。
SOAP	パッケージ化	シンプル・オブジェクト・アクセス・プロトコル (Simple Object Access Protocol)。Web サービス・メソッドとクライアント間で送信されるメッセージをエンコードするための XML ベースのプロトコル。Web サービス・メソッドに渡される引数とメソッドからクライアントに返される値をエンコードします。
WSDL	説明	Web サービス記述言語 (Web Service Description Language)。Web サービスを記述するための XML ベースのプロトコル。WSDL ドキュメントには、Web サービスのメソッドの署名に加え、Web サービスに関連するデータ型についてのその他の情報が記述されます。WSDL を参照することで、クライアント・コードはその Web サービスのタイプとメソッドを使用することができます。
UDDI	検出	Universal Description, Discovery, and Integration。アプリケーションが Web サービスの記述の検出に使用できる Web サービス・レジストリを作成するための XML ベースのプロトコル。

概要: Web サービス・プロデューサとしての Caché

Caché は、Web サービスを完全にサポートしています。さらに、Web サービスのサポートが Caché に組み込まれているため、追加のミドルウェアは必要ありません。“Web メソッド”としてタグ付けしたメソッドを含む Caché クラスを提供すればすみます。Caché によって、これらのメソッドが自動的に Web サービスとして使用可能になります。また、Web サービスの WSDL ドキュメントが自動的に生成され、クライアント・アプリケーションでそのドキュメントが使用可能になります。

Caché が SOAP クライアントからの要求を処理する基本的な手順は、以下のとおりです。

1. Caché SOAP サーバが、CSP ゲートウェイ経由でクライアント・アプリケーションからの SOAP 要求を受信します。
2. Caché SOAP サーバが SOAP メッセージをアンパックし、すべてのパラメータを適切な Caché 表記に変換して、該当する Web メソッドを呼び出します。
3. Web メソッドがそのコードを実行し、必要に応じて Caché アプリケーション・クラスでメソッドを呼び出すか、またはデータベースに対してクエリを実行します。
4. Web メソッドが応答を作成し、Caché SOAP サーバに返します。
5. Caché SOAP サーバが SOAP メッセージに応答をパッケージ化し、CSP ゲートウェイ経由でクライアントに返します。



Note:

この図は、SOAP クライアントと Caché Web サービスの論理構成を示しています。すべての論理要素を同一の物理マシンに配置することができます。

Caché Web サービス・プロデューサの要件

Web サービスを定義する Caché クラスは、以下の要件を満たしている必要があります。

- Cachéクラスは [%SOAP.WebService](#) を拡張したものである必要があります。このクラスは [%CSP.Page](#) を拡張したものです。
- Web サービスとして公開されるメソッドは、宣言の中に `WebMethod` キーワードを含める必要があります。
- Webサービス・メソッドから返されるCachéオブジェクト、または Webサービスメソッドへの引数として使用されるCachéオブジェクトはすべて“XML 対応”でなければなりません。つまり、[%XML.Adaptor](#) を拡張したクラスのインスタンスである必要があります。

[%SOAP.WebService](#) を拡張したクラスは、以下のパラメータの値を提供する必要があります。

これらのパラメータは、Web サービスの重要な特性を示します。

パラメータ	説明と値
LOCATION	このパラメータの値は、クライアントが Web サービスへのアクセスに使用する URL です。この URL は、Web サービスの WSDL ドキュメントに含まれています。
NAMESPACE	このパラメータの値は URI です。これはサービスにネームスペースを提供して、その名前が別のサービスの名前と競合しないようにします。既定では、値 <code>http://tempuri.org</code> が割り当てられます。開発者は通常、Web サービス開発時にこの値を使用します。導入時には、これを変更する必要があります。
SERVICENAME	このパラメータの値は、サービスにクライアントの名前を提供します。これは、有効な識別子でなければなりません。つまり、文字から開始し、英数字のみを使用する必要があります。

スタジオを使用した Caché Web サービス・クラスの定義

Caché スタジオ開発環境には、Web サービス・クラスを定義するためのウィザードが用意されています。

Contact Management アプリケーションの Web サービス・クラスを定義するには、以下の手順を実行します。

1. Caché スタジオを開きます。 **[ファイル]**→**[ネームスペース変更]**をクリックし、TUTORIALS などの、作業対象のネームスペースに接続します。
2. **[ファイル]**→**[新規作成]**をクリックします。
3. **[新規作成]**ダイアログ・ボックスで**[カスタム]**タブをクリックし、**[ウェブサービス]**アイコンをクリックします。 **[OK]**をクリックします。
4. Web サービス・ウィザードで、Web サービス・パラメータの値とクラス名を入力します。Service Methods の値を追加する必要はありません。チュートリアルの例と演習のサービス・クラスを作成する場合に追加する値は、以下のとおりです。
 - パッケージ名: SOAPTutorial
 - クラス名: SOAPService
 - サービス名: SOAPService
 - サービス・ロケーション: `http://localhost/csp/<ネームスペース名>`。
<ネームスペース名> は、TUTORIALS など、Contact Management アプリケーションをインストールしたネームスペースの名前です。
 - サービス・ネームスペース: `http://tempuri.org`
5. **[OK]**をクリックします。スタジオのクラス・エディタに、指定したクラス定義が表示されます。

Note:

通常、Class Name と Service Name の値は同じでなくてもかまいません。

Cachéスタジオ開発環境の使用に関する詳細は、Cachéドキュメント: [\[Cachéツールとユーティリティ\]](#)-[\[Cachéスタジオの使用法\]](#)を参照してください。

新しいCachéネームスペースの作成方法を学習するには、Cachéドキュメント: [\[Cachéシステム管理\]](#)-[\[Cachéシステム管理ガイド\]](#)-[\[Cachéの構成\]](#)-[\[ネームスペースの構成\]](#)を参照してください。

Web メソッドの追加

Webメソッドは、Webサービスとして公開されるメソッドです。SOAPクライアントは、メソッドにアクセスすることができます。前述のように、Webメソッドは、以下の基準を満たす通常の Caché メソッドです。

- `WebMethod` 属性でタグ付けされる。
- Webメソッドが返すオブジェクト、または引数として受け入れるオブジェクトはすべて、XML対応である。つまり [%XML.Adaptor](#) を拡張したクラスのインスタンスです。

次のメソッドでは、指定された id 値を持つ `Contact` インスタンスが取得され、それがクライアントに返されます。Webサービスのクライアントは、このメソッドを使用して Caché から `Contact` インスタンスを取得することができます。Caché スタジオのクラス・エディタを使用して、

`CacheService` クラスにこのメソッドを追加し、**[ビルド]**→**[コンパイル]**をクリックしてクラスをコンパイルします。

```
ClassMethod GetContact(id As %String) As SOAPTutorial.Contact [ WebMethod ]
{
  if ##class(SOAPTutorial.Contact).%ExistsId(id) {
    quit ##class(SOAPTutorial.Contact).%OpenId(id)
  }
  else {
    quit ""
  }
}
```

Note:

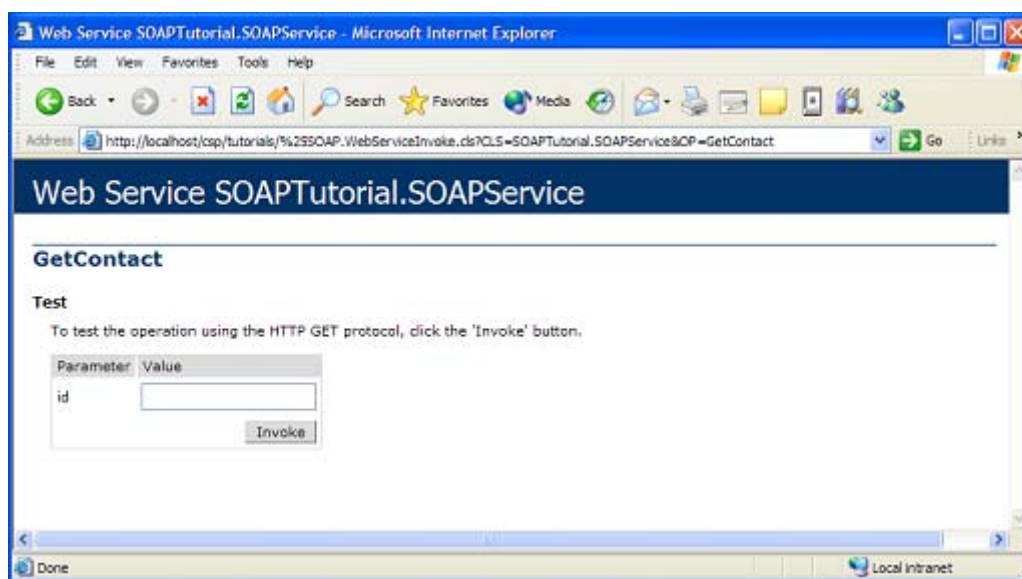
このチュートリアル内の Caché メソッドはすべて Caché ObjectScript を使用してコード化されています。Caché Basic の方が使いやすい場合は、Caché Basic を使用してください。Caché ObjectScript についてさらに学習するには、"[Caché ObjectScript チュートリアル](#)" および、Caché ドキュメント: [\[Caché 開発リファレンス\]](#)-[\[Caché ObjectScript リファレンス\]](#) を参照してください。Caché Basic についてさらに学習するには、"[Caché Basic チュートリアル](#)" および Caché ドキュメント: [\[Caché 開発リファレンス\]](#)-[\[Caché Basic リファレンス\]](#) を参照してください。

Web メソッドの追加

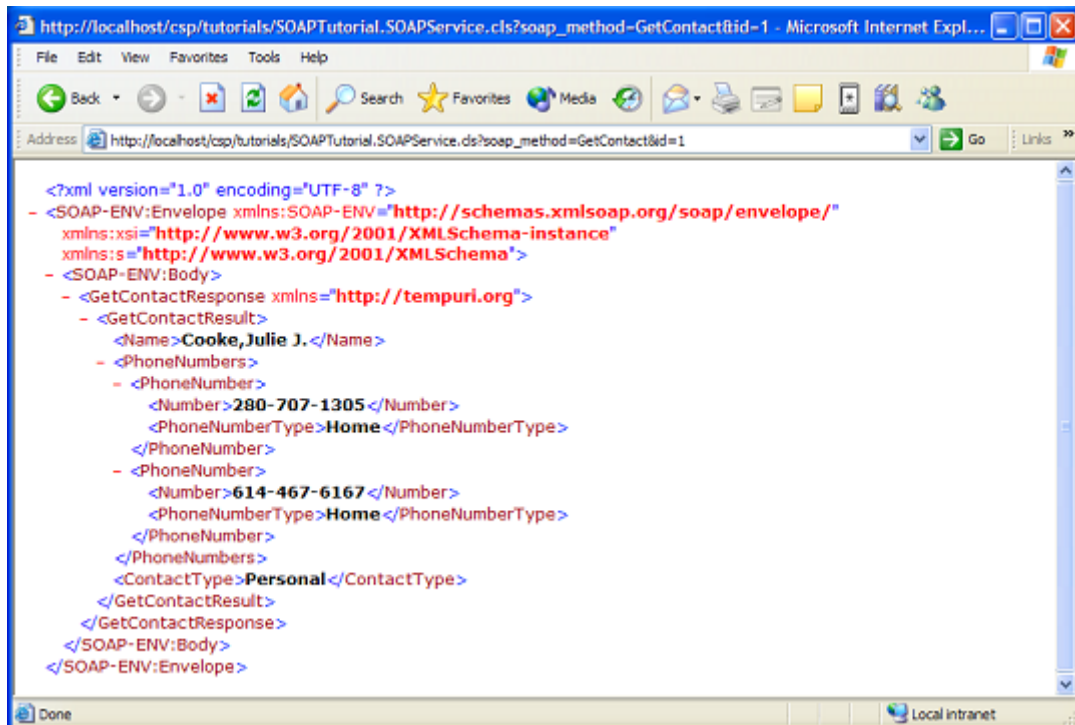
Caché には、新しく定義した Web サービスをテストするためのテスト・ページが用意されています。**SOAPService** をコンパイルしたら、スタジオで **[表示]**→**[ブラウザで表示]** をクリックします。これによってブラウザが起動して、**SOAPService** の Web サービス・テスト・ページが表示されます。



[GetContact] リンクをクリックします。 **GetContact** をテストするページが表示されます。



[id]の有効な値を入力します。[実行]をクリックします。これによって **SOAPService GetContact** メソッドが呼び出されて、テキスト・ボックス内の値がパラメータとしてメソッドに渡されます。メソッドから、SOAP メッセージとしてエンコードされた **Contact** インスタンスが返されます。SOAP でエンコードされた戻り値が次のようにブラウザに表示されます。



Note:

上の画像は、返された SOAP メッセージを Internet Explorer に表示した状態を示しています。ブラウザの種類ごとに異なる既定のスタイルシートを使用して XML ファイルが表示されるため、Internet Explorer 以外のブラウザでは SOAP メッセージが異なって表示されることがあります。

WSDL ドキュメントの表示

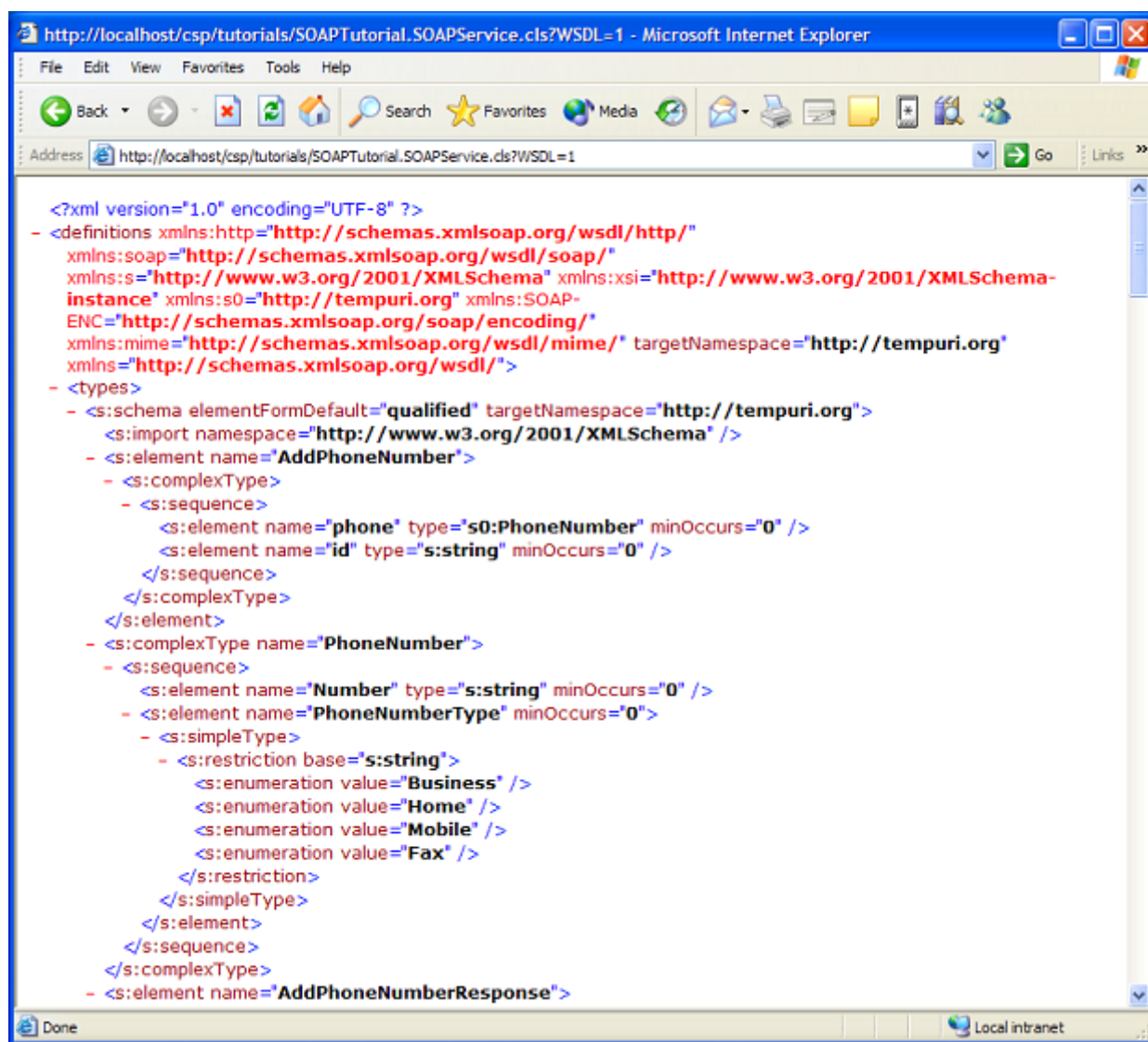
Caché では、Web サービス・クラスの WSDL ドキュメントが自動的に生成されます。ブラウザで WSDL ドキュメントを表示するには、クラスの Web サービス・テスト・ページの URL に?WSDL=1 を追加した形式の URL を開きます。

例えば、**SOAPTutorial.SOAPService** のテスト・ページの URL が

http://localhost/csp/tutorials/SOAPTutorial.SOAPService.cls であるとする、その WSDL ドキュメントの URL は

http://localhost/csp/tutorials/SOAPTutorial.SOAPService.cls?wsdl=1 となります。

以下に、Internet Explorer を使用して表示した **SOAPTutorial.SOAPService** の WSDL ドキュメントの一部を示します。



```
<?xml version="1.0" encoding="UTF-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:s0="http://tempuri.org" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" targetNamespace="http://tempuri.org"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
- <types>
- <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org">
  <s:import namespace="http://www.w3.org/2001/XMLSchema" />
  - <s:element name="AddPhoneNumber">
    - <s:complexType>
      - <s:sequence>
        <s:element name="phone" type="s0:PhoneNumber" minOccurs="0" />
        <s:element name="id" type="s:string" minOccurs="0" />
      </s:sequence>
    </s:complexType>
  </s:element>
  - <s:complexType name="PhoneNumber">
    - <s:sequence>
      <s:element name="Number" type="s:string" minOccurs="0" />
      - <s:element name="PhoneNumberType" minOccurs="0">
        - <s:simpleType>
          - <s:restriction base="s:string">
            <s:enumeration value="Business" />
            <s:enumeration value="Home" />
            <s:enumeration value="Mobile" />
            <s:enumeration value="Fax" />
          </s:restriction>
        </s:simpleType>
      </s:element>
    </s:sequence>
  </s:complexType>
  - <s:element name="AddPhoneNumberResponse">
```

複雑なオブジェクトでの作業

SOAPService GetContact メソッドは、**Contact** のインスタンスである複雑なオブジェクト・インスタンスを返します。 Caché Web サービス・メソッドは、複雑なオブジェクトを引数として受け入れることもできます。 複雑なオブジェクトの戻り値と同様、引数の複雑なオブジェクト・タイプは XML 対応である必要があります。 つまり、**%XMLAdaptor** を拡張したものである必要があります。

以下のメソッド **SaveContact** は、**Contact** のインスタンスをパラメータとして受け入れ、それをデータベースに保存します。 このメソッドは、新規に作成された **Contact** インスタンスの ID 値を返します。

スタジオ・クラス・エディタを使用して、以下のメソッドを **SOAPService** クラスに追加します。**[ビルド]**→**[コンパイル]** をクリックしてクラスをリコンパイルします。

```
ClassMethod SaveContact(contact As SOAPTutorial.Contact)
As %Integer [ WebMethod ]
{
  set sc = contact.%Save()
  if sc {
    quit contact.%Id()
  }
  else {
    quit 0
  }
}
```

Caché ターミナルを使用した複雑なオブジェクト・コードのテスト: パート 1

Web サービス・テスト・ページを使用して、複雑なオブジェクトを引数として受け入れる Caché Web サービス・メソッドをテストすることはできません。これらのメソッドは、Caché、.NET、Java またはその他の Web サービス対応プログラミング言語を使用して Web サービス・クライアントを構築することでテストできます。Caché ターミナルを使用してメソッドをテストすることもできます。Caché ターミナルを使用して、**SOAPService** の **SaveContact** メソッドをテストする手順を以下に示します。

1. ターミナルを開きます。
2. **ZN** コマンドを使用して Tutorials ネームスペースに切り替えます。ここでは **SOAPService** を Tutorials にインストールしたことを前提としています。

```
USER>ZN "Tutorials"
```

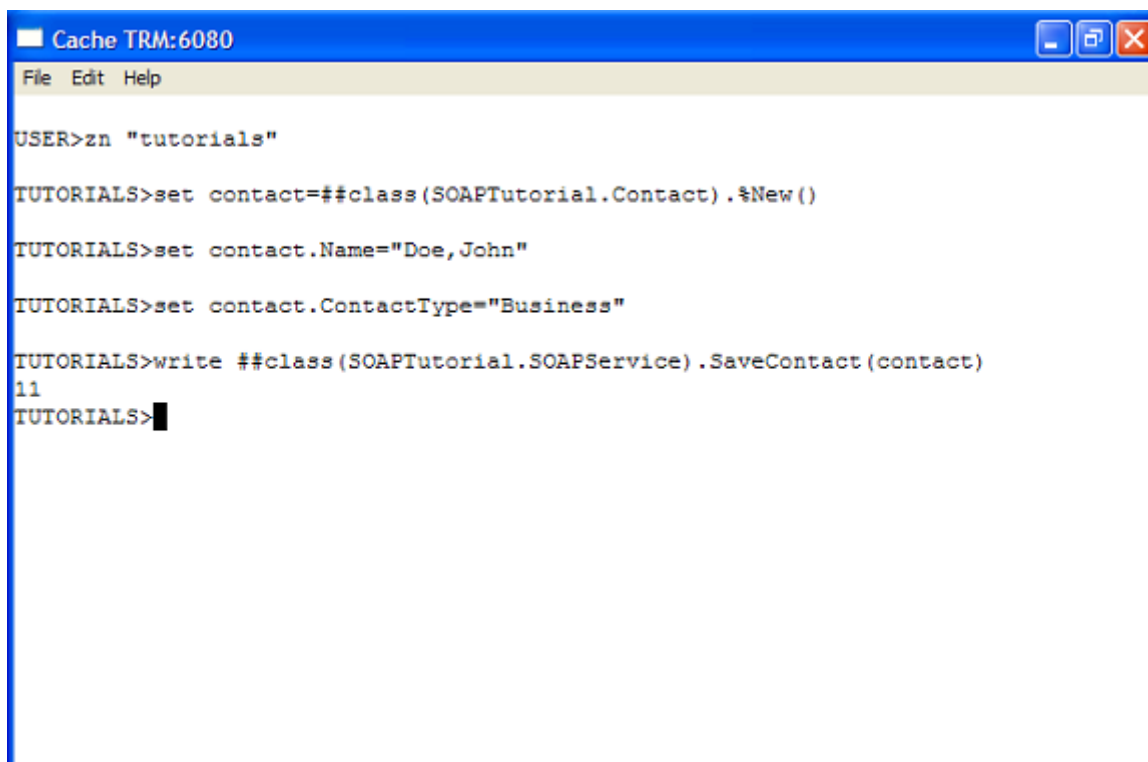
3. 以下のコマンドを入力して Contact の新しいインスタンスを作成し、そのプロパティである Name と ContactType に値を割り当てます。

```
TUTORIALS>Set contact = ##class(SOAPTutorial.Contact).%New()  
TUTORIALS>Set contact.Name = "Doe, John"  
TUTORIALS>Set contact.ContactType = "Business"
```

4. 以下のコマンドを入力して SaveContact メソッドを呼び出し、新規にインスタンス化された Contact オブジェクトをこのメソッドに渡し、メソッドの戻り値である新しい Contact インスタンスの ID 値を書き込みます。

```
TUTORIALS>Write ##class(SOAPTutorial.SOAPService).SaveContact(contact)  
11
```

以下は、このターミナル・セッションの画面です。



```
Cache TRM:6080
File Edit Help

USER>zn "tutorials"

TUTORIALS>set contact=##class(SOAPTutorial.Contact).%New()

TUTORIALS>set contact.Name="Doe, John"

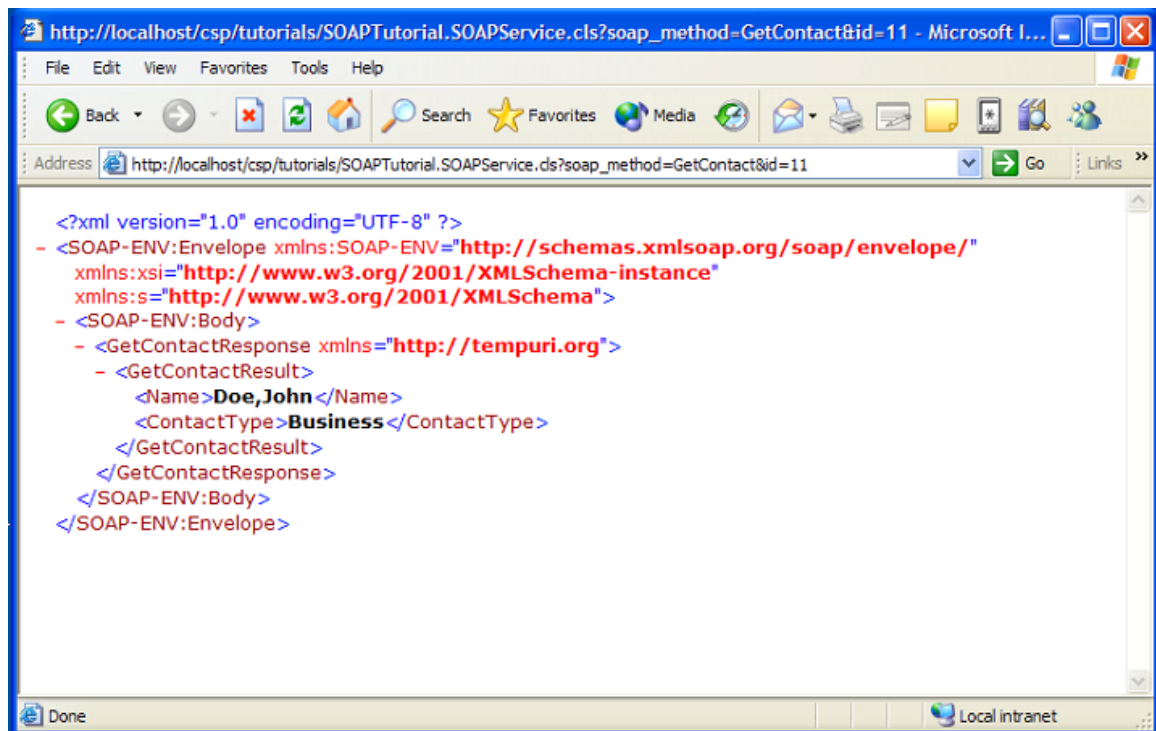
TUTORIALS>set contact.ContactType="Business"

TUTORIALS>write ##class(SOAPTutorial.SOAPService).SaveContact(contact)
11
TUTORIALS>
```

Caché ターミナルを使用した複雑なオブジェクト・コードのテスト: パート 2

Web サービス・テスト・ページと **GetContact** メソッドを使用すると、データベースに **Contact** インスタンスが存在することを確認できます。

1. Web ブラウザで **SOAPService** の Web サービス・テスト・ページを開きます。
2. `[GetContact]` リンクをクリックします。
3. **GetContact** メソッドを呼び出します。そのメソッドに、**SaveContact** メソッドによって返された ID 値を渡します。
4. **SOAPService GetContact** メソッドから SOAP メッセージとして **Contact** インスタンスが返されます。



クエリの追加

クエリを実行するメソッドをWebサービス・クラスに追加することができます。 Cachéには、.NETクライアントがCaché Webサービスを使用するときのために、SOAPを使用してクエリ結果を返すクラス **%XML.DataSet** が用意されています。 **%XML.DataSet** は、 **%Library.ResultSet** のSOAP対応のサブクラスです。 .NETクライアント・アプリケーションでは、返されたSOAPメッセージは自動的に.NET **DataSet** オブジェクトに変換されます。

%XML.DataSet を使用するには、以下の手順を実行します。

1. Webメソッドの返りタイプとして **%XML.DataSet** を指定します。
2. 新しい **%XML.DataSet** インスタンスをインスタンス化します。
3. **Prepare** を使用して、実行する SQL を指定します。 SQL は、コンパイル時または実行時に決定することができます。
4. **SetArgs** を使用して、クエリに必要なパラメータを設定します。
5. メソッドから **%XML.DataSet** インスタンスを返します。 **Execute** は呼び出されないことに注意してください。

次のメソッドでは、以下の基準を満たす各 **PhoneNumber** インスタンスの **PhoneNumberType** 値と **Number** 値が返されます。

- 関連する **Contact** インスタンスに、クエリ・パラメータで指定された **id** 値がある。
- **PhoneNumberType** に値が指定されている。

例えば、このクエリを使用して、 **Contact** 1 の "Business" 電話番号に関するすべての電話番号情報を取得することができます。

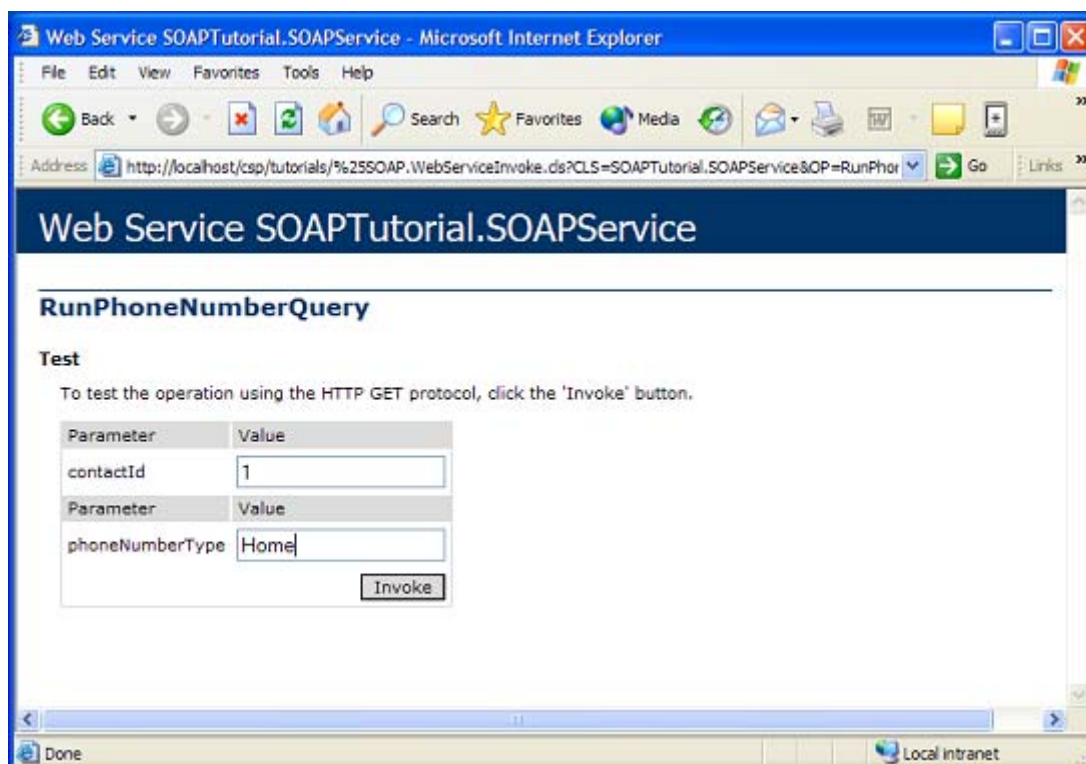
次の Web メソッドを **SOAPTutorial.SOAPService** クラスに追加します。 クラスをリコンパイルします。

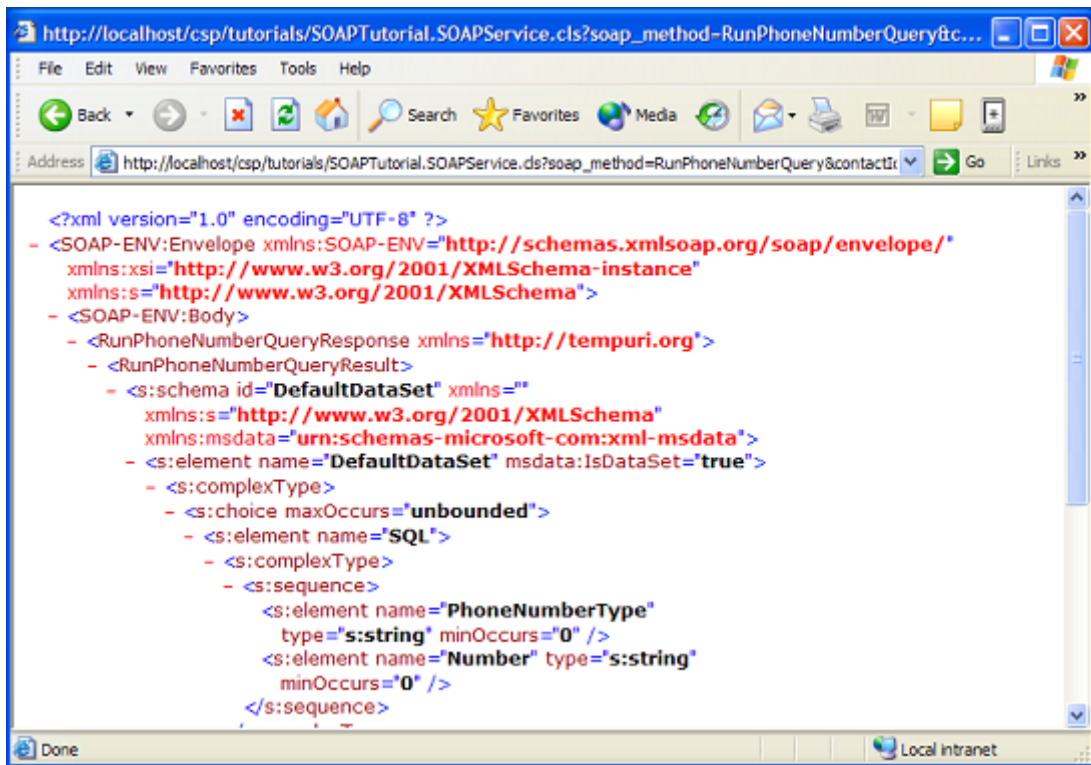
```
ClassMethod RunPhoneNumberQuery(contactId As %String,
phoneNumberType As %String)
As %XML.DataSet [ WebMethod ]
{
  set query = ##class(%XML.DataSet).%New("%DynamicQuery:SQL")
  set sql = "SELECT p.PhoneNumberType, p.Number FROM SOAPTutorial.Contact"
  set sql = sql|_ " as c inner join SOAPTutorial.PhoneNumber as p"
  set sql = sql|_ " on p.Contact=c.ID WHERE c.ID=? AND p.PhoneNumberType=?"
  do query.Prepare(sql)
  do query.SetArgs(contactId, phoneNumberType)
  quit query
}
```

クエリのテスト

Web サービス・テスト・ページを使用して、クエリをテストすることができます。以下の手順を実行します。

1. ブラウザで URL `http://localhost/csp/<ネームスペース名>/SOAPTutorial.SOAPService.cls` を開きます。 <ネームスペース名>は、TUTORIALS など、Contact Management アプリケーションの Caché クラスをインストールしたネームスペースの名前です。
2. [RunPhoneNumberQuery]リンクをクリックします。
3. [contactId]と[phoneNumberType]に有効な値を入力します。 [実行]をクリックします。 テスト・ページに Web メソッドが呼び出されて、指定したパラメータ値がメソッドに渡されます。



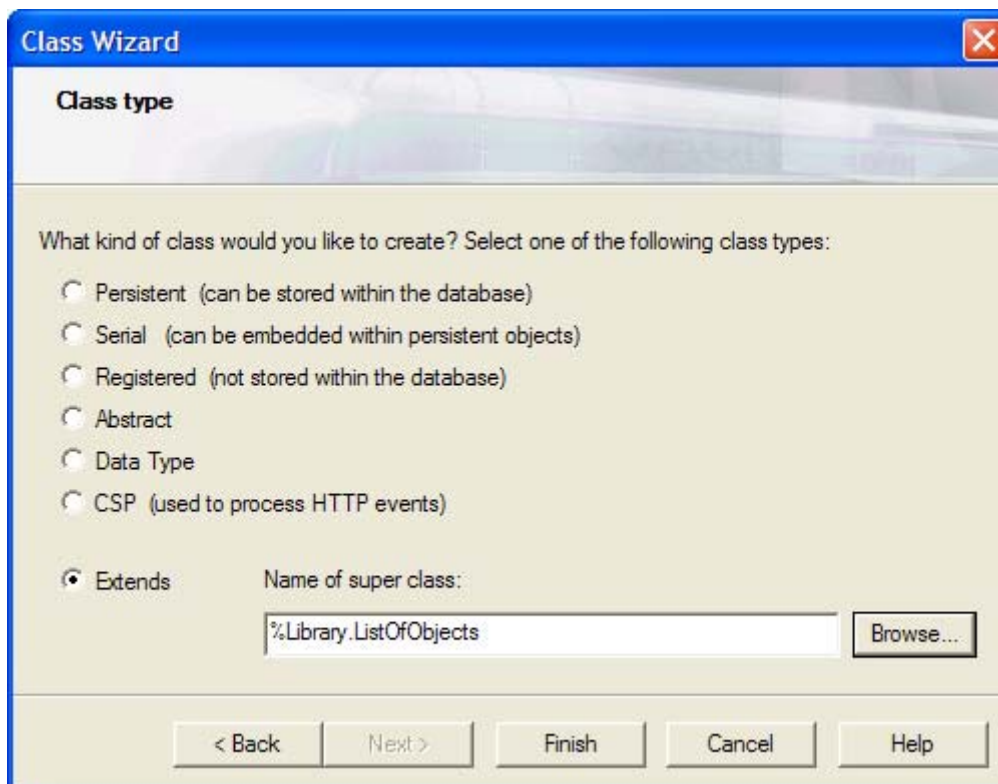
4. **SOAPService** によって、SOAP でエンコードされたクエリ結果が返されます。

Web メソッドから返すコレクションの作成

状況によっては、Webメソッドからオブジェクトのコレクションを返す必要が生じることがあります。[%ListOfObjects](#)または[%ArrayOfObjects](#)インスタンス・タイプなど、標準のCachéコレクション・オブジェクトを単に返すことはできません。その代わりに、Cachéコレクション・タイプを拡張して新しいコレクション・タイプを作成する必要があります。その後、コレクションに格納されているオブジェクトのタイプを指定します。これを行うには、ELEMENTTYPEパラメータの値を適切なタイプに設定します。

Caché スタジオの新規クラス・ウィザードを使用して **ContactList** という名前のコレクション・タイプを作成し、そのコレクション・タイプを使用して **SOAPService** から **Contact** インスタンスのコレクションを返す手順を以下に示します。

1. **[ファイル]**→**[新規作成]**をクリックします。
2. **[一般]**タブをクリックし、**[Cache クラス定義]**アイコンをクリックします。
3. 新しいクラスが **SOAPTutorial** パッケージの一部であり、その名前が **ContactList** であることを指定します。 **[次へ]**をクリックします。
4. **[クラス・タイプ]**画面で**[Extends]**をクリックします。 **[参照]** ボタンをクリックし、[%Library.ListOfObjects](#)をクリックします。



5. **[完了]**をクリックします。

6. クラス・エディタで値"SOAPTutorial.Contact"を ELEMENTTYPE プロパティに割り当てます。これによって、**ContactList** のインスタンスに**Contact** オブジェクトのコレクションが含まれていることが指定されます。

```
Class SOAPTutorial.ContactList Extends %Library.ListOfObjects
[ ClassType = serial, ProcedureBlock ]
{
Parameter ELEMENTTYPE = "SOAPTutorial.Contact";
}
```

7. **[ビルド]**→**[コンパイル]**をクリックして新しいクラスをコンパイルします。

Web メソッドでのコレクションの使用

ContactList オブジェクトを使用して、**Contact** オブジェクトのリストを **SOAPService** Web メソッドから返します。以下のメソッド **GetContactByName** は、指定された文字列で始まる Name 値を持つ **Contact** インスタンスをすべて検索するクエリを実行します。このメソッドは、検索された各インスタンスを **ContactList** オブジェクトに追加し、そのコレクションをクライアントに返します。

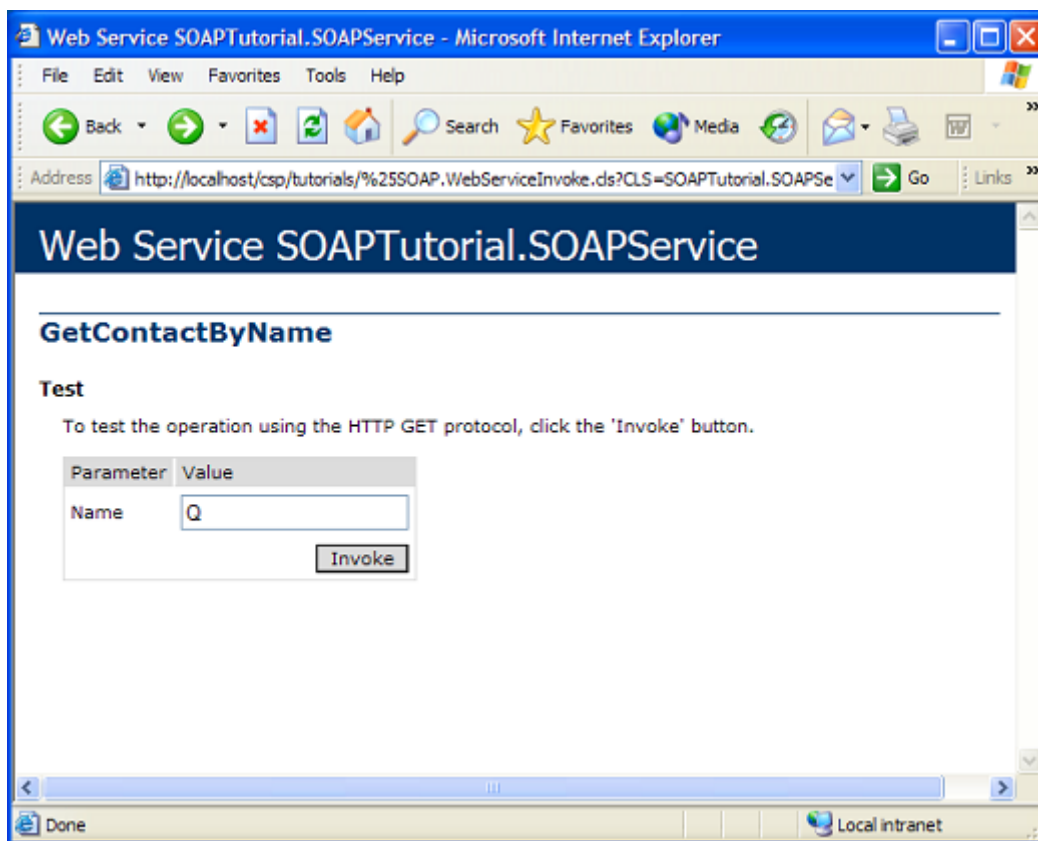
スタジオ・クラス・エディタを使用して、以下のメソッド定義を **SOAPService** に追加し、**[ビルド]**→**[コンパイル]**をクリックして **SOAPService** をリコンパイルします。

```
ClassMethod GetContactByName (Name As %String)
As SOAPTutorial.ContactList [ WebMethod ]
{
  set rs=##class(%ResultSet).%New ("%DynamicQuery:SQL")
  set query="SELECT %Id FROM SOAPTutorial.Contact WHERE Name %STARTSWITH ?"
  set list=##class(SOAPTutorial.ContactList).%New()
  do rs.Prepare(query)
  do rs.Execute(Name)
  while rs.Next()
  {
    set ref=##class(SOAPTutorial.Contact).%OpenId(rs.Get("ID"))
    do list.Insert(ref)
  }
  quit list
}
```

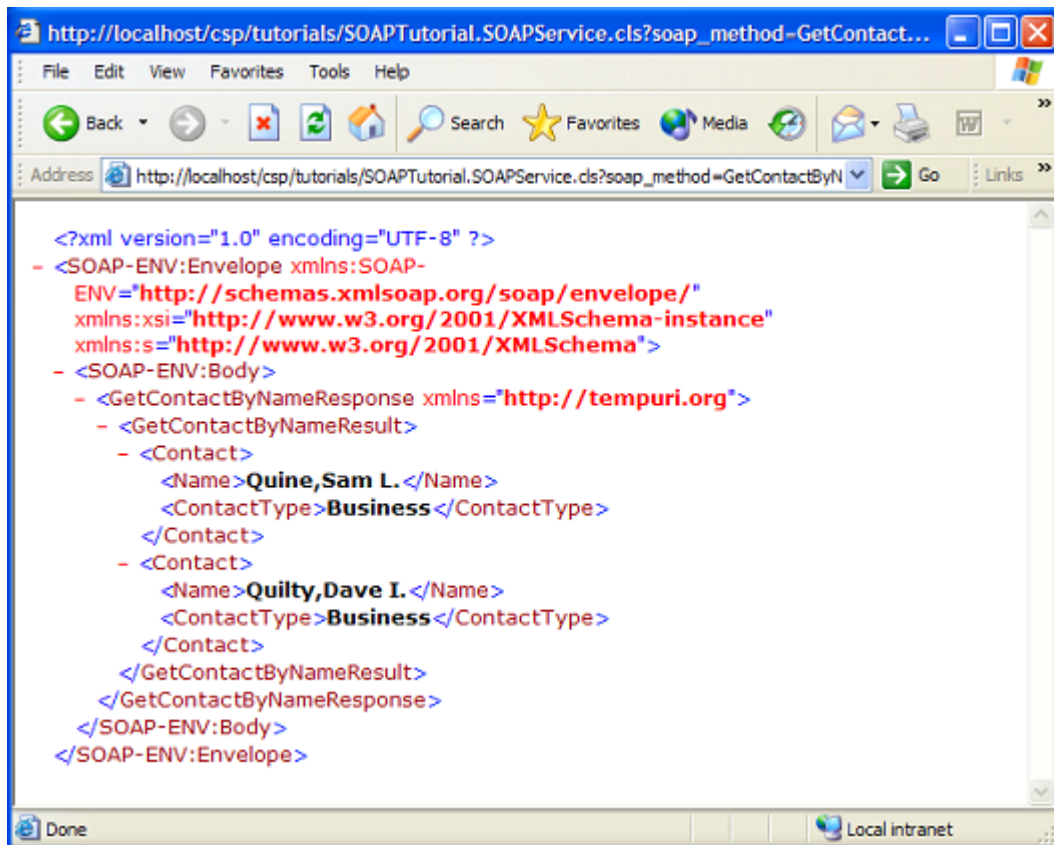
コレクションを返す Web メソッドのテスト

Web サービス・テスト・ページを使用して **GetContactByName** メソッドをテストするには、以下の手順を実行します。

1. ブラウザで URL `http://localhost/csp/<ネームスペース名>/SOAPTutorial.SOAPService.cls` を開きます。 <ネームスペース名>は、TUTORIALS など、Contact Management アプリケーションの Caché クラスをインストールしたネームスペースの名前です。
2. [GetContactByName]リンクをクリックします。
3. **[名前]**の有効な値を入力します。 **[実行]**をクリックします。 テスト・ページに Web メソッドが呼び出されて、指定したパラメータ値がメソッドに渡されます。



4. **SOAPService** によって、SOAP でエンコードされたオブジェクトが返されます。



The screenshot shows a web browser window with the address bar containing the URL: `http://localhost/csp/tutorials/SOAPTutorial.SOAPService.cls?soap_method=GetContactByN...`. The browser's address bar also shows the method `GetContactByN`. The main content area displays the following XML response:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-
  ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:s="http://www.w3.org/2001/XMLSchema">
- <SOAP-ENV:Body>
- <GetContactByNameResponse xmlns="http://tempuri.org">
- <GetContactByNameResult>
- <Contact>
  <Name>Quine, Sam L.</Name>
  <ContactType>Business</ContactType>
</Contact>
- <Contact>
  <Name>Quilty, Dave I.</Name>
  <ContactType>Business</ContactType>
</Contact>
</GetContactByNameResult>
</GetContactByNameResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The status bar at the bottom of the browser window shows "Done" and "Local intranet".

演習

演習 1: Web メソッドを **SOAPTutorial.SOAPService** に追加します。それによって、**PhoneNumber** オブジェクトがクライアントの指定した **Contact** オブジェクトに追加されます。このメソッドは以下の要件を満たしている必要があります。

- メソッドの名前は **AddPhoneNumber** です。
- **SOAPTutorial.PhoneNumber** オブジェクトと (**SOAPTutorial.Contact** の ID を表す) **%String** 値の両方をクライアントから受け取ります。指定された ID 値を持つ **Contact** の **PhoneNumbers** 配列に **PhoneNumber** オブジェクトを追加します。
- **Contact %Save** メソッドを使用して、変更をデータベースに保存します。
- 処理が成功した場合は **Contact** ID 値を返し、そうでない場合は 0 を返します。
- Caché ターミナルを使用してメソッドをテストします。Web サービス・テスト・ページを使用してメソッドをテストできない理由を調べます。

演習 2: Web メソッドを **SOAPTutorial.SOAPService** に追加します。それによって、クライアントが指定した **ContactType** の値を持つすべての **Contact** インスタンスが返されます。このメソッドは以下の要件を満たしている必要があります。

- メソッドの名前は **GetContactByType** です。
- **ContactType** 値を指定する **%String** の引数タイプを受け入れます。
- 指定された **ContactType** 値を持つ **Contact** インスタンスを含むオブジェクトのリストを返します。メソッドが **%XML.DataSet** オブジェクトではなく、オブジェクトのリストを返すことを確認してください。
- ダイナミックSQLを使用して、指定された **ContactType** 値を持つ **Contact** インスタンスを取得します。ダイナミックSQLの詳細は、Cachéドキュメント: [\[Caché開発ガイド\]-\[Caché SQLの使用法\]-\[ダイナミックSQL\]](#) を参照してください。
- Web サービス・テスト・ページを使用してメソッドをテストします。

要約

このチュートリアルの第 I 章では、以下について学習しました。

- 単純な Caché Web サービス・プロデューサ・アプリケーションの作成
- Caché によって自動的に生成された Caché Web サービス・プロデューサ・アプリケーションの WSDL ドキュメントの表示
- Caché Web サービス・テスト・ページと Caché ターミナルの両方を使用した Caché Web サービス・プロデューサ・アプリケーションのテスト
- Caché Web サービス・プロデューサ・アプリケーションにおけるクエリ、複雑なオブジェクト、およびオブジェクトのコレクションの使用

はじめに

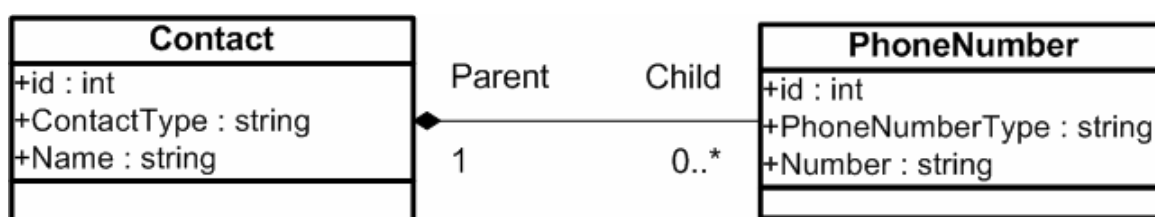
このチュートリアルの第Ⅱ章では、Web サービスを利用する Caché クライアントの作成に焦点をあてて説明します。Web サービスには、Delphi、Java、.NET などの任意の言語とプラットフォームを使用できます。この章の学習を終えると、以下のことを実行できるようになります。

- Caché SOAP クライアント・ウィザードを使用した Web サービスとの通信用のプロキシ・クラスの作成
- SOAP クライアント・ウィザードで生成されたプロキシ・クラスを使用した Web サービスの呼び出し
- Web サービスから Caché Web サービス・クライアントに返された結果の処理

Web サービスのサンプル・プロデューサ・アプリケーション

このチュートリアルの例と演習では、単純な Contact Management Web サービス・プロデューサ・アプリケーションで Web サービス・コンシューマを作成します。アプリケーションのファイルは、Caché インストールの一部として提供されています。作業する Caché ネームスペースにそれらのファイルをインストールする必要があります。すべての例と演習用ファイルのインストール情報へのリンクは、下記のメモを参照してください。

アプリケーションのデータ・モデルは、**Contact** と **PhoneNumber** という 2 つのクラスで構成されています。これらのクラスは、連絡先とその電話番号を表します。次の図は、アプリケーションのオブジェクト・モデルを表しています。



Contact には、以下のプロパティがあります。

- *id* — オブジェクト ID。 **Contact** の各インスタンスに対して一意です。値は、Caché によって自動的に割り当てられます。Caché スタジオのクラス・エディタやインスペクタには、このプロパティは表示されません。
- *ContactType* — 連絡先のタイプを表します。有効な値は、“Business”と“Personal”のみです。
- *Name* — 連絡先の名前を表します。値は、任意の **%String** です。

PhoneNumber には、以下のプロパティがあります。

- *id* — オブジェクト ID。 **PhoneNumber** の各インスタンスに対して一意です。値は、Caché によって自動的に割り当てられます。Caché スタジオのクラス・エディタやインスペクタには、このプロパティは表示されません。
- *PhoneNumberType* — 電話番号のタイプを表します。有効な値は、“Business”、“Home”、“Fax”および“Mobile”のみです。
- *Number* — 電話番号を表します。値は、任意の **%String** です。形式に関する制限はありません。

これらのプロパティのサポートに加えて、2つのクラス間には親子リレーションシップが確立されています。つまり、各 **Contact** (親クラス) オブジェクトに任意の数の **PhoneNumber** (子クラス) オブジェクトを含めることができます。ただし、各 **PhoneNumber** オブジェクトは1つの **Contact** のみに含める必要があります。**PhoneNumber** オブジェクトは、**Contact** と独立して存在することはできません。また、**PhoneNumber** オブジェクトを複数の **Contact** に含めることはできません。このリレーションシップは、次の2つのプロパティによって表されます(図には示されていません)。

- *PhoneNumbers* — **Contact** のプロパティ。 **Contact** に含まれる **PhoneNumber** (子) オブジェクトのコレクションを表します。
- *Contact* — **PhoneNumber** のプロパティ。 **PhoneNumber** の親である **Contact** オブジェクトへの参照を含みます。

Note:

Caché アプリケーションの配置とインストールの詳細は、[AppendixA: 例と演習ファイルのインストール] を参照してください。 Caché アプリケーションをインストールしたら、サンプル・データを入力します。 アプリケーションへのデータの入力の詳細は、[AppendixB: サンプル・アプリケーションのコンパイルと生成] を参照してください。 このアプリケーションは、第 I 章の例と演習で作成するアプリケーションと同じです。

Web サービスのバックグラウンド

Web サービスは関連する一連のメソッドで、インターネットなどのネットワークを介して呼び出すことができます。Web サービスとそのクライアントは、単純な非占有プロトコルのみを使用して情報を交換します。そのため、場所、プラットフォーム、またはプログラミング言語に関係なく、相互に通信することができます。Web サービス・プロトコルは基本的に、リモート・プロシージャ・コール (RPC) を実行するためのプラットフォーム独立の方法を提供します。

次のテーブルは主な Web サービス・プロトコルとその目的の一覧で、各プロトコルについて簡単に説明しています。

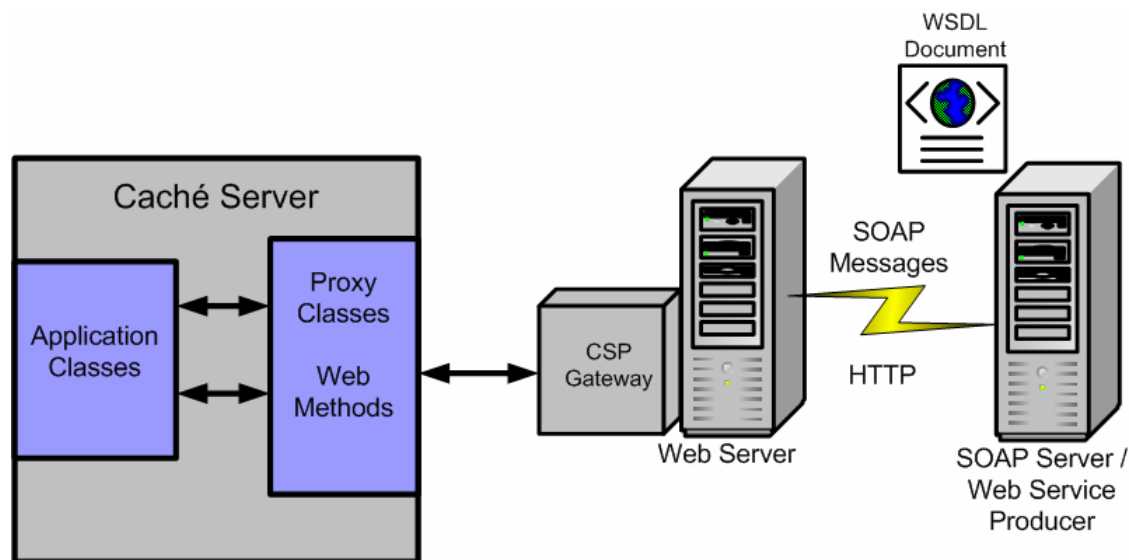
	目的	説明
HTTP	転送	ハイパーテキスト転送プロトコル(HyperText Transfer Protocol)。インターネットで使用される基本的なネットワーク・プロトコル。
SOAP	パッケージ化	シンプル・オブジェクト・アクセス・プロトコル(Simple Object Access Protocol)。Web サービス・メソッドとクライアント間で送信されるメッセージをエンコードするための XML ベースのプロトコル。Web サービス・メソッドに渡される引数とメソッドからクライアントに返される値をエンコードします。
WSDL	説明	Web サービス記述言語(Web Service Description Language)。Web サービスを記述するための XML ベースのプロトコル。WSDL ドキュメントには、Web サービスのメソッドの署名に加え、Web サービスに関連するデータ型についてのその他の情報が記述されます。WSDL を参照することで、クライアント・コードはその Web サービスのタイプとメソッドを使用することができます。
UDDI	検出	Universal Description, Discovery, and Integration。アプリケーションが Web サービスの記述の検出に使用できる Web サービス・レジストリを作成するための XML ベースのプロトコル。

概要: Web サービス・コンシューマとしての Caché

Cachéは、Webサービスを完全にサポートしています。さらに、Webサービスのサポートが Cachéに組み込まれているため、追加のミドルウェアは必要ありません。Cachéには、各種のウィザードと`%SOAP.WSDL.Reader`クラスが用意されています。このクラスは、SOAP Webサービス・プロデューサ用のプロキシ・クラスを WSDLドキュメントに基づいて自動的に生成します。Cachéは、SOAPからオブジェクトおよびオブジェクトからSOAPへの変換をすべて自動的に実行するため、CachéアプリケーションとWebサービス間の通信が完全に透過的になります。

以下に、Cachéアプリケーションが Web サービスを利用するときのプロセスの一部を示します。

1. Cachéは、プロデューサのWSDLドキュメントに基づいて、そのローカル・プロキシ・クラスを自動的に生成します。
2. Cachéアプリケーション・クラスは、標準的なオブジェクト指向手法を使用して、プロキシ・クラスのメソッドを呼び出します。
3. Cachéは、メソッドの呼び出しに基づいて SOAP メッセージを作成し、HTTP を使用してプロデューサに送信します。
4. SOAP サーバは要求を処理し、SOAP でエンコードされた応答を返します。
5. Cachéは、SOAP の応答を Caché オブジェクトに変換します。
6. ローカル・プロキシ・クラスは、標準的な Caché 値を Cachéアプリケーションに返します。



Note:

この図は、CachéクライアントとWebサービスの論理構成を示しています。すべての論理要素を同一の物理マシンに配置することができます。

Caché Web サービス・プロキシ・クラスの要件

外部 Web サービスのプロキシとして Caché が使用するクラスは、以下の要件を満たしている必要があります。

- クラスは [%SOAP.WebClient](#) を拡張したものである必要があります。
- Web サービス・プロデューサによってクラスに投影されるメソッドは、インスタンス・メソッドであり、WebMethod 属性でタグ付けされている必要があります。

[%SOAP.WebClient](#) を自動的に拡張したクラスは、以下のパラメータの値を含みます。これらのパラメータは、Web サービスの重要な特性を示します。

パラメータ	説明と値
LOCATION	このパラメータの値は、クライアントが Web サービスへのアクセスに使用する URL です。この URL は、Web サービスの WSDL ドキュメントに含まれています。
NAMESPACE	このパラメータの値は URI です。これはサービスにネームスペースを提供して、その名前が別のサービスの名前と競合しないようにします。既定では、値 <code>http://tempuri.org</code> が割り当てられます。開発者は通常、Web サービス開発時にこの値を使用します。導入時には、これを変更する必要があります。
SERVICENAME	このパラメータの値は、サービスにクライアントの名前を提供します。これは、有効な識別子でなければなりません。つまり、文字から開始し、英数字のみを使用する必要があります。

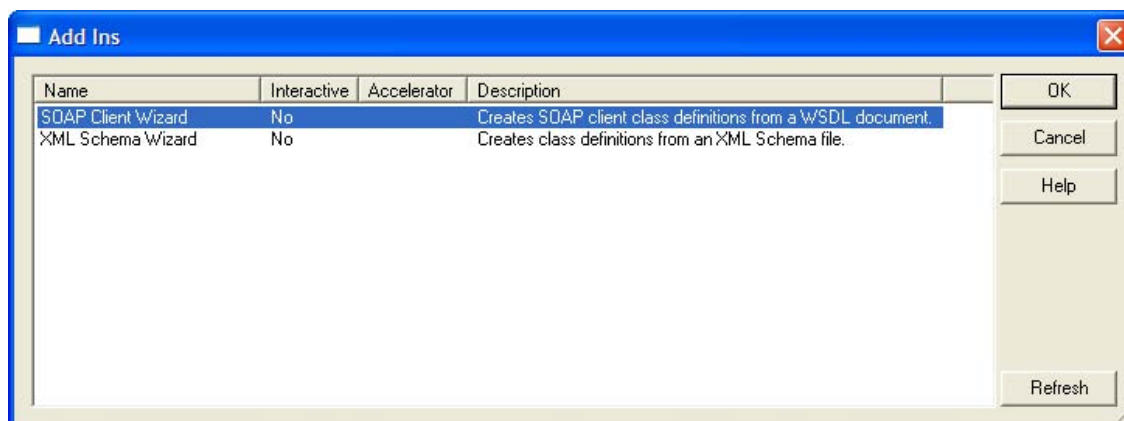
ほとんどの状況では、Caché で Web サービス・プロデューサの WSDL ドキュメントを使用して、プロキシ・クラスを自動的に生成します。プロキシ・クラスは、これらの要件をすべて満たします。

ウィザードを使用したプロキシ・クラスの作成：パート 1

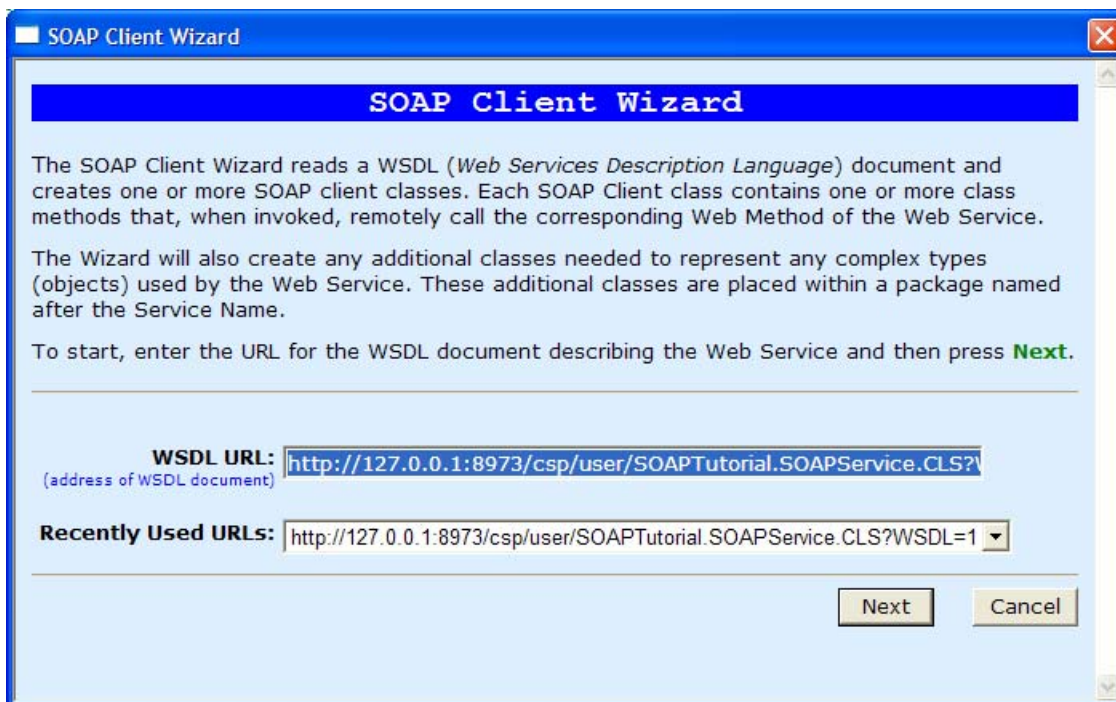
Caché Web サービス・コンシューマが Web サービス・プロデューサと通信するには、プロデューサのプロキシ・クラスを必要とします。これらのプロキシ・クラスは、Web サービス・プロデューサの WSDL ドキュメントの仕様に従って生成されます。Caché Web サービス・クライアント・ウィザードでは、WSDL ドキュメントを使用してプロキシ・クラスを自動的に生成します。

Caché Web サービス・クライアント・ウィザードを使用して、サンプルの Web サービス・プロデューサ・アプリケーションの Caché クライアントを作成するには、以下の手順を実行します。

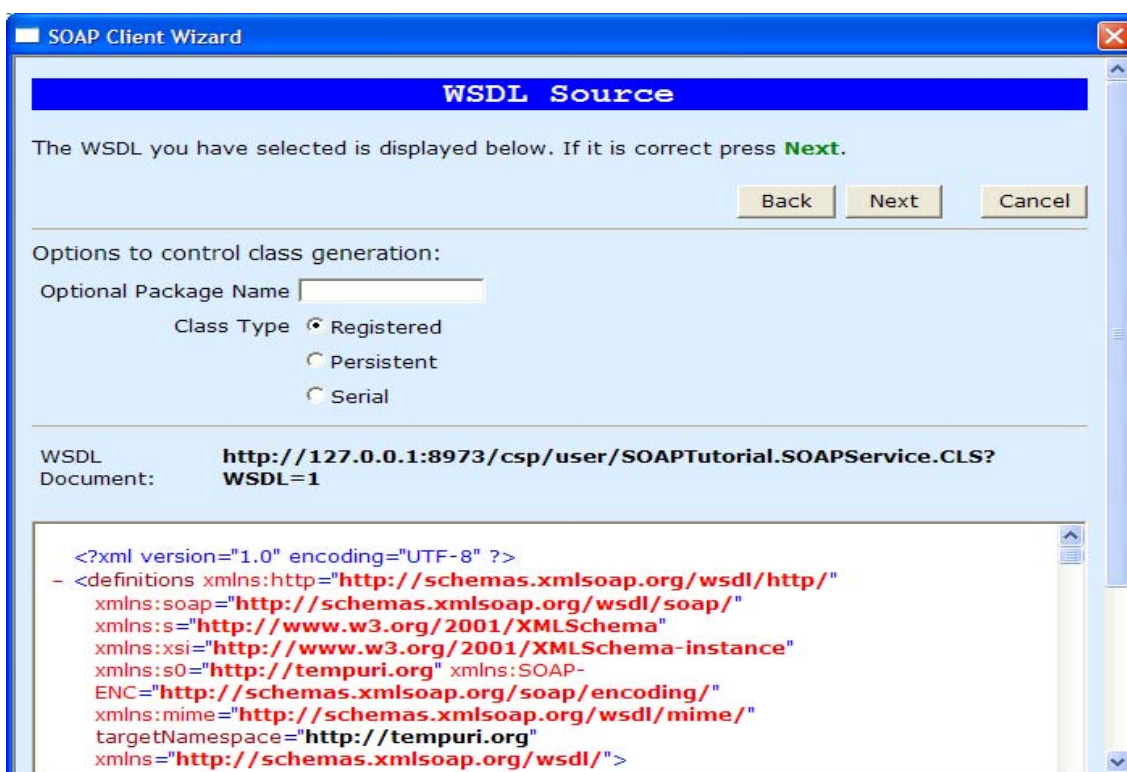
1. Caché スタジオを開きます。サンプル・プロデューサ・アプリケーションがインストールされているネームスペースとは別のネームスペースに切り替えます。これによって、プロデューサとコンシューマ間のリモート通信をシミュレートできるようになります。
2. **[ツール]**→**[アドイン]**→**[アドイン]** をクリックします。 **[アドイン]** リストが表示されます。**[SOAP クライアントウィザード]** をクリックします。



- SOAP クライアント・ウィザードの最初の画面で、サンプル・プロデューサ・アプリケーションの WSDL ドキュメントの URL を入力します。[次へ]をクリックします。WSDL ドキュメントの URL は、`http://localhost/csp/<サーバ・ネームスペース>/SOAPTutorial.SOAPService.cls?WSDL=1`です。<サーバ・ネームスペース>は、アプリケーションがインストールされている Caché ネームスペースです。



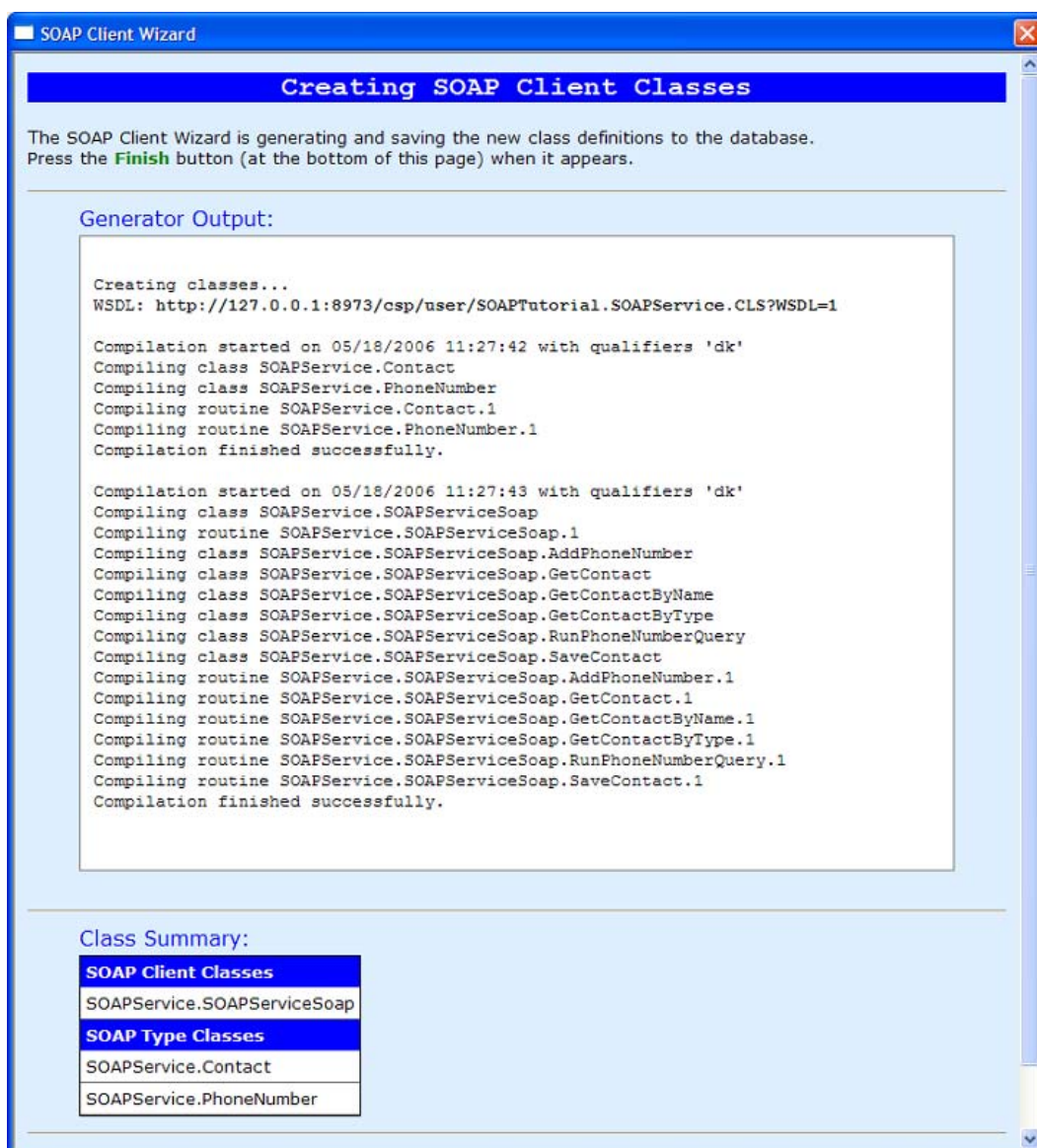
- SOAP クライアント・ウィザードに、サンプル・プロデューサ・アプリケーションの WSDL ドキュメントが表示されます。[クラスの生成を制御するオプション]の見出しの下は既定値のままにしておきます。[オプションのパッケージ名]フィールドは空白にし、[登録クラス]は選択された状態にしておきます。[次へ]をクリックします。



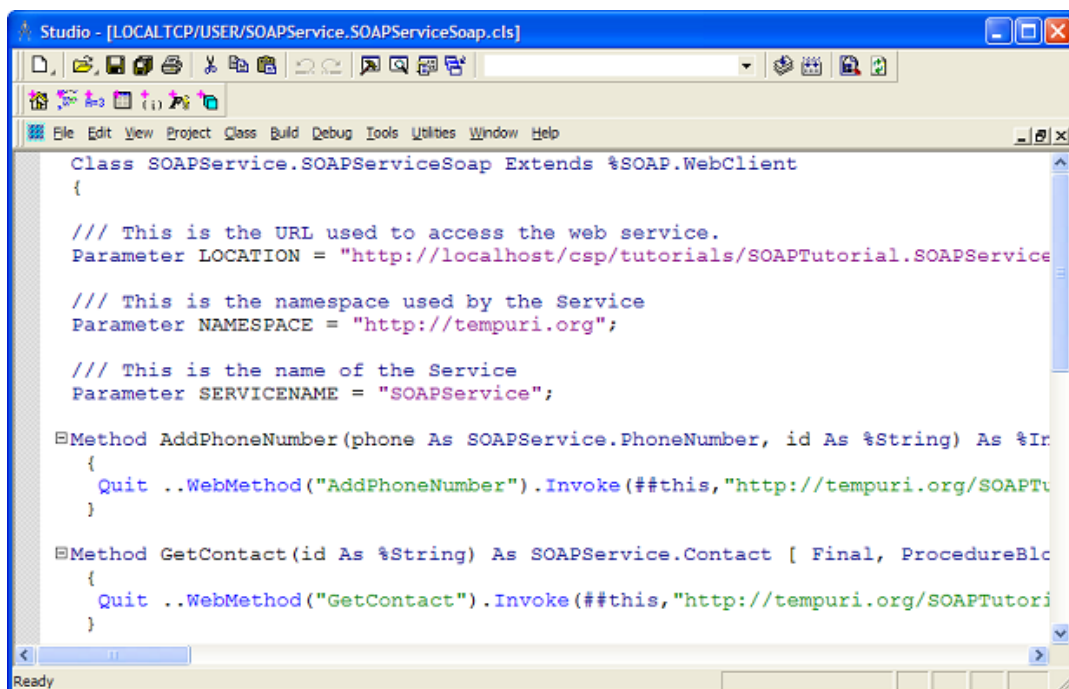
ウィザードを使用したプロキシ・クラスの作成: パート 2

Web サービス・クライアント・ウィザードを使用してプロキシ・クラスの作成を完了するには、以下の手順を実行します。

1. WSDL ドキュメントが表示されている画面で[次へ]をクリックすると、プロキシ・クラスが生成されます。この場合、次の 3 つのクラスが生成されます。
 1. **SOAPService.SOAPServiceSoap** — SOAP クライアント・クラス
 2. **SOAPService.Contact** — SOAP タイプ・クラス
 3. **SOAPService.PhoneNumber** — SOAP タイプ・クラス



2. **[完了]**をクリックします。スタジオが開いているネームスペースにプロキシ・クラスがインストールされます。この場合、クラスは **SOAPService** パッケージにインストールされます。
3. クライアント・アプリケーションではプロデューサ・アプリケーションのプロキシとして **SOAPService.SOAPServiceSoap** を使用します。スタジオで **SOAPService.SOAPServiceSoap** を開きます。**SOAPTutorial.SOAPService** の各メソッドのプロキシが含まれています。各プロキシ・メソッドは、**SOAPTutorial.SOAPService** 内の元のメソッドと同じ名前、パラメータ・リスト、および返りタイプを持っています。プロキシ・メソッドは、インスタンス・メソッドです。これらを使用するには、**SOAPService.SOAPServiceSoap** のインスタンスを作成する必要があります。



```
Studio - [LOCALTCP/USER/SOAPService.SOAPServiceSoap.cls]
File Edit View Project Class Build Debug Tools Utilities Window Help
Class SOAPService.SOAPServiceSoap Extends %SOAP.WebClient
{
    /// This is the URL used to access the web service.
    Parameter LOCATION = "http://localhost/csp/tutorials/SOAPTutorial.SOAPService";
    /// This is the namespace used by the Service
    Parameter NAMESPACE = "http://tempuri.org";
    /// This is the name of the Service
    Parameter SERVICENAME = "SOAPService";
    @Method AddPhoneNumber(phone As SOAPService.PhoneNumber, id As %String) As %Integer
    {
        Quit ..WebMethod("AddPhoneNumber").Invoke(##this,"http://tempuri.org/SOAPTutorial.SOAPService/AddPhoneNumber");
    }
    @Method GetContact(id As %String) As SOAPService.Contact [ Final, ProcedureBlock ]
    {
        Quit ..WebMethod("GetContact").Invoke(##this,"http://tempuri.org/SOAPTutorial.SOAPService/GetContact");
    }
}
```

プロキシ・クラスのテスト

Web サービスをテストするには、Caché ターミナルを使用して **SOAPService.SOAPServiceSoap** のインスタンスを作成します。

1. ターミナルを開き、**ZN** コマンドを使用して、**SOAPService.SOAPServiceSoap** がインストールされているネームスペースに切り替えます。この場合、プロキシは **USER** にインストールされています。

```
TUTORIALS> ZN "USER"  
USER>
```

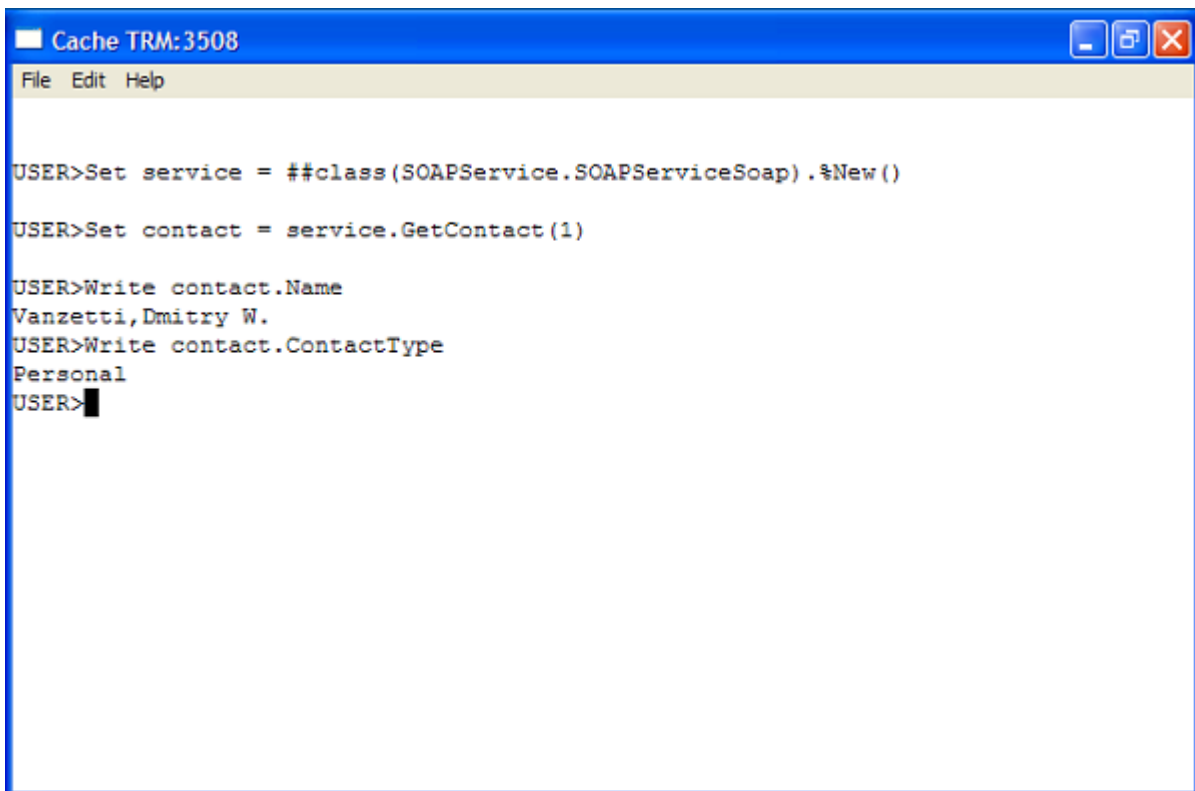
2. **SOAPService.SOAPServiceSoap** のインスタンスを作成します。 **GetContact** を使用して、**Contact** インスタンスを取得します。

```
USER> Set service = ##class(SOAPService.SOAPServiceSoap).%New()  
USER> Set contact = service.GetContact(1)
```

3. **Contact** インスタンスの **Name** プロパティと **ContactType** プロパティの値を **WRITE** します。

```
USER> Write contact.Name  
Vanzetti, Dmitry W.  
USER> Write contact.ContactType  
Personal
```

以下は、このターミナル・セッションの画面です。



```
Cache TRM:3508
File Edit Help

USER>Set service = ##class(SOAPService.SOAPServiceSoap).%New()

USER>Set contact = service.GetContact(1)

USER>Write contact.Name
Vanzetti,Dmitry W.
USER>Write contact.ContactType
Personal
USER>
```

Caché ターミナルを使用したプロキシ・クラスの作成とテスト

Web サービス・クライアント・ウィザードを使用するほかに、Cachéターミナルと [%SOAP.WSDL.Reader](#) クラスを使用してプロキシ・クラスを作成することもできます。

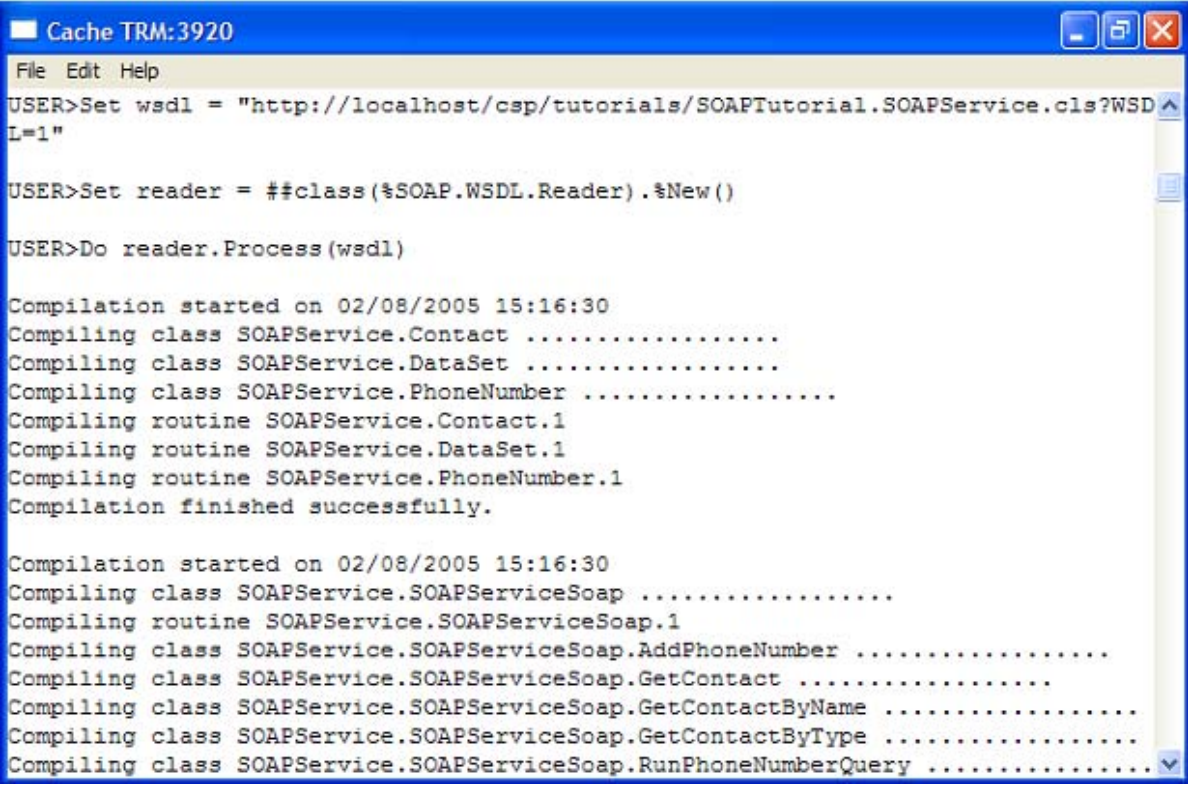
1. Caché ターミナルを開きます。使用するネームスペースはどれでもかまいません。
2. Web サービス・プロデューサの WSDL ドキュメントの URL を変数に割り当てます。

```
USER> Set
wsdl="http://localhost/csp/tutorials/SOAPTutorial.SOAPService.cls?WSDL=1"
```

3. [%SOAP.WSDL.Reader](#) のインスタンスを作成し、その Process メソッドを使用して Web サービスのプロキシ・クラスを生成します。

```
USER> Set reader=##class(%SOAP.WSDL.Reader).%New()
USER> Do reader.Process(wsdl)
```

以下は、ターミナル・セッションの画面です。



```
Cache TRM:3920
File Edit Help
USER>Set wsdl = "http://localhost/csp/tutorials/SOAPTutorial.SOAPService.cls?WSDL=1"
USER>Set reader = ##class(%SOAP.WSDL.Reader).%New()
USER>Do reader.Process(wsdl)

Compilation started on 02/08/2005 15:16:30
Compiling class SOAPService.Contact .....
Compiling class SOAPService.DataSet .....
Compiling class SOAPService.PhoneNumber .....
Compiling routine SOAPService.Contact.1
Compiling routine SOAPService.DataSet.1
Compiling routine SOAPService.PhoneNumber.1
Compilation finished successfully.

Compilation started on 02/08/2005 15:16:30
Compiling class SOAPService.SOAPServiceSoap .....
Compiling routine SOAPService.SOAPServiceSoap.1
Compiling class SOAPService.SOAPServiceSoap.AddPhoneNumber .....
Compiling class SOAPService.SOAPServiceSoap.GetContact .....
Compiling class SOAPService.SOAPServiceSoap.GetContactByName .....
Compiling class SOAPService.SOAPServiceSoap.GetContactByType .....
Compiling class SOAPService.SOAPServiceSoap.RunPhoneNumberQuery .....
```

プロキシ・クラス **SOAPService.SOAPServiceSoap** のインスタンスを作成してテストします。

```
USER> Set service = ##class(SOAPService.SOAPServiceSoap).%New()
USER> Set contact = service.GetContact(1)
USER> Write contact.Name
Vanzetti, Dmitry W.
USER> Write contact.ContactType
Personal
```

演習

演習 1: Cachéターミナルと`%SOAP.WSDL.Reader`クラスを使用して、xmethods.netにあるTemperature Service Webサービス・プロデューサのプロキシ・クラスを作成します。 Cachéターミナルを使用して、これらのプロキシ・クラスとWebサービスをテストします。

- WSDLドキュメントの URL は
http://www.xmethods.net/sd/2001/TemperatureService.wsdl です。
- `TemperatureService.TemperaturePort` クラスの `getTemp` は、クライアントが指定した郵便番号の地域の現在の気温を返します。
- サービスをテストするには、`getTemp` に米国の有効な郵便番号を表す文字列(02142 など)を渡します。

要約

このチュートリアルの第Ⅱ章では、以下について学習しました。

- Caché SOAP クライアント・ウィザードを使用した、外部 Web サービス・プロデューサとの通信用のプロキシ・クラスを作成
- SOAP クライアント・ウィザードで生成されたプロキシ・クラスを使用した外部 Web サービス・メソッドの呼び出し
- 外部 Web サービス・プロデューサから Caché Web サービス・コンシューマに返された結果の処理

はじめに

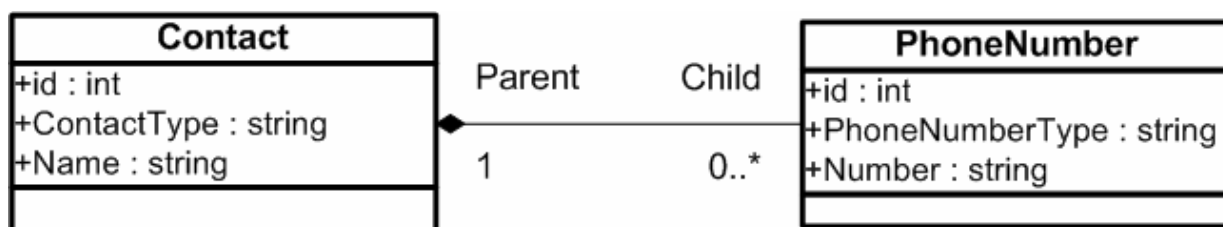
このチュートリアルの第Ⅲ章では、SOAP フォルトを使用した Web サービスのエラー処理に焦点をあてて説明します。ここでは、Caché Web サービス・クライアントと Caché Web サービス・プロデューサの両方の観点からエラー処理を見ていきます。この章の学習を終えると、以下のことを実行できるようになります。

- Caché Web サービス・プロデューサ・クラスへの SOAP フォルト生成コードの追加
- SOAP フォルトを SOAP エンベロープに入れて返す Web サービス・プロデューサ・アプリケーションへのコードの追加
- Web サービス・プロデューサによって返される SOAP フォルトを処理する Caché Web サービス・コンシューマ・クラスへのコードの追加

Web サービスのコンシューマ・アプリケーションとプロデューサ・アプリケーション

このチュートリアルの例と演習では、エラー処理コードを Web サービス・プロデューサ・アプリケーションと Web サービス・コンシューマ・アプリケーションの両方に追加します。Web サービス・プロデューサ・アプリケーションには、単純な Caché Contact Management アプリケーションが含まれています。これらのアプリケーションのファイルは、Caché インストールの一部として提供されています。作業する Caché ネームスペースにこれらのファイルをインストールする必要があります。すべての例と演習用ファイルのインストール情報へのリンクは、下記のメモを参照してください。

Contact Management アプリケーションのデータ・モデルは、**Contact** と **PhoneNumber** という 2 つのクラスで構成されています。これらのクラスは、連絡先とその電話番号を表します。次の図は、アプリケーションのオブジェクト・モデルを表しています。



Contact には、以下のプロパティがあります。

- id — オブジェクト ID。 **Contact** の各インスタンスに対して一意です。値は、Caché によって自動的に割り当てられます。Caché スタジオのクラス・エディタやインスペクタには、このプロパティは表示されません。
- ContactType — 連絡先のタイプを表します。有効な値は、“Business”と“Personal”のみです。
- Name — 連絡先の名前を表します。値は、任意の **%String** です。

PhoneNumber には、以下のプロパティがあります。

- id — オブジェクト ID。 **PhoneNumber** の各インスタンスに対して一意です。値は、Caché によって自動的に割り当てられます。Caché スタジオのクラス・エディタやインスペクタには、このプロパティは表示されません。
- PhoneNumberType — 電話番号のタイプを表します。有効な値は、“Business”、“Home”、“Fax”および“Mobile”のみです。
- Number — 電話番号を表します。値は、任意の **%String** です。形式に関する制限はありません。

これらのプロパティのサポートに加えて、2つのクラス間には親子リレーションシップが確立されています。つまり、各 **Contact** (親クラス)オブジェクトに任意の数の **PhoneNumber**(子クラス)オブジェクトを含めることができます。ただし、各 **PhoneNumber** オブジェクトは1つの **Contact** のみに含める必要があります。**PhoneNumber** オブジェクトは、**Contact** と独立して存在することはできません。また、**PhoneNumber** オブジェクトを複数の **Contact** に含めることはできません。このリレーションシップは、次の2つのプロパティによって表されます(図には示されていません)。

- PhoneNumbers — **Contact** のプロパティ。 **Contact** に含まれる **PhoneNumber**(子)オブジェクトのコレクションを表します。
- Contact — **PhoneNumber** のプロパティ。 **PhoneNumber** の親である **Contact** オブジェクトへの参照を含みます。

Note:

Caché アプリケーションの配置とインストールの詳細は、[AppendixA: 例と演習ファイルのインストール] を参照してください。 Caché アプリケーションをインストールしたら、サンプル・データを入力します。 アプリケーションへのデータの入力の詳細は、[AppendixB: サンプル・アプリケーションのコンパイルと生成] を参照してください。

概要: SOAP フォルト

SOAPの仕様では、Webサービス・プロデューサからエラー・メッセージを返すための標準メカニズムとしてフォルト要素が提供されています。この要素には、フォルトの原因となった特定のエラーに関する役立つ情報が含まれています。Cachéでは、[%SOAP.Fault](#)オブジェクトを使用してWebサービス・エラーを処理します。Caché Webサービス・プロデューサは、これらのオブジェクトを標準のSOAPフォルト要素に自動的に変換してからクライアントに返します。Caché Webサービス・コンシューマは、プロデューサから受け取ったSOAPフォルト要素を[%SOAP.Fault](#)オブジェクトに自動的に変換します。

[%SOAP.Fault](#)には、以下のプロパティがあります。

プロパティ	説明
detail	フォルトの原因に関する情報を含みます。
faultactor	フォルトを生成したサービスの URI を含みます。
<i>faultcode</i>	<p>SOAP の仕様で定義されている SOAP フォルト・コードを含みます。Caché には、フォルト・コード要素用に以下のマクロが用意されています。</p> <ul style="list-style-type: none"> \$\$\$FAULTVersionMismatch — クライアントとサーバ間の SOAP バージョンに互換性がない。 \$\$\$FAULTMustUnderstand — サーバが理解できない MustUnderstand 属性がヘッダに含まれている。 \$\$\$FAULTClient — クライアントが誤った/不完全な要求をした。 \$\$\$FAULTServer — サーバ側エラー。
faultstring	フォルトの原因に関する、人間が読める形式の説明を含みます。

SOAP フォルトの生成

Web サービス・プロデューサ・コードでは、エラーが生成されるすべての場所で SOAP フォルトを生成できる必要があります。SOAP フォルト生成コードが何度も実行される事態を回避するために、フォルトを生成する 1 つのメソッドを作成できます。このメソッドは、プロデューサ・コードのエラーが生成されるすべての場所で呼び出すことができます。

以下のメソッドは、エラー・コードと詳細をパラメータとして受け取り、適切な SOAP フォルトを生成します。これは、[%SOAP.WebServiceReturnFault](#) メソッドを呼び出します。このメソッドは、Web サービスの実行を即座に停止し、有効な SOAP メッセージをクライアントに返します。この SOAP エンベロープには、通常の日本文ではなく、SOAP フォルトが含まれています。

このメソッドを [SOAPTutorial.SOAPService](#) クラスに追加してリコンパイルします。

```
ClassMethod ApplicationError (code, detail)
{
    set fault=##class(%SOAP.Fault).%New()
    set fault.faultcode=code
    set fault.detail=detail
    set fault.faultstring="application error"
    // ReturnFault must be called to send the fault to the client.
    // ReturnFault will not return here.
    do ..ReturnFault(fault)
}
```

エラー処理ロジック

ApplicationError メソッドを使用すると、Web サービスがエラーを生成する可能性のあるすべての場所で SOAP フォルトを生成できます。例えば、クライアントの入力に一致する ID 値を持つ **Contact** が存在しない場合、現在の **SOAPTutorial.SOAPService GetContact** メソッドは""を返して正常に終了します。このような場合、メソッドがエラーを適切に生成する必要があります。

SOAPTutorial.SOAPService GetContact を変更して、SOAP フォルトを生成するようにします。新しいメソッドは以下のとおりです。faultcode と detail の適切な値が

ApplicationError に渡されることに注意してください。

```
ClassMethod GetContact(id As %String) As SOAPTutorial.Contact [ WebMethod ]
{
  if ##class(SOAPTutorial.Contact).%ExistsId(id) {
    quit ##class(SOAPTutorial.Contact).%OpenId(id)
  }
  else {
    do ..ApplicationError("ContactNotFound",
                        "Contact with id "_id_" could not be found!")
  }
}
```

SOAP フォルト生成のテスト

Cachéターミナルを使用すると、**SOAPTutorial.SOAPService**に追加した SOAP フォルト生成コードをテストできます。 **GetContact** を呼び出して、無効な ID 値を渡すだけです。

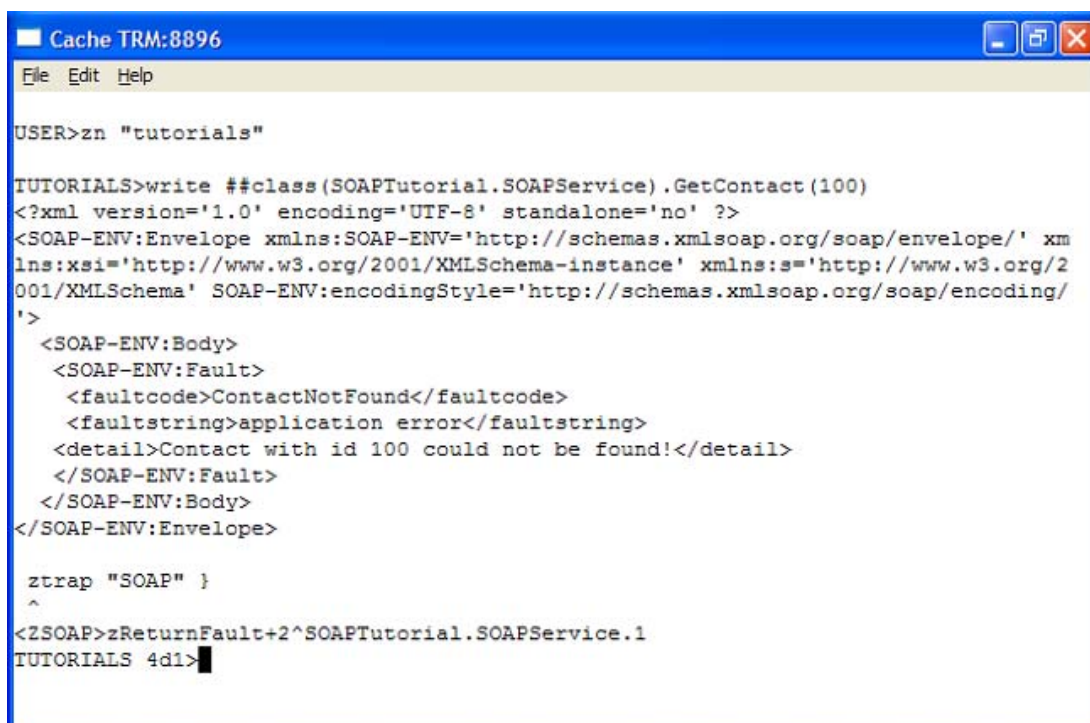
1. Cachéターミナルを開きます。 **ZN** コマンドを使用して、TUTORIALS ネームスペース(またはサンプル・アプリケーションをインストールしたネームスペース)に切り替えます。

```
USER> ZN "Tutorials"
TUTORIALS>
```

2. **GetContact** を呼び出して、無効な ID 値を渡します。

```
TUTORIALS> Write ##class(SOAPTutorial.SOAPService).GetContact(100)
```

3. メソッドによって SOAP フォルトが返されます。 以下は、その結果を示した画面です。



```
Cache TRM:8896
File Edit Help

USER>zn "tutorials"

TUTORIALS>write ##class(SOAPTutorial.SOAPService).GetContact(100)
<?xml version='1.0' encoding='UTF-8' standalone='no' ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/' xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' xmlns:s='http://www.w3.org/2001/XMLSchema' SOAP-ENV:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>ContactNotFound</faultcode>
      <faultstring>application error</faultstring>
      <detail>Contact with id 100 could not be found!</detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

ztrap "SOAP" }
^
<ZSOAP>zReturnFault+2^SOAPTutorial.SOAPService.1
TUTORIALS 4d1>
```

Web サービス・クライアントでの SOAP フォルトのトラップ

Web サービス・クライアント・コードは、Web サービス・プロデューサによって返された SOAP フォルトを処理できる必要があります。Caché Web サービス・クライアントでは、標準的な Caché エラー・トラップ方法を使用できます。

以下のメソッドを **SOAPService.SOAPServiceSoap** に追加します。このメソッドは Caché のエラー・トラップ機能を使用して、**SOAPTutorial.SOAPService GetContact** メソッドによって返された SOAP フォルトを処理します。

```
ClassMethod Demo(id As %Integer)
{
  set error = 0
  set $ZTRAP = "SOAPError"
  // call my web method
  set service = ##class(SOAPService.SOAPServiceSoap).%New()
  set person = service.GetContact(id)
  write person.Name
  quit

SOAPError
  set $ZTRAP = ""
  if $ZE["<ZSOAP>" {
    do $System.OBJ.DisplayError(%objlasterror)
  }
  quit 1
}
```

Note:

Cachéのエラー・トラップ機能についてさらに学習するには、Cachéドキュメント: [\[Caché開発ガイド\]-\[Caché ObjectScript の使用法\]-\[エラー処理\]](#) を参照してください。

SOAP フォルト・トラップ・コードのテスト

Caché ターミナルを使用すると、**SOAPService.SOAPServiceSoap Demo** メソッドでエラー・トラップ・コードをテストできます。

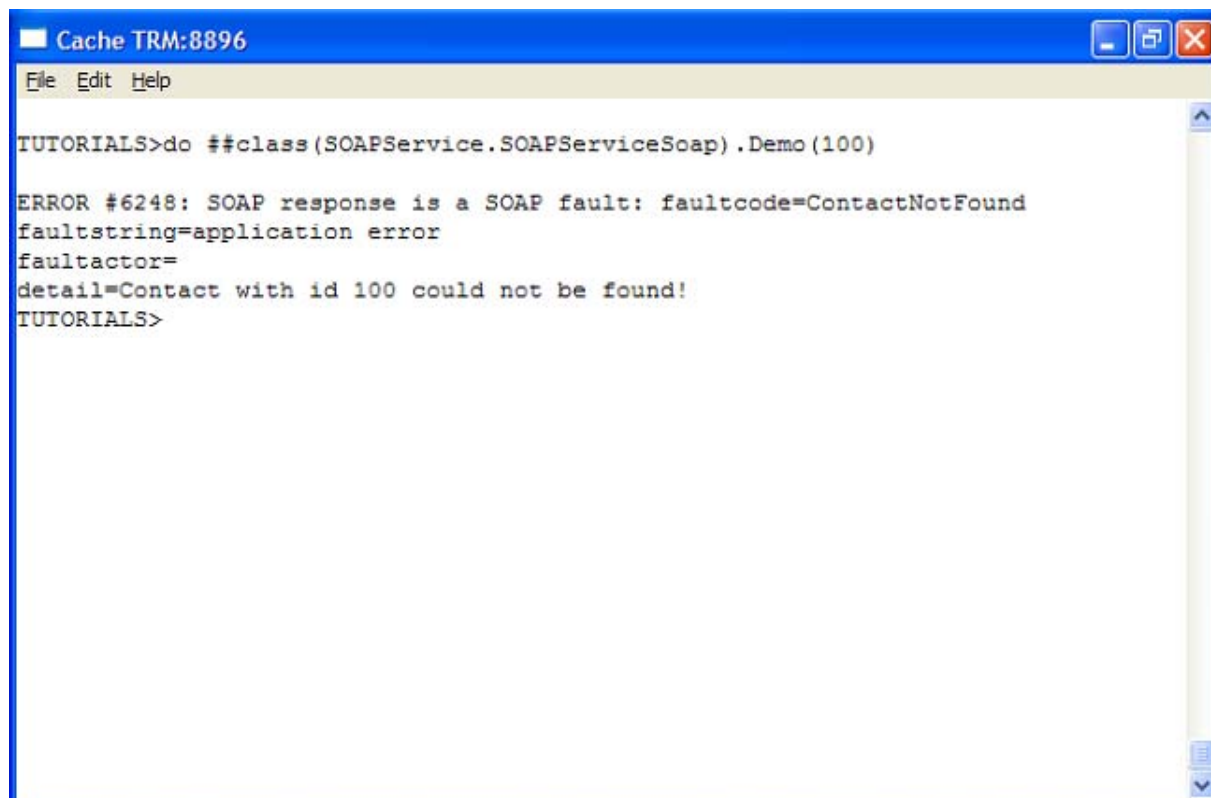
1. Cachéターミナルを開きます。 **ZN** コマンドを使用して、TUTORIALS ネームスペース(またはサンプル・アプリケーションをインストールしたネームスペース)に切り替えます。

```
USER> ZN "TUTORIALS"  
TUTORIALS>
```

2. **SOAPService.SOAPServiceSoap Demo** メソッドを呼び出して、無効な ID 値を渡します。

```
TUTORIALS> do ##class(SOAPService.SOAPServiceSoap).Demo(100)
```

以下は、**Demo** メソッドのエラー・トラップ・コードの結果を表示した画面です。



```
Cache TRM:8896  
File Edit Help  
TUTORIALS>do ##class(SOAPService.SOAPServiceSoap).Demo(100)  
ERROR #6248: SOAP response is a SOAP fault: faultcode=ContactNotFound  
faultstring=application error  
faultactor=  
detail=Contact with id 100 could not be found!  
TUTORIALS>
```

演習

演習 1: **SOAPTutorial.SOAPService** の **SaveContact** メソッドを変更して、**Contact** インスタンスを保存できない場合に SOAP フォルトが生成されるようにします。

演習 2 : **DemoSave** という名前のクラス・メソッドを **SOAPService.SOAPServiceSoap** クラスに追加します。このメソッドは以下の要件を満たしている必要があります。

- **SOAPTutorial.SOAPService SaveContact** メソッドを使用して、**Contact** インスタンスを保存します。この際、**SOAPService.SOAPServiceSoap SaveContact** メソッドをプロキシとして使用します。
- **SOAPTutorial.SOAPService SaveContact** メソッドによって生成されたすべての SOAP フォルトをトラップします。

Caché ターミナルを使用して、演習 1 と 2 で作成したエラーを生成するコードとエラー・トラップ・コードをテストします。そのためには、**SOAPService.SOAPServiceSoap DemoSave** メソッドを呼び出して、**Contact** インスタンスの代わりに空文字列を渡します。

要約

このチュートリアルの第Ⅲ章では、以下について学習しました。

- Cache Web サービス・プロデューサへの SOAP フォルト生成コードの追加
- SOAP フォルトを SOAP エンベロープに入れて返す Web サービス・プロデューサ・アプリケーションへのコードの追加
- Web サービス・プロデューサによって返される SOAP フォルトを処理する Cache Web サービス・コンシューマ・クラスへのコードの追加

例と演習ファイルのインストール

Caché インストールの `<cachsys>tutorials/webservices` ディレクトリには、チュートリアルの演習と例の完了に必要なファイルがすべて含まれています。これらのファイルは、新しい Caché ネームスペースにインストールできます。Caché ネームスペースの作成方法の詳細は、ページ下部のメモを参照してください。チュートリアルの 3 つの章の設定は、それぞれ多少異なります。設定手順については、以下の該当するセクションを参照してください。

第 I 章のインストールと設定

1. スタジオを開きます。**[ファイル]**→**[ネームスペース変更]** をクリックし、*TUTORIALS* などの、作業対象のネームスペースに接続します。
2. **[ツール]**→**[ローカルからインポート]** をクリックします。
3. Caché インストール・ディレクトリの `<cachsys>/dev/tutorials/webservices/` フォルダにある *Server.xml* をクリックします。**[開く]**→**[OK]** をクリックし、クラスをロードしてコンパイルします。
4. メモ : 演習と例を自分で行わない場合は、*Server.xml* の代わりに *ServerSolution.xml* をロードしてください。*ServerSolution.xml* には、作成済みの演習と例が含まれています。

第 II 章のインストールと設定

1. スタジオを開きます。**[ファイル]**→**[ネームスペース変更]** をクリックし、*TUTORIALS* などの、作業対象のネームスペースに接続します。
2. **[ツール]**→**[ローカルからインポート]** をクリックします。
3. Caché インストール・ディレクトリの `<cachsys>/dev/tutorials/webservices/` フォルダにある *ServerSolution.xml* をクリックします。**[開く]**→**[OK]** をクリックし、クラスをロードしてコンパイルします。
4. メモ : 演習と例を自分で行わない場合は、以下の追加手順を実行して、作成済みの演習と例をロードしてください。
 1. 上記の *ServerSolution.xml* のロード手順を使用して、*ClientSolution.xml* をロードします。このファイルには、作成済みの演習と例が含まれています。リモート Web サービスへのアクセスをより厳密にシミュレートするために、*ClientSolution.xml* と *ServerSolution.xml* をそれぞれ別のネームスペースにロードすることもできます。
 2. Caché スタジオ・クラス・エディタを使用して、**SOAPService.SOAPServiceSoap** の *LOCATION* パラメータ値を `http://localhost/csp/<ネームスペース名>/SOAPTutorial.SOAPService.cls` に変更します。`<ネームスペース名>` は、*ServerSolution.xml* をインストールしたネームスペースです。

第 III 章のインストールと設定

1. スタジオを開きます。**[ファイル]**→**[ネームスペース変更]** をクリックし、*TUTORIALS* などの、作業対象のネームスペースに接続します。
2. **[ツール]**→**[ローカルからインポート]** をクリックします。
3. Caché インストール・ディレクトリの `<cachsys>/dev/tutorials/webservices/` フォルダにある *ServerSolution.xml* をクリックします。**[開く]**→**[OK]** をクリックし、クラスをロードしてコンパイルします。
4. 上記の手順 1 ~ 3 を繰り返して、*ClientSolution.xml* をロードします。リモート Web サービスへのアクセスをより厳密にシミュレートするために、*ClientSolution.xml* と *ServerSolution.xml* をそれぞれ別のネームスペースにロードすることもできます。
5. Caché スタジオ・クラス・エディタを使用して、**SOAPService.SOAPServiceSoap** の *LOCATION* パラメータ値を `http://localhost/csp/<ネームスペース名>/SOAPTutorial.SOAPService.cls` に変更します。<ネームスペース名> は、*ServerSolution.xml* をインストールしたネームスペースです。
6. メモ： 演習と例を自分で行わない場合は、以下の追加手順を実行して、作成済みの演習と例をロードしてください。
 1. 上記の *ServerSolution.xml* のロード手順を使用して、*FaultsSolution1.xml* をロードします。このファイルには、作成済みの演習と例が含まれています。
 2. Caché スタジオ・クラス・エディタを使用して、**SOAPService.SOAPServiceSoap** の *LOCATION* パラメータ値を `http://localhost/csp/<ネームスペース名>/SOAPTutorial.SOAPService.cls` に変更します。<ネームスペース名> は、*FaultsSolution1.xml* をインストールしたネームスペースです。

Note:

Windows で Caché を標準インストールしている場合、<cachsys> は `C:¥Program Files¥Cache` になります。標準の Unix または Linux の環境では、cachsys は `/usr/cachsys` になります。

新しい Caché ネームスペースの作成方法を学習するには、Cachéドキュメント：[\[Cachéシステム管理\]](#)-[\[Caché システム管理ガイド\]](#)-[\[Caché の構成\]](#)-[\[ネームスペースの構成\]](#) を参照してください。

Appendix B:

サンプル・アプリケーションのコンパイルと生成

サンプル Caché アプリケーションをインストールしたら、クラスをコンパイルし、**Contact** と **PhoneNumber** にサンプル・データを入力します。

Contact と **PhoneNumber** をコンパイルするには、それぞれについて順番に以下の手順を実行します。

1. Caché スタジオでクラスを開きます。
2. Caché スタジオのメニュー・バーで **[ビルド]**→**[コンパイル]** をクリックします。

Contact と **PhoneNumber** のコンパイルが正常に完了したら、データを入力することができます。Caché ターミナルからデータ生成ユーティリティを実行します。

Contact と **PhoneNumber** について、順番に以下の手順を実行します。

1. Caché キューブからターミナルを起動します。
2. ターミナル・プロンプトで、**ZN** コマンドを使用して、クラスが格納されているネームスペースに切り替えます。例えば、*USER* でターミナルを開いており、クラスが *TUTORIALS* にある場合は、次のように入力します。

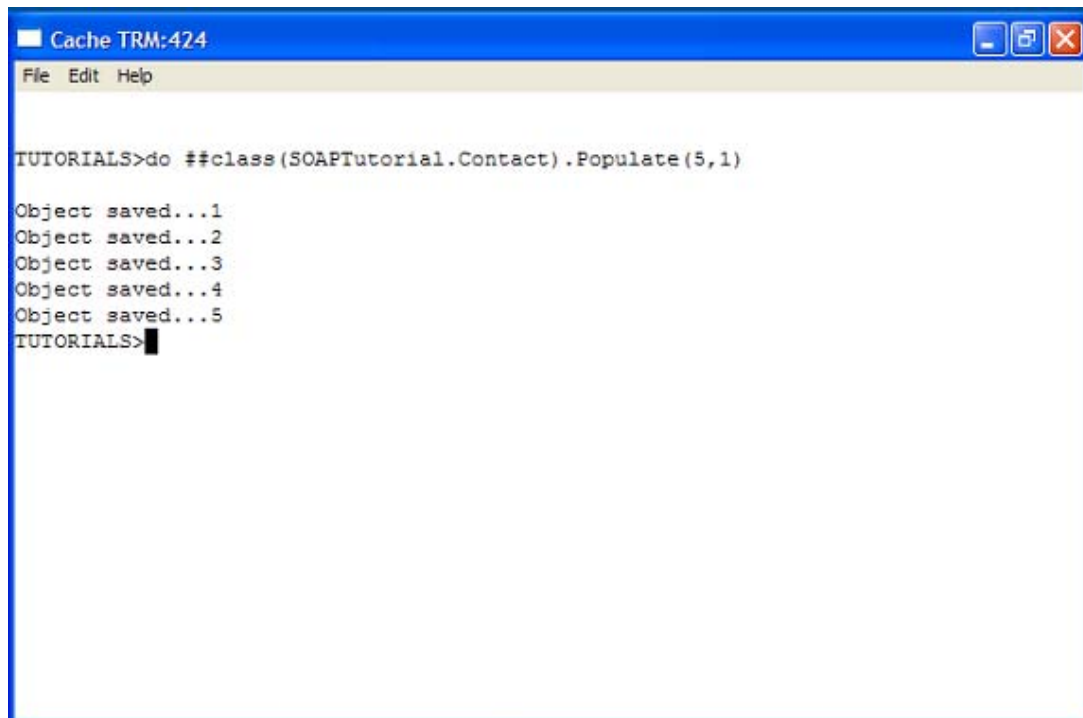
```
USER>zn "TUTORIALS"  
TUTORIALS>
```

3. 次に、データ生成ユーティリティを使用して **Contact** のインスタンスを作成します。5 つのインスタンスを作成するコマンドは、以下のとおりです。

```
TUTORIALS>do ##class(SOAPTutorial.Contact).Populate(5, 1)
```

このコマンドでは、*PackageName.ClassName* 構文を使用して、パッケージ名とクラス名を指定する必要があります。**Populate** の最初の引数（この場合は 5）は、作成するインスタンスの数を示します。**Populate** の 2 つ目の引数で 1 を渡すと、各インスタンスが作成されたときに、ユーティリティからフィードバックが提供されます。

次の画像は、*USER* ネームスペースに作成した、データ入力先となる **Contact** の 5 つのインスタンスを示しています。



```
Cache TRM:424
File Edit Help

TUTORIALS>do ##class(SOAPTutorial.Contact).Populate(5,1)

Object saved...1
Object saved...2
Object saved...3
Object saved...4
Object saved...5
TUTORIALS>
```

後で **Contact** または **PhoneNumber** のすべてのインスタンスを削除する場合は、**%KillExtent** を使用します。次のコマンドでは、*USER* から **ContactDB.Contact** のすべてのインスタンスが削除されます。

```
TUTORIALS>do ##class(SOAPTutorial.Contact).%KillExtent()
```

Note:

Contact のインスタンスを作成する前に、**PhoneNumber** のインスタンスを作成することはできません。これは、**Contact** を親とする親子リレーションシップが 2 つのクラス間に確立されているためです。各 **PhoneNumber** インスタンスは、**Contact** インスタンスに属している必要があります。