CHAPTER 10: WEB SERVICES

Objectives

The objectives are:

- Provide an overview on how Microsoft Dynamics® NAV supports Web services.
- Discuss historical integration options, Web services benefits and ideas.
- Create a codeunit Web service, expose a page and a codeunit Web service and consume them from an external application.

Introduction

Web services are a standardized way for independent software systems to communicate with one another over standard Internet protocols. Web services architecture is designed for dynamic program-to-program interaction.

Many kinds of distributed systems can be implemented in Web services architecture. Examples of distributed systems include synchronous and asynchronous messaging systems, distributed computational clusters, mobile-networked systems, grid systems, and peer-to-peer environments.

The broad spectrum of requirements for program-to-program interactions means that the protocols that are used by Web services are more flexible than Web protocols. However, just as the World Wide Web, Web services also rely on a few specific protocols, such as SOAP.

Microsoft Dynamics® NAV 2009 supports Web services, which makes it easy to integrate Microsoft Dynamics NAV with other systems. This is possible with the introduction of Microsoft Dynamics NAV Server.
Web Services Overview

Web services are designed to facilitate the highly dynamic data interchange that is required in business transactions. Web services, as standardized integration technologies, bring value to businesses by breaking down data silos that are created by proprietary integration options. These proprietary integration technologies make it difficult to obtain information in and out of the different systems.

Web services are described by using machine-readable metadata, which provide a robust development and operational environment. Web service metadata serves several purposes. The metadata is used to describe the message interchange formats that a Web service supports, and the valid message exchange patterns of a service. Metadata is also used to describe the capabilities and requirements of a service.

Web Services Description Language (WSDL), which is an XML-based language for defining Web services, is used to express the interchange formats and message exchange patterns of the Web services.

Most major software development environments, such as Microsoft® Visual Studio® 2005, can be used to build applications that use Web services. Moreover, because Web services are XML based, Web services can be built across platforms and programming languages.

Web Services Architecture

The Web service integration with Microsoft Dynamics NAV is facilitated through Web service enabled codeunits and pages. External systems read and write data on pages and call codeunits as defined by the common Web service protocols, with correct authentication and authorization. The Web service capabilities in Microsoft Dynamics NAV help customers reap the benefits of a service-oriented architecture (SOA).
Microsoft Dynamics NAV Web services are useful to customers and partners who want to use business logic or use a standard interface to access data from outside Microsoft Dynamics NAV.

FIGURE 10.1 MICROSOFT DYNAMICS NAV 2009 THREE-TIER ARCHITECTURE

Microsoft Dynamics NAV Web Services Support

There are several simple types of Web services that can be published by Microsoft Dynamics NAV. The types of Web services correspond to the degree of complexity required in the Web service interface. All Web services can run C/AL code and validation triggers.

The simplest Web service type is developed by using the page object. By using the page object, Microsoft Dynamics NAV constructs a default Web service that has a fixed set of methods. Generally, it corresponds to general data access, such as get and set individual values or retrieve and update lists of values.

Another Web service type involves including codeunits and the functions from those codeunits in the Web service.

The last type of Web service includes the ability to pass complex data types by using an XMLport object as a parameter in a codeunit function.

How Microsoft Dynamics NAV integrates Web services saves the complexity of manually setting up Web service frameworks, such as managing the WSDL descriptions. To publish any type of Microsoft Dynamics NAV Web service (page or codeunit), add it to the Web Service table.
Pages and codeunits that are added to the Web Service table in Microsoft Dynamics NAV are immediately published and available for Web service requests over the network.

Consumers of these Web services, which are systems integrating with Microsoft Dynamics NAV, only have to know the network name (or address) of the computer that is running Microsoft Dynamics NAV Server and the names given to the individual pages and codeunits.

For example, with the following configurations:

- Computer name that runs Microsoft Dynamics NAV Service: NAV_Server1
- Web service port: 85
- Microsoft Dynamics NAV Server instance: DynamicsNAV
- Company Name: CRONUS International Ltd.
- The published page Web service: MyCustomer

Then the MyCustomer Web service is available at the following URL:

http://Nav_Server1:85/DynamicsNAV/WS/CRONUS_International_Ltd/Page/MyCustomer

Microsoft Dynamics NAV manages Web service requests exactly like it handles requests from end-users. User rights authorization and validation, input data validation, business logic invocation, and concurrency control are all managed in the same manner as requests from a Microsoft Dynamics NAV client.

This guarantees that the integrity of the Microsoft Dynamics NAV data is not compromised by using Web services. It also means that code that validates data or invokes business logic for systems that use the Web services that are provided by Microsoft Dynamics NAV does not have to be replicated.

When writing code for Web services, avoid using any end-user confirmation dialog boxes or message boxes. The code for Web services cannot interact with the client that called the code. It cannot respond to a dialog box or any other client interaction requests. Running code with client interaction causes an exception to be thrown. The exception can be caught and handled, but the Web service task will not be completed.

This operating system function is used to check whether the C/AL code is allowed to show any information on the screen. When the code is run from Web services, GUIALLOWED always returns FALSE. This is the same with when the code is run in Microsoft Dynamics NAV Application Server.
Web Services Opportunities

Integration with other systems and communicating with different entities outside Microsoft Dynamics NAV, is always a challenge in any implementation scenarios.

In most cases, customers have different systems that are running together and not only Microsoft Dynamics NAV. Not only that, they may have to exchange data with entities outside their organization, which may use a different system or even a different platform and operating systems.

Historical Integration Option

Historically, Microsoft Dynamics NAV offers some integration possibilities by using the following technologies:

- C/Front
- C/ODBC
- OCX
- Automation
- Navision Application Server (NAS)

These integration options have their own pros and cons, and choosing the best technology, sometimes depends on a case by case basis.

Microsoft Dynamics NAV 2009 offers Web services support as a new method of interacting programmatically with the system. It can be accessed over a network and enables execution on a remote system. It stands as a widely-used, industry standard, which means it is used by other systems and products across the industry. Overall, Microsoft Dynamics NAV Web services support offers the best way to integrate across other systems.

Web Services Benefits

Web services support in Microsoft Dynamics NAV offers great value and benefits over its predecessors, these are as follows:

- Standard integration method, which takes advantage of developer skill sets. This in turn increases development speed and reduces the cost of implementation.
- Increase return on Microsoft Dynamics NAV investment with the integration of data and business logic. Codes have to be written one time and do not have to be redeveloped as the same codes are run whenever using Web services or using Microsoft Dynamics NAV Client.
What's New for Microsoft Dynamics® NAV 2009 Installation and Development

- Consistent integration point, which reduces management overhead, because of the same security, validation and business logic as using Web services and Microsoft Dynamics NAV Client.
- It also benefits from all other advantages of Web services as an industry standard.

Suitable Candidates for Web Services

Web services are suited for communication across platforms and programming language. With Microsoft Dynamics NAV 2009, it is easy to integrate with other systems and benefit from the flexibility that Web services offer.

The following are candidates for Web services:

- Solutions that have to execute business logic or read data from Microsoft Dynamics NAV.
- Solutions that write data to Microsoft Dynamics NAV and use the system to validate the data by using the existing business logic.
- Solutions that extend Microsoft Dynamics NAV with additional information such as customer information, exchange rates, or product information and have the extension that can be accessed from other systems.

The following are examples of several scenarios that are suited to Web services implementation:

- When a new order is created in the system, a Web service communicates with the shipping companies directly to make sure transportation of the order items, checking prices and availability.
- Integration with exchange rates, ensuring that the system is continuously updated with the latest exchange rates, which provides precise calculations of data.
- Integration with a Web service that retrieves the latest address information when new customers are added to the system to make sure accuracy in data.
- Integration with production planning, CRM, data warehouses, other ERP systems, specialized budgeting, planning or reporting systems, mobile solutions and many more.

Demonstration: Create, Expose and Consume Web Services

Microsoft Dynamics NAV supports page objects, and codeunits, and XML port objects that are passed as parameters in a codeunit function to be published as Web services.
An example of the usage of the Web services in Microsoft Dynamics NAV is publishing a Web service that lists all customers and has that Web service immediately available for authorized requests over the network. Web services allow for easy communication and data exchange in a secured environment.

The following demonstration shows how to:

- Create a simple codeunit.
- Register the codeunit Web service and a page Web service and publish them.
- Create console applications and consume these two Web services.

Create a Codeunit Web Service

The following steps show how to create a simple codeunit, with a function to capitalize lowercase input string.

In Microsoft Dynamics NAV Classic Client:

1. Click Tools > Object Designer. The Object Designer opens.
2. Click the Codeunit button to open the Codeunit list.
3. Click New. The C/AL Editor opens.
4. Click View > C/AL Globals. The C/AL Global window opens.
5. Click the Functions tab, and type the following:
   - Name: Capitalize

6. Click the Locals button. The C/AL Locals window opens.
7. In the Parameters tab, type the following:
   - Name: InputString
   - DataType: Text
   - Length: 250

8. Click the Return Value tab, and type the following:
   - Name: OutputString
   - ReturnType: Text
   - Length: 250

9. Close the C/AL Locals window and then close the C/AL Globals window.
10. In the Capitalize function, type the following:

    ```
    OutputString := UPPERCASE(InputString);
    ```

11. Save the codeunit by clicking File > Save As. The Save As dialog box opens.
12. Type 123456701 in the **ID** and MyCodeunit in the **Name** and then click **OK**. Now, the codeunit looks like the following:

![Codeunit 123456701 MyCodeunit - C/AL Editor](image)

FIGURE 10.2 MYCODEUNIT CODEUNIT

13. Close the C/AL Editor.

**Expose the Web Services**

After the codeunit is created and saved, register it in the Web Services form and select the Published check box. The following steps show how to register Web services:

In the Object Designer:

1. Click the **Form** button to open the Form list.
2. Select form 810, Web Services and then click **Run**.
   The Web Services form opens.
3. Type the following:

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Object ID</th>
<th>Service Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Codeunit</td>
<td>123456701</td>
<td>DemoCU</td>
</tr>
<tr>
<td>Page</td>
<td>21</td>
<td>Customer</td>
</tr>
</tbody>
</table>

The first line registers the codeunit Web service that is created in the previous steps. The second line registers a page Web service, in this case the Customer page.
4. Select the **Published** check box on both lines to publish the Web services. Now, the Web Services form looks like the following:

![Web Services Form](image)

**FIGURE 10.3 THE WEB SERVICES FORM**

When the Web service is marked as published in the Web Services form, it is immediately available for requests over the network.

5. Close the Web Services form.

**Checking the WSDL**

Before continuing, it is recommended to check that the Web services that previously published are available. The following steps show how to browse to the WSDL document to make sure that the Web services are available.

1. Start Internet Explorer.
2. In the Address bar, type the following:

```
http://localhost:85/DynamicsNAV/WS/CRONUS_International_Ltd/Services
```

**NOTE:** This address is based on the name of the computer that is running the Microsoft Dynamics NAV Service and the company that is used. The company name is case-sensitive. In this case, **localhost** is used instead of the computer name.
3. The page lists the Web services registered and published in the Expose The Web Service section.


Create a Console Application to Consume the Codeunit Web Service

The codeunit Web service is available. The following steps show how to create a simple console application in Visual Studio to consume the codeunit Web service. These steps start with creating a new project in Visual Studio and then adding a Web Reference for the Web services published from Microsoft Dynamics NAV.

2. Click File > New, and then click Project. The New Project window opens.
3. Expand the Visual C# node, and select Console Application.
4. Type the following:
   - Name: UseDemoCU
5. Click OK. The UseDemoCU project initiates with default files and references.

Add Codeunit Web Service as a Web Reference

The next step is to add the published Web service as a Web reference into the project.
In Visual Studio:

1. In the Solution Explorer, right-click the References node in the project, and then click Add Web Reference. The Add Web Reference window opens.
2. Type the following in the URL bar:

   http://localhost:85/DynamicsNAV/WS/CRONUS_International_Ltd/Services

   **NOTE:** This address is the same as the address that is used when checking the WSDL by using Internet Explorer.

3. Click GO. The DemoCU service is displayed together with other services that are available.

   ![Add Web Reference Window](image)

   **FIGURE 10.5 THE ADD WEB REFERENCE WINDOW**
4. Click **View Service** for the DemoCU service and then click **Add Reference**.

![Image of Add Web Reference dialog](image)

**FIGURE 10.6 THE DEMOCU WEB SERVICE DESCRIPTION**

**Type the Code**

The following steps show how to create a code to consume the codeunit Web service.

In Visual Studio:

1. On the **Program.cs** tab, type the following:

```csharp
using System;
using System.Collections.Generic;
using System.Text;
namespace UseDemoCU
{
    using localhost; // using the web reference
class Program
    {
        static void Main(string[] args)
        {
```

Microsoft Official Training Materials for Microsoft Dynamics ©
Your use of this content is subject to your current services agreement
// Create a new instance of the service.
DemoCU ws = new DemoCU();
// Use default credentials for authenticating
// against Microsoft Dynamics NAV.
ws.UseDefaultCredentials = true;
// Declare variables to work with.
string InputString = "";
string OutputString = "";
Console.WriteLine("Input lowercase sentence,
end by [ENTER] ");
InputString = Console.ReadLine();
// Call the Microsoft Dynamics NAV codeunit Web
service.
try
{
    OutputString = ws.Capitalize(InputString);
}
catch (Exception e)
{
    Console.WriteLine(e.Message.ToString());
}
// Write output to the screen.
Console.WriteLine("Result: {0}",
OutputString);
// Keep the console window up until you press
ENTER.
    Console.ReadLine();
}
2. Click **File > Save All** to save all the changes in the project.

![FIGURE 10.7 THE CODE IN VISUAL STUDIO](image)

**Consume the Codeunit Web Service**

The following steps show how to test the console application and consume the codeunit Web service:

In Visual Studio:

1. Press **F5** to run the application in debug mode. The console window opens.
2. Type the following:
   - this is a lowercase sentence
3. Press ENTER. Now, the console application looks like the following:

![FIGURE 10.8 THE CONSOLE APPLICATION RESULT](image)

The input string ‘this is a lowercase sentence’ is capitalized by using the codeunit Web service that is running on Microsoft Dynamics NAV Service. This results in the output string ‘THIS IS A LOWERCASE SENTENCE.’

4. Press ENTER to close the console window.

Create Console Application to Consume the Page Web Service

In the Expose the Web Service section, two Web services are exposed and published. The previous sections show how to create a console application to consume the first Web service, which is the codeunit Web service. The next section shows how to create a console application to consume the second Web service, which is the page Web service.

In Visual Studio:

1. Click File > New, and then click Project. The New Project window opens.
2. Expand the Visual C# node, and select Console Application.
3. Type the following:
   - Name: UseDemoPage
4. Click OK. The UseDemoPage project initiates with default files and references.
Add Page Web Service as a Web Reference

The next step is to add the published Web service as a Web reference into the project.

In Visual Studio:

1. Repeat steps 1 to 3 in the Add Codeunit Web Service as a Web Reference section.
2. Instead of using the DemoCU service, click View Service for the Customer service and then click Add Reference.

![Add Web Reference](image)

**FIGURE 10.9 THE CUSTOMER WEB SERVICE DESCRIPTION**

Type the Code

The following steps show how to create a code to consume the codeunit Web service.
In Visual Studio:

1. On the **Program.cs** tab, type the following:

```csharp
using System;
using System.Collections.Generic;
using System.Text;
namespace UseDemoPage
{
    // import newly generated Web service proxy
    using localhost;

    class Program
    {
        static void Main(string[] args)
        {
            // create instance of service and setting credentials
            Customer_Service service = new Customer_Service();
            service.UseDefaultCredentials = true;
            // create instance of customer
            Customer cust = new Customer();
            cust.Name = "Customer New";

            // insert customer
            service.Insert(ref cust);
            PrintCustomer(cust);
            Msg("The program inserted a new customer, press [ENTER] to continue");
            Console.ReadLine();

            // create filter for searching for customers
            List<Customer_Filter> filter = new List<Customer_Filter>();
            Customer_Filter nameFilter = new Customer_Filter();
            nameFilter.Field = Customer_Fields.Name;
            nameFilter.Criteria = "C*";
            filter.Add(nameFilter);
            Msg("List before modification");
            PrintCustomerList(service, filter);
            Console.ReadLine();

            // modify customer
            cust.Name = cust.Name + " and Modified";
            service.Modify(ref cust);
            PrintCustomer(cust);
            Msg("The program modified the new customer name, press [ENTER] to continue");
            Console.ReadLine();
        }
    }
}
```
```csharp
Msg("List after modification");
PrintCustomerList(service, filter);
Console.ReadLine();
//delete customer
service.Delete(cust.Key);
Msg("List after deletion");
PrintCustomerList(service, filter);
Console.ReadLine();
Msg("Press [ENTER] to exit program!");
Console.ReadLine();
}

static void PrintCustomerList(Customer_Service service, List<Customer_Filter> filter)
{
    Msg("Printing Customer List");
    //conduct the actual search
    Customer[] list = service.Find(filter.ToArray(), null, 100);
    foreach (Customer c in list)
    {
        PrintCustomer(c);
    }
    Msg("End of List, press [ENTER] to continue");
}

static void PrintCustomer(Customer c)
{
    Console.WriteLine("No: {0} Name: {1}", c.No, c.Name);
}

static void Msg(string msg)
{
    Console.WriteLine(msg);
}
```
2. Click **File > Save All** to save all the changes in the project.
Consume the Page Web Service

The following steps show how to test the console application and consume the page Web service:

In Visual Studio:

1. Press **F5** to run the application in debug mode. The console window opens.

   The application inserts a new customer named Customer New with the default Customer No.

   ![FIGURE 10.11 INSERT A NEW CUSTOMER](image)

2. Press **ENTER** to continue.
Chapter 10: Web Services

The application prints a list of customers, which name starts with the letter C.

![Image](file:///C/Users/josanto/Documents/Visual%20Studio%202005/Projects/UserDemoPage/UserDemoPage/bin/Debug/...)

**FIGURE 10.12 CUSTOMER LIST WHICH NAME STARTS WITH THE LETTER C**

3. Press **ENTER** to continue.

The application modifies the new customer name to Customer New and Modified.

![Image](file:///C/Users/josanto/Documents/Visual%20Studio%202005/Projects/UserDemoPage/UserDemoPage/bin/Debug/...)

**FIGURE 10.13 MODIFY THE NEW CUSTOMER NAME**

4. Press **ENTER** to continue.
The application prints a customer list, which name starts with the letter C, after modification.

![Customer List After Modification](image1)

**FIGURE 10.14 CUSTOMER LIST AFTER MODIFICATION**

5. Press ENTER to continue.

The application deletes the new customer and prints a customer list, which name starts with the letter C, after deletion.

![Customer List After Deletion](image2)

**FIGURE 10.15 CUSTOMER LIST AFTER DELETION**

6. Press ENTER to close the console window.
Summary

This chapter discussed:

- Microsoft Dynamics NAV 2009's new integration option by using Web services.
- Web services support in Microsoft Dynamics NAV 2009.
- The opportunities using Web services as an integration option compared to historical options, the benefits of Web services and the suitable candidates for Web services implementation.

The chapter also demonstrated how to use create a codeunit, expose and publish a page and a codeunit Web service, and create console applications to consume the published Web services.
Quick Interaction: Lessons Learned

Take a moment and write down three key points you have learned from this chapter:

1.

2.

3.