



capevo
business process evolution

www.capevo.com

Webservice for udviklere

Indholdsfortegnelse

INDHOLDSFORTEGNELSE.....	2
VERSIONSHISTORIK	3
INDLEDNING	4
PROGRAMMERING	4
TUTORIAL.....	5
NY WEBSERVICE I VISUAL STUDIO.....	5
EN SIMPEL METODE.....	5
XML-FORMAT.....	6
OPSÆTNING I XFORM.....	6
TILKNYT WEBSERVICE TIL FORMULARSIDE.....	8
AFPRØVNING	9
WEBSERVICE-TYPER.....	11
ONLOAD	11
INTERACTIVE.....	13
DATALIST.....	15
FIELDVALIDATION.....	16
ONSHOW	17
USERLOGON OG ADMINLOGON	18
PUSH	19
WORKFLOW.....	21
FEJLHÅNTERING	22

Versionshistorik

Version	Dato	Ændringer / Beskrivelse
4.2	16-07-2008	Versionshistorik tilføjet
4.3	22-02-2010	Opdateret i henhold til XForm 4.4.1.0

Indledning

Denne vejledning er henvendt udviklere af webservices og indeholder kode-eksempler og trin-for-trin anvisninger til opsætning af en webservice. Vejledningen er henvendt til udviklere der kender C#, men som ikke nødvendigvis har erfaring med udvikling af webservices. Der forventes i øvrigt et basalt kendskab til hvordan formularer opsættes i XForm.

Vejledningen er ment som supplement til den eksisterende webservice-manual, som indeholder følgende:

- Hvilken XML der forventes som svar fra en webservice.
- Hvilke [#nøgleord#], som kan benyttes i den XML der opsættes i XForm.
- Overfladiske beskrivelser af hver webservice-type.

Derfor gentages de oplysninger ikke i denne vejledning.

Programmering

Kode-eksemplerne er i C#, men samme resultat kan opnås og samme fremgangsmåde kan benyttes, med et andet .NET-sprog.

Benyttes et helt andet sprog, f.eks. Java, har vi ikke belæg for at garantere, at samme fremgangsmåde kan benyttes med et godt resultat. I de tilfælde bør du se direkte på den XML der sendes frem og tilbage mellem XForm og dine webservices.

Tutorial

Her er en tutorial som starter helt fra bunden og slutter med at vise en fungerende OnLoad-webservice på en formular. Selvom nogle af tingene virker trivielle, anbefales det at du starter her. Senere kommer mere specifikke oplysninger om hver webservice-type, hvor det forventes du kender det grundlæggende som er beskrevet i dette afsnit.

Ny webservice i Visual Studio

For at komme hurtigt i gang kan du vælge at starte et nyt projekt af typen "ASP.NET Web Service". Hvis du allerede har et ASP.NET-projekt som du ønsker at tilføje en webservice til, kan du højreklikke og vælge "Add Web Service". Effekten af dette er, at du har en .asmx-fil som basis for at skrive nogle webservice-metoder.

Hvis du åbner din nye .asmx.cs-fil ser du bl.a. følgende kode, som er en udkommenteret webservice-metode.

```
// [WebMethod]
// public string HelloWorld()
// {
//     return "Hello World";
// }
```

En simpel metode

Fjern udkommenteringen og rediger metoden så den i stedet modtager en streng som sættes sammen med den streng som returneres.

```
[WebMethod]
public string Hello(string greet)
{
    return "Hello "+greet;
}
```

Hvis metoden modtager "Asia", returnerer den "Hello Asia".

Besøg nu din .asmx i browseren, f.eks. ved at højreklikke på den og vælge "View in browser". I browseren åbnes en liste over metoder i din webservice. Hvis du har fulgt ovenstående anvisninger, vises kun én metode i listen, nemlig "Hello".

XML-format

Klik på metoden og du ser nu den XML som webservicen forventer at modtage og den XML som den returnerer.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Hello xmlns="http://tempuri.org/">
      <greet>string</greet>
    </Hello>
  </soap:Body>
</soap:Envelope>
```

XML som webservicen forventer at modtage

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <HelloResponse xmlns="http://tempuri.org/">
      <HelloResult>string</HelloResult>
    </HelloResponse>
  </soap:Body>
</soap:Envelope>
```

XML som webservicen returnerer

Bemærk: Hvis du benytter .NET 2.0 får du præsenteret XML i både Soap 1.1 og Soap 1.2. Ovenstående eksempler afspejler Soap 1.1 som er det format XForm er testet med.

Opsætning i XForm

For at opsætte din webservice i XForm, skal du allerførst logge på XForm som administrator. I øverste højre hjørne er der en knap med titlen "Admin". Her er der en række faneblade, derunder et faneblad med titlen "Webservices".

Oversigten viser de webservices der allerede er opsat. Der er ikke opsat nogle webservices som udgangspunkt, så derfor vil listen være tom, hvis du ikke tidligere har opsat en webservice.

Klik på knappen "Opret ny webservice" og gør følgende:

1. **Afkryds "Service er aktiv"-fluebenet.**
Hvis fluebenet ikke er afkrydset, bliver webservicen ikke udført, selvom den er tilknyttet en formular.
2. **Under rullemenuen "Service type", vælg "OnLoad"**
Webservice-typen OnLoad forudfylder indholdet af et felt på en formular-side når siden indlæses.
3. **Ved "Service navn" skriv "Hello Test" el. lign.**
Navnet benyttes så du som administrator kan kende forskel på de forskellige webservices, men navnet vises i nogle tilfælde også for blanketudfylder, f.eks. hvis webserveren er nede.
4. **Ved "EndPointUrl" skrives http-adressen til din .asmx**
F.eks. "http://www.site.dk/Webservice/Service.asmx" afhængigt af hvor du har valgt at placere din webservice.

Bemærk: Hvis du ønsker sikker, krypteret kommunikation med din webservice, kan du indtaste en EndPointUrl med https. Det kræver naturligvis at der er et gyldigt SSL-certifikat på det websted som din webservice er udgivet til.

5. **Ved "Action" skriver du SOAPAction i din webservice**
Du finder SOAPAction i header-delen af den XML som din webservice forventer at modtage. I vores eksempel skal du skrive:
http://tempuri.org/Hello

Bemærk: Du kan ændre namespace på dine webservice-metoder ved at indsætte følgende lige over din klasse-erklæring. Så slipper du for at der står "tempuri.org":

```
[ Webservice(Description="Mine webservices",  
Namespace="http://www.mitnamespace.dk") ]
```

6. **Maks antal kald i timen efterlades som 0**
0 betyder at der ikke er nogen begrænsning for hvor meget webservicen må benyttes.
7. **ErrorNode kan efterlades som den er**
XForm leder efter denne node i webservicens svar og betragter kaldet som fejlet, hvis noden indeholder en tekst. Anvendelse beskrives under afsnittet "Fejlhåndtering".

8. **Indtast din egen e-mail-adresse i "Send email ved fejl til"**
Du vil modtage en email med detaljer om eventuelle fejl og den XML der blev sendt og modtaget af XForm.
9. **Timeouts i sekunder kan du efterlade som de er**
Hvis du f.eks. ved at der er sket en fejl i din webservice, hvis den ikke har svaret efter 5 sekunder, kan du skrive 5 ved "Receive timeout" og så vil XForm stoppe med at vente på svar efter 5 sekunder.
10. **I "Envelope"-feltet indsættes det som skal sendes**
Du kopierer blot hele XML-delen fra det request som vises når du besøger din webservice-metode i browseren.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Hello xmlns="http://tempuri.org/">
      <greet>string</greet>
    </Hello>
  </soap:Body>
</soap:Envelope>
```

11. **Udskift *string* med et nøgleord som henviser til et felt**
Ved at skrive [#hilsen#] i stedet for *string*, indsættes indholdet af et felt med navnet "hilsen" fra den formular som webservicen senere tilknyttes.
12. **Tryk på "Gem"-knappen nederst til venstre**

Du er nu færdig med at opsætte webservicen. Du kan knytte den til så mange formularsider som du ønsker. Men hvis du f.eks. ønsker at den skal fungere sammen med en formular, som har et andet feltnavn end "hilsen", skal du opsætte den påny.

Tilknyt webservice til formularside

For at knytte formularen til en formularside er det nødvendigt, at du først opretter en formular i XForm til formålet. Eksemplet her er baseret på du opretter en formular som:

- Har to formularsider.
- Har et felt på første side med feltnavn: "hilsen".
- Har et felt på anden side med feltnavn: "resultat".

Når du har oprettet formularen, skal du klikke på ikonet for "Indstillinger" på formularen. Ikonet ser sådan ud:



Klik på fanebladet "Webservices". Planen er at tage imod indholdet af "hilsen"-feltet fra side 1 og vise resultat i "resultat"-feltet på side 2. Øverst vises en rullemenu, hvor formularen er valgt. Den webservice du har opsat er af typen "OnLoad" som skal knyttes til en specifik formularside. Vælg side 2 i rullemenuen, da det er her feltet som du vil udfylde er placeret.

Når du har valgt side 2 kan du, i panelet til venstre, se den webservice som du har opsat. Tryk på plusset og den flyttes over til højre, hvilket indikerer at webservicen er tilknyttet formularsiden.

For at mappe den værdi som din webservice returnerer til et felt på formularen, skal du klikke på XML-ikonet ved den webservice du netop har tilknyttet.



Under rullemenuen ved "Formular-felt" vælger du det formular-felt som skal udfyldes. Vælg feltet "resultat" på listen.

I tekstfeltet "Webservice feltnavn" skriver du navnet på den node som indeholder den værdi, formular-feltet skal udfyldes med.

Da din webservice-metode returnerer en enkelt unavngiven værdi (string), sammensættes nodens navn af metodens navn, "Hello", og teksten "Result".

```
<HelloResult>string</HelloResult>
```

Den del af webservice-svaret som indeholder resultatet

Som illustreret herover, er det nodenavn du skal indtaste "HelloResult". Klik herefter på "Tilføj" og bekræft at det du har valgt og indtastet, vises i panelet til højre.

Afprøvning

Klik tilbage til oversigten over formulartyper. Klik på det andet ikon, som hedder "Brugerbetjening", hvilket åbner et pop-up-vindue som giver dig mulighed for at afprøve din formular.



1. Indtast noget i dit "hilsen"-felt på første side. F.eks. "John".
2. Bladr til næste side med "Næste"-knappen.

Bekræft at dit "resultat"-felt på anden side indeholder teksten "Hello John".
Du har nu været gennem alle trin fra udvikling til opsætning og afprøvning af en webservice. Uanset hvilken webservice-type du skal opsætte, fungerer det i store træk på denne måde.

Webservice-typer

Forskellene på de forskellige webservice-typer er primært hvilket resultat XForm forventer at modtage fra din webservice. Det beskrives her ved hjælp af C#-klasser, så du ikke behøver at tænke i XML.

Det er vigtigt at pointere, at du frit kan opsætte hvad du ønsker din webservice skal modtage fra XForm. Du indsætter blot den tekst (gerne XML) du ønsker, i "Envelope"-feltet under opsætningen af en webservice. XForm udskifter de nøgleord du har indsat og sender derefter teksten ukritisk til din webservice.

Omvendt stiller XForm krav til hvilket svar din webservice giver til XForm. I følgende afsnit fokuseres på hvad din webservice returnerer, da det er det som XForm stiller krav til.

Endvidere beskrives de små forskelle der er ved opsætningen af de forskellige webservice-typer. Kun forskellene beskrives, så det forventes at du har været gennem ovenstående tutorial eller på anden måde har grundlæggende kendskab til hvordan en webservice opsættes i XForm.

OnLoad

Forudfylder feltværdier på en formularside. Hvis du blot skal udfylde en enkelt feltværdi kan du benytte en metode som blot returnerer en enkelt værdi, som her:

```
public string GetCustomer ()
{
    return "Microsoft";
}
```

Hvis du derimod skal returnere flere værdier, kan du gøre det på to måder, lidt afhængigt af hvor mange værdier du ønsker at returnere.

Returnering af objekt

Du kan skabe en klasse som indeholder de attributter du gerne vil returnere, f.eks. som følgende:

```
public class Customer
{
    public string name;
    public string address;
    public string email;

    public Customer()
    {
    }
}
```

Eksempel på anvendelse af klassen:

```
[WebMethod]
public Customer GetCustomer(int customerID)
{
    Customer customer = new Customer();
    customer.name = "Microsoft";
    customer.address = "One Microsoft Way";
    customer.email = "billg@microsoft.com";
    return customer;
}
```

Fordelen ved dette er, at det er meget klart hvad din webservice returnerer. Når du mapper mellem din webservice og XForm-formularen, skriver du blot attributternes navne som "Webservice feltnavn".

Webservice-feltnavne i ovenstående eksempel vil være "name", "address" og "email".

Returnering af XmlDocument

Hvis du ikke er tilfreds med at skulle definere hver retur-værdi deklarativt i en klasse, kan du i stedet vælge at returnere et XmlDocument. Dette anbefales hvis du skal returnere mange værdier eller de værdier du skal returnere varierer fra gang til gang, f.eks. fordi den samme webservice benyttes til flere formularer.

Du skal blot sørge for, at dit XmlDocument indeholder en node for hver returværdi, hvor teksten i din node er den værdi som XForm skal udfylde i feltet.

Ulempen ved denne metode er, at man ved at besøge .asmx-filen ikke kan se nøjagtig hvad den returnerer.

```
<GetCustomerResponse xmlns="http://tempuri.org/">
  <GetCustomerResult>xml</GetCustomerResult>
</GetCustomerResponse>
```

Udsnit af hvordan en webservice der returnerer et "XmlDocument" ser ud

Herunder vises et kodeeksempel hvor et XmlDocument udfyldes med returnværdier som vil være forståelige af XForm.

```
[WebMethod]
public XmlDocument GetCustomer(int customerID)
{
    XmlDocument doc = new XmlDocument();

    doc.AppendChild(doc.CreateElement("root"));

    AddNode(doc, "name", "Microsoft");
    AddNode(doc, "address", "One Microsoft Way");
    AddNode(doc, "email", "billg@microsoft.com");

    return doc;
}

private void AddNode(XmlDocument doc, string name, string text)
{
    XmlElement element = doc.CreateElement(name);
    element.InnerText = text;
    doc.FirstChild.AppendChild(element);
}
```

Ovenstående metode mappes til XForm på nøjagtig samme måde som ved returnering af objektet.

Interactive

Ligesom OnLoad, udfylder denne webservice værdier i felter på formularen. Forskellen er, at udfyldelsen sker med det samme. Altså ikke ved bladrning i formularen. Et muligt scenarie kan være, at brugeren indtaster sit kundenummer og når feltet forlades, udfyldes formularens felter med adresseoplysninger, baseret på det indtastede kundenummer.

Teknisk fungerer Interactive på samme måde som OnLoad, med undtagelse af følgende:

- **Trigger field:**
Under opsætning af en Interactive, er der en ekstra indstilling der skal tages stilling til: "Trigger field". Her skrives navnet på det felt som skal aktivere webservicen. I ovenstående kundenummer-eksempel kan det være feltet i formularen hedder "CustomerID". Derfor indtastes *CustomerID* i feltet.

- **Nøgleord**

I Envelope kan du ikke benytte nøgleord for felter som eksisterer på den side hvor du tilknytter din Interactive. En undtagelse er dit trigger field. Men i stedet for [#CustomerID#], skal du skrive [#TRIGGER_FIELD_VALUE#]

Under afprøvning af en Interactive kan du lægge mærke til om "Vent et øjeblik" vises kortvarigt i bundpanelet, når du har ændret i dit trigger field og forlader det, f.eks. ved at klikke udenfor feltet.

Valgmuligheder i en rullemenu

Som noget særligt for Interactive, er det muligt at definere valgmuligheder i en rullemenu (<select> i html). Scenariet kan være, at brugeren indtaster et kundenummer og derefter får vist en rullemenu med medarbejdere hos den specifikke kunde.

Oplysningerne hentes via en Interactive webservice.

Det gøres ved at returnere en node med navnet "Options" som indeholder Option-objekter, baseret på denne klasse:

```
public class Option
{
    public string Value;
    public string Text;

    public Option(string item)
    {
        Value = item;
        Text = item;
    }
}
```

Da noden som indeholder objekterne skal hedde "Options", er du nødt til at indpakke Option-objekterne i en klasse, som f.eks. kan se sådan ud (.NET 2.0):

```
public class OptionListContainer
{
    public List<Option> Options;
}
```

Eller som array:

```
public class OptionListContainer
{
    public Option[] Options;
}
```

Du kan også indkapsle din attribut i en property, sådan:

```
public class OptionListContainer
{
    private List<Option> optionList;

    public List<Option> Options
    {
        get
        {
            return optionList;
        }
    }
}
```

Her er eksempel på anvendelse af ovenstående objekter i en metode:

```
public OptionListContainer GetAssociates(int customerID)
{
    OptionListContainer associates = new OptionListContainer();

    associates.Options = new List<Option>();
    associates.Options.Add(new Option("Brad Abrams"));
    associates.Options.Add(new Option("Alex Barnett"));
    associates.Options.Add(new Option("Rob Caron"));

    return associates;
}
```

Det "Webservice feltnavn" som du skal mappe til i ovenstående tilfælde, er "GetAssociatesResult". Hvis du selv vil styre feltnavnet, skal du indkapsle din OptionListContainer i endnu et objekt og returnere det.

Datalist

Udfylder valgmuligheder i en rullemenu. Ligesom OnLoad, udføres denne webservice når en formularside indlæses. XForm forventer at en Datalist returnerer et array af Option-objekter (se ovenstående beskrivelse af Interactive)

Forskellen i forhold til Interactive er, at ved Datalist behøver man ikke at pakke Option-objekterne ind i en klasse. Man kan altså godt returnere objekterne direkte, sådan:

```
public Option[] GetAssociates(int customerID)
{
    Option[] options = new Option[3];

    options[0] = new Option("Brad Abrams");
    options[1] = new Option("Alex Barnett");
    options[2] = new Option("Rob Caron");

    return options;
}
```

Tip: Du kan benytte en System.Collections.Generic.List i .NET 2.0: new List<Option>()

FieldValidation

Validerer indholdet af udfyldte felter. Denne webservice udføres når brugeren forsøger at bladre væk fra en formularside. Hvis indholdet ikke er gyldigt, får brugere ikke lov at fortsætte til næste formularside.

FieldValidation kan fremhæve de felter som er fejlet og evt. udskifte felterne med rullemenuer, hvis det er relevant. Det gøres ved at returnere et ValidationResult-objekt fra din webservice.

```
public class ValidationResult
{
    public Input[] Inputs;

    public ValidationResult()
    {
        Inputs = new Input[0];
    }
}
```

Bemærk: Se afsnittet om "Fejlhåndtering" hvis du ønsker at vise en fejlbesked for brugeren. Det bør altid anvendes i forbindelse med en FieldValidation, så brugere kan blive informeret om hvad fejlen består i og hvad de kan gøre for at rette det.

Klassen består af et array af Input-objekter. Det array repræsenterer de felter som skal fremhæves for brugeren som fejlet. Hvis du ikke vil fremhæve nogle felter som fejlet, kan du efterlade listen tom.

Input-klassen ser sådan ud:

```
public class Input
{
    public string Name;
    public Option[] Options;

    public Input()
    {
        Options = new Option[0];
    }
}
```

Name er navnet på det felt som du gerne vil fremhæve.

Hvis du gerne vil udskifte feltet med en rullemenu med valgmuligheder, skal du indsætte valgmulighederne i et Option-array. Option-objektet er beskrevet tidligere i manualen under Interactive.

Sådan kan ValidationResult-klassen anvendes:

```
public ValidationResult IsGreetingValid(string greet)
{
    ValidationResult result = new ValidationResult();

    if (greet == "invalid")
    {
        result.Inputs = new Input[1];
        result.Inputs[0] = new Input();
        result.Inputs[0].Name = "hilsen";
    }

    return result;
}
```

Som udgangspunkt vil ovenstående fremhæve et felt med feltnavnet "hilsen". Du kan dog anvende mapping til at fremhæve et andet felt. I "Webservice feltnavn" udfylder du blot "hilsen" og vælger så det felt som skal fremhæves i stedet. Det muliggør anvendelse af den samme FieldValidation på formularer med forskellige feltnavne.

OnShow

Kan afgøre om en formularside skal vises eller ej. Hvis formularsiden ikke skal vises, bliver den sprunget over, og den næste formularside vises i stedet.

Det gøres meget nemt i C#-kode. Der skal ganske enkelt returneres en boolean som angiver om en formularside skal vises.

```
public bool ShowGreeting(string greet)
{
    return (greet == "Asia");
}
```

Ovenstående springer formularsiden over, hvis *greet* ikke er "Asia".

Det giver ikke mening at foretage mapping på en OnShow og det er derfor heller ikke muligt.

UserLogon og AdminLogon

UserLogon og AdminLogon virker på samme måde. UserLogon benyttes til at give adgang til en sagsmappe som har adgangsmetoden "Webservice". AdminLogon benyttes til at afgrænse hvilke sagsbehandlere der må foretage brugersimulering.

Begge webservices modtager [#LogonUsername#] og [#LogonPassword#] og returnerer en boolean som angiver om der skal gives adgang til brugeren.

Her vises en simpel webservice-metode som både kan repræsentere en UserLogon og en AdminLogon webservice:

```
public bool Logon(string username, string password)
{
    if (username == "billg" && password == "apple")
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Under opsættelse af AdminLogon, er der mulighed for at vælge om den opsatte webservice skal benyttes til at afgrænse sagsbehandlers adgang til at simulere brugere i sagsmappen.

Push

Udføres efter brugeren trykker på "Indsend" på sidste side og inden kvitteringsteksten vises. Det forventes af denne webservice, at den returnerer et PushResult-objekt, som ser sådan ud:

```
public class PushResult
{
    public string Status;
    public string KvitteringScreen;
    public string KvitteringMail;
    public string Note;

    public PushResult ()
    {
    }
}
```

Herunder er betydningen af hver af ovenstående attributter beskrevet:

- **Status** er en statuskode (et heltal) som opsættes i "Statuskoder"-fanebladet af en XForm-administrator. Det kan være metoden har analyseret formularens indhold og identificeret at den skal have en bestemt status.
- **KvitteringScreen** er den tekst som formularudfylderens får vist på skærmen umiddelbart efter indsendelsen.
- **KvitteringMail** er den tekst som sendes til formularudfylderens, hvis denne har indtastet sin e-mail-adresse og afkrydset at vedkommende ønsker at modtage en kvitterings-mail.
- **Note** er en tekst som skrives i den indsendte formulars "Intern note". Der er en intern note tilknyttet hver indsendt formular. Den kan redigeres af sagsbehandlere og kan vises i sagsmappens arkiv.

Alle ovenstående attributter er optionelle. En Push kan altså returnere en helt tom PushResult.

Her er et eksempel på anvendelse af PushResult, hvor den indsendte formular gives en anden status hvis *greet* er udfyldt:

```
public PushResult PushGreeting(string greet)
{
    PushResult result = new PushResult();
    result.Status = "1";
    if (greet != "")
    {
```

```

        result.Status = "2";
    }
    result.KvitteringScreen = "You greeted "+greet;
    return result;
}

```

Meget ofte benyttes Push til at modtage hele formularens indhold. Det kan gøres ved at indsætte [#FormXML#] i din Envelope når du opsætter Push-webservicen. Det giver følgende resultat i XML:

```

<field name="dwbserviceblanketid" type="system"><![CDATA[1895]]></field>
<field name="dwbserviceblanketnavn" type="system"><![CDATA[Test-formular]]></field>
<field name="page1_MitFelt" type="text"><![CDATA[]]></field>
<field name="page1_DitFelt" type="text"><![CDATA[tekst]]></field>
<field name="page1_Afkrydsning" type="check"><![CDATA[on]]></field>
<field name="page1_EnRulleMenu" type="select"><![CDATA[]]></field>
<field name="page1_EtRadioFelt" type="radio"><![CDATA[]]></field>

```

Bemærk: En afkrydset checkbox indeholder "on" og indeholder ellers bare "".

Desværre er ovenstående XML ikke særligt velegnet til en .NET webservice, da det ikke direkte kan modtages i nogen metode-parameter. Derfor har vi indført [#FormXMLBase64#] som kan anvendes på følgende måde:

```

public PushResult PushGreeting(string formXmlAsBase64)
{
    byte[] bytes = Convert.FromBase64String(formXmlAsBase64);
    string xml = Encoding.UTF8.GetString(bytes);
    xml = string.Format("<root>{0}</root>", xml);

    XmlDocument doc = new XmlDocument();
    doc.LoadXml(xml);

    return new PushResult();
}

```

Base64-strengen konverteres til XML, lægges ind i et root-element og indlæses i et XmlDocument, så det er nemt at tilgå programmatisk.

Under opsættelse af en Push er det muligt at angive med et flueben, om Push skal ske ved indsendelse. Hvis afkrydsningen i dette flueben fjernes, bliver Push ikke udført når formularen indsendes. En sagsbehandler skal logge på XForm, åbne den indsendte formular og klikke på "Overfør til fagsystem" før denne Push udføres.

Workflow

Denne webservice-type kan udføres i forbindelse med evaluering af en regel i et Workflow. XForm forventer blot af den, at den returnerer en boolean som angiver om kaldet gik godt eller ej.

Da en Workflow-webservice ikke udføres undervejs i udfyldelsen af en formular som de fleste andre webservices, er det ikke alle nøgleord der kan benyttes. Nøgleord som ikke kan anvendes er bl.a. dem som handler om betaling, spørgerunder og digital signatur. Ønsker man nogle af disse oplysninger, skal man have modtaget dem tidligere, f.eks. ved hjælp af Push-webservice.

Her er et eksempel på en Workflow-metode som bearbejder noget formular-indhold og returnerer hvor vidt det gik godt:

```
public bool WorkflowWebservice(string formXmlAsBase64)
{
    try
    {
        ProcessXml(formXmlAsBase64);
        return true;
    }
    catch
    {
        return false;
    }
}
```

Metoden returnerer true med mindre ProcessXml-metoden smider en Exception.

Fra og med XForm 4.4.1.0 kan man i et workflow få kaldt webservices ved aktørindsendelse af en formular (dvs. redigering af en allerede indsendt formular). Selvom disse webservices skal oprettes som typen workflow, må de gerne returnere et PushResult-objekt. Dog er det kun feltet KvitteringScreen, der kan benyttes - De andre felter i objektet bliver ignoreret.

Fejlhåndtering

XForm har et simpelt system til fejlhåndtering. Det implementeres nemt i .NET ved at tilknytte en SoapHeader til det resultat som webservicen returnerer. Formålet med denne SoapHeader er at returnere en tekst eller en url hvis der opstår fejl i vores webservice. Derfor har vi kaldt den ErrorHandler. Det er en klasse som ser sådan ud:

```
public class ErrorHandler : SoapHeader
{
    public string ErrorMessage;
    public string ErrorUrl;
}
```

Bemærk: Klassen implementerer SoapHeader som ligger i dette namespace: System.Web.Services.Protocols;

For at kunne benytte din ErrorHandler skal du instantiere den øverst i din webservice-klasse, sådan:

```
public ErrorHandler errorHandler = new ErrorHandler();
```

For de metoder hvor du vil kunne returnere en ErrorHandler, skal du tilføje en attribut lige over metoden, sådan:

```
[WebMethod]
[SoapHeader("errorHeader", Direction=SoapHeaderDirection.Out)]
public string Hello(string greet)
{
    return "Hello "+greet;
}
```

Fra nu af vil din Hello-metode altid returnere en SoapHeader med to noder: ErrorMessage og ErrorUrl. Så længe du ikke udfylder de to værdier, reagerer XForm ikke på dem.

Hvis **ErrorMessage** er udfyldt, bliver indeholdet vist for formular-udfylder i en Javascript-alert.

Hvis **ErrorUrl** er udfyldt med en webadresse, bliver brugeren i stedet flyttet til den angivne adresse. Anvendelse af ErrorUrl er generelt til lidt mere alvorlige fejl, da brugeren føres væk fra den formular vedkommende er ved at udfylde.

Her er et eksempel på en anvendelse af ErrorHandler.

```
[WebMethod]
```

```
[SoapHeader("errorHeader", Direction=SoapHeaderDirection.Out)]
public string Hello(string greet)
{
    try
    {
        if (greet == "")
        {
            throw new Exception("An error occurred");
        }
        else if (greet == "0")
        {
            throw new ArgumentException("Incorrect greeting");
        }

        return "Hello "+greet;
    }
    catch (ArgumentException)
    {
        errorHeader.ErrorMessage = "Du gav en forkert hilsen";
    }
    catch (Exception)
    {
        errorHeader.ErrorUrl = "http://www.google.com";
    }

    return "";
}
```

Hvis *greet* ikke indeholder noget, sendes brugeren hen til Googles forside. Hvis *greet* indeholder '0', får brugeren at vide, at vedkommende har afgivet en forkert hilsen.

Det som webservicen returnerer, bliver ignoreret af XForm hvis enten ErrorMessage eller ErrorUrl er udfyldt.