PDF SHARE FORMS

Online, Offline, OnDemand

PDF forms and SharePoint are better together

Reading PDF forms values by using PDF Share Forms Web services

Contents

Dynamic (XFA) form details 2

Static (Acro) form details 2

Creating console application 4

- Required references and namespaces 4
- Getting service client 4
- Reading information from dynamic form 4
- Console result for dynamic form 5
- Reading information from static form 5
- Console result for static form 6

Whole console application code 7

This guide shows how to read data from PDF forms using PDF Share Forms Web services in console application. PDF Share Forms provides a web services that allow to retrieve and update PDF forms data and template. Each dynamic PDF form contains a PDF structure that acts as a container for XFA package. XFA format describes form template and data. It is XML based, so it is possible to work with it like with regular XML. For static PDF forms they provide simple dictionary interface to retrieve and update field values by name.

Dynamic (XFA) form details

”dynamicForm” PDF template is used in this example :

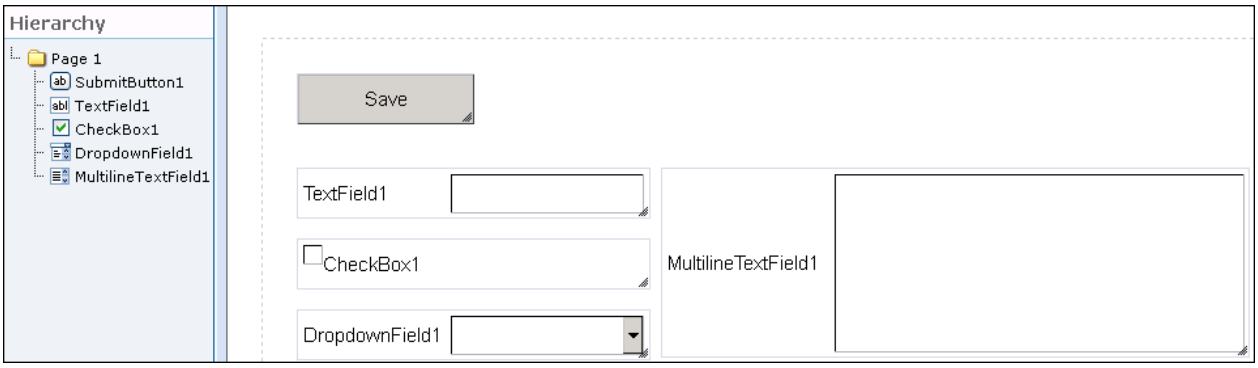

Figure 1. This dynamic template has text field, checkbox, dropdown and multiline text field

Target library (called Documents):

Following form is created (called dynamicForm_4):

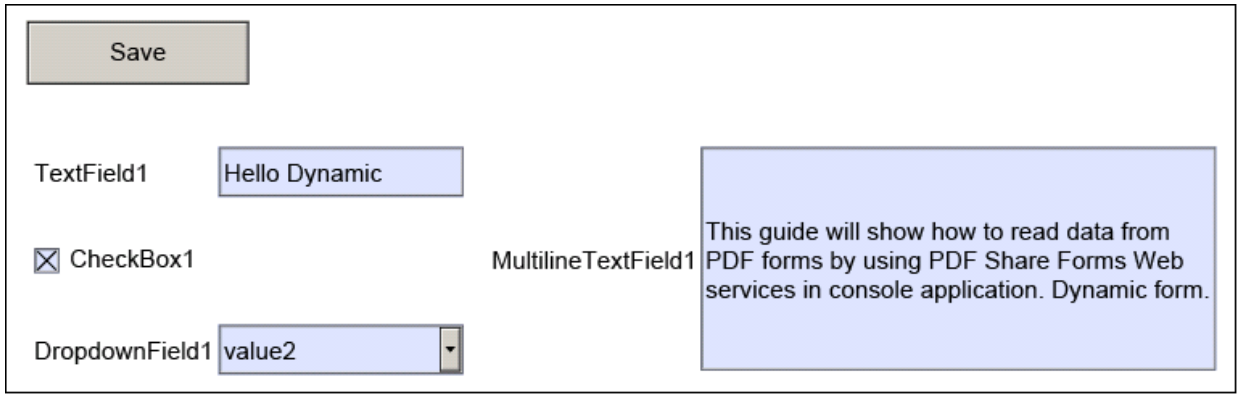

Figure 2. Filled in dynamic form

Each field is filled with information so it could be extracted later through console application.

Static (Acro) form details

“staticForm” PDF template is used in this example

780 Roosevelt, Irvine, CA 92620 • Tel +1 888 792 7871 • Fax +1 949 861 6447 • info@pdfshareforms.com • www.pdfshareforms.com

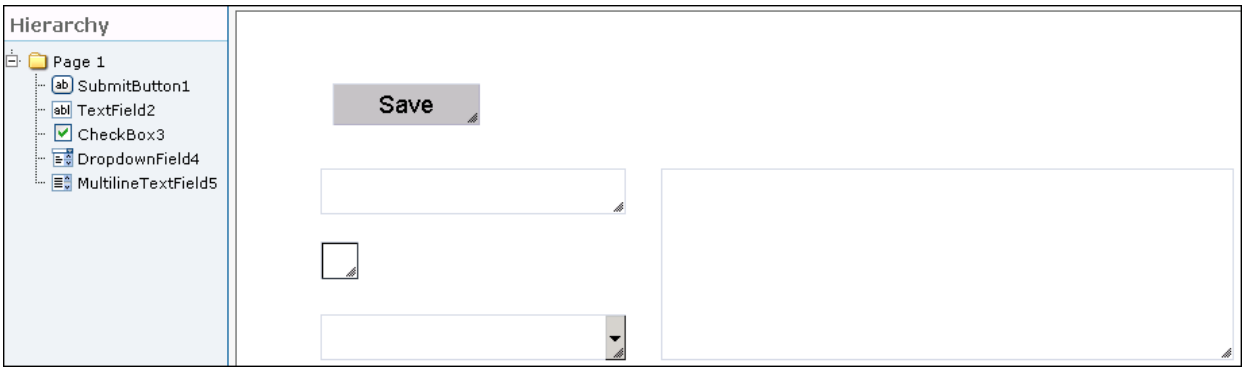

Figure 3. Static template with same controls

This template has text field, checkbox, dropdown and multiline text field.

Target library (called Documents):

Following form is created (called staticForm_6):

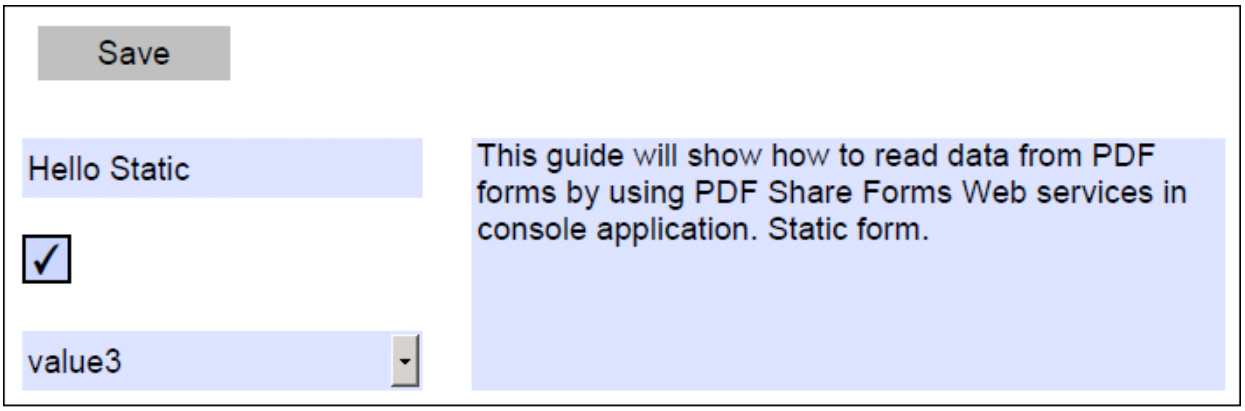

Figure 4. Filled in static form

Each field is filled with information so it could be extracted later through console application.

780 Roosevelt, Irvine, CA 92620 • Tel +1 888 792 7871 • Fax +1 949 861 6447 • info@pdfshareforms.com • www.pdfshareforms.com

Creating console application

Required references and namespaces

Namespaces that are used in this guide:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.ServiceModel;
using ConsoleApplication2.ServiceReference1;
using System.Xml.Linq;
using Microsoft.SharePoint;
```

For PDF Share Forms Web services it is required to create a service reference:

http://yourSiteName/_vti_bin/PDFForms/PDFForms.svc/MEX - this is a web service url. Proper site name should be used instead of “yourSiteName”. When it is used as reference, “/MEX” should be placed at the end, but in the code this url is used without “/MEX”.

Getting service client

This code prepares service client for work:

```
BasicHttpBinding binding = new BasicHttpBinding();
binding.ReaderQuotas.MaxArrayLength = int.MaxValue;
binding.ReaderQuotas.MaxStringContentLength = int.MaxValue;
binding.ReaderQuotas.MaxDepth = 128;
binding.MaxReceivedMessageSize = int.MaxValue;
binding.Security.Mode = BasicHttpSecurityMode.TransportCredentialOnly;
binding.Security.Transport.ClientCredentialType = HttpClientCredentialType.Ntlm;
EndpointAddress endpoint = new EndpointAddress("http://yourSiteName/_vti_bin/PDFForms/PDFForms.svc
");
PDFFormsExternalServiceClient client = new PDFFormsExternalServiceClient(binding, endpoint);
client.ClientCredentials.Windows.AllowedImpersonationLevel = System.Security.Principal.TokenImpers
onationLevel.Impersonation;
```

Note: Usually, it is necessary to increase quotas because XFA document size is larger than default quota would allow.

Reading information from dynamic form

Following code reads information from selected dynamic PDF form from SharePoint library and displays gathered information in the console in the following way: “Field name : value”.

```
string xmlString = client.GetXFAData("http://yourSiteName/", "Documents", 4);
XDocument document = XDocument.Parse(xmlString);
Console.WriteLine("Dynamic form:");
foreach (XElement node in document.Root.Elements())
{
 Console.WriteLine(node.Name + " : " + node.Value);
}
Console.Read();
```


780 Roosevelt, Irvine, CA 92620 • Tel +1 888 792 7871 • Fax +1 949 861 6447 • info@pdfshareforms.com • www.pdfshareforms.com

First, it is required to extract xml from PDF form by executing “GetXFAData” using site url, library name and field ID as parameters and parse extracted xml string to XDocument. Then each node from Root.Elements is read to get each field and its value.

Console result for dynamic form

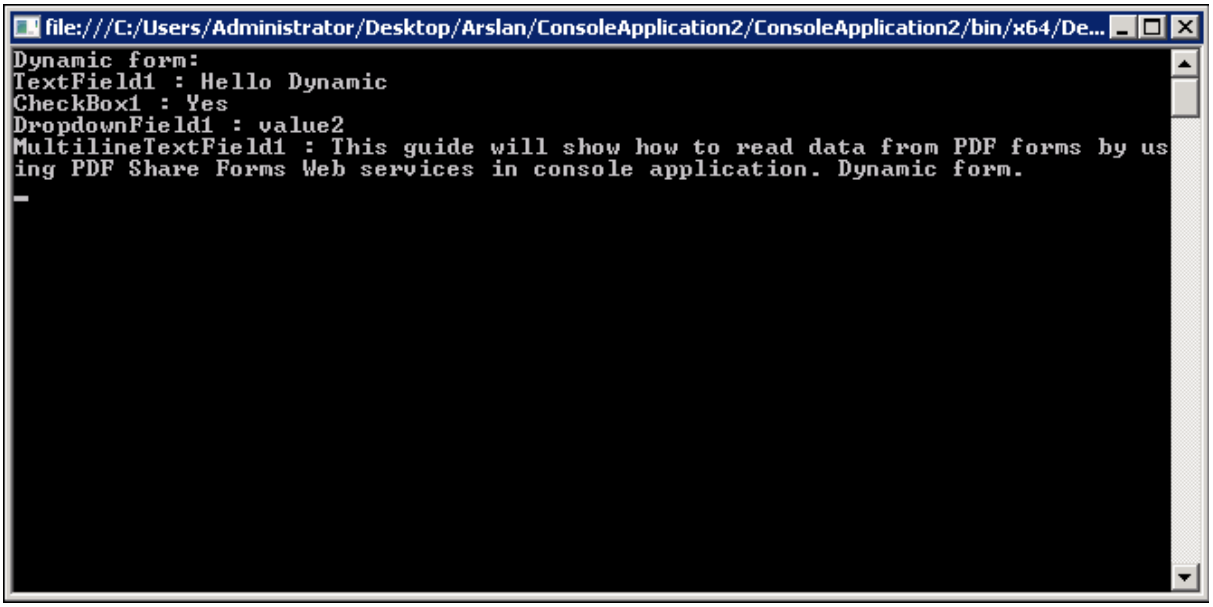

Figure5. Field data as it is seen in a console extracted from dynamic form

As it can be seen in Figure 5 – values and field names were extracted.

Reading information from static form

Following code reads information from selected static PDF form from SharePoint library and displays gathered information in the console in the following way: “Field name : value”.

```
using (SPSite oSite = new SPSite("http://yourSiteName/"))
{
 using (SPWeb oWeb = oSite.OpenWeb())
 {
 SPList list = oWeb.Lists["Documents"];
 Guid gd = list.ID;
 Dictionary<object, object> data = new Dictionary<object, object>();
 data = client.GetAcroFormData("http://yourSiteName/", gd, 6);
 Console.WriteLine("Static form:");
 foreach (KeyValuePair<object, object> pair in data)
 Console.WriteLine(pair.Key +" : "+ pair.Value);
 Console.Read();
 }
}
```

First, it is required to get list ID which is used to extract information from PDF form, and dictionary of objects to store extracted information. Then extract information from static PDF form by using “GetAcroFormData” with site url, library

780 Roosevelt, Irvine, CA 92620 • Tel +1 888 792 7871 • Fax +1 949 861 6447 • info@pdfshareforms.com • www.pdfshareforms.com

ID and PDF form ID as parameters and save extracted information into the dictionary. Once the data from PDF forms is extracted, actual field names and values can be read from each dictionary pair.

Console result for static form

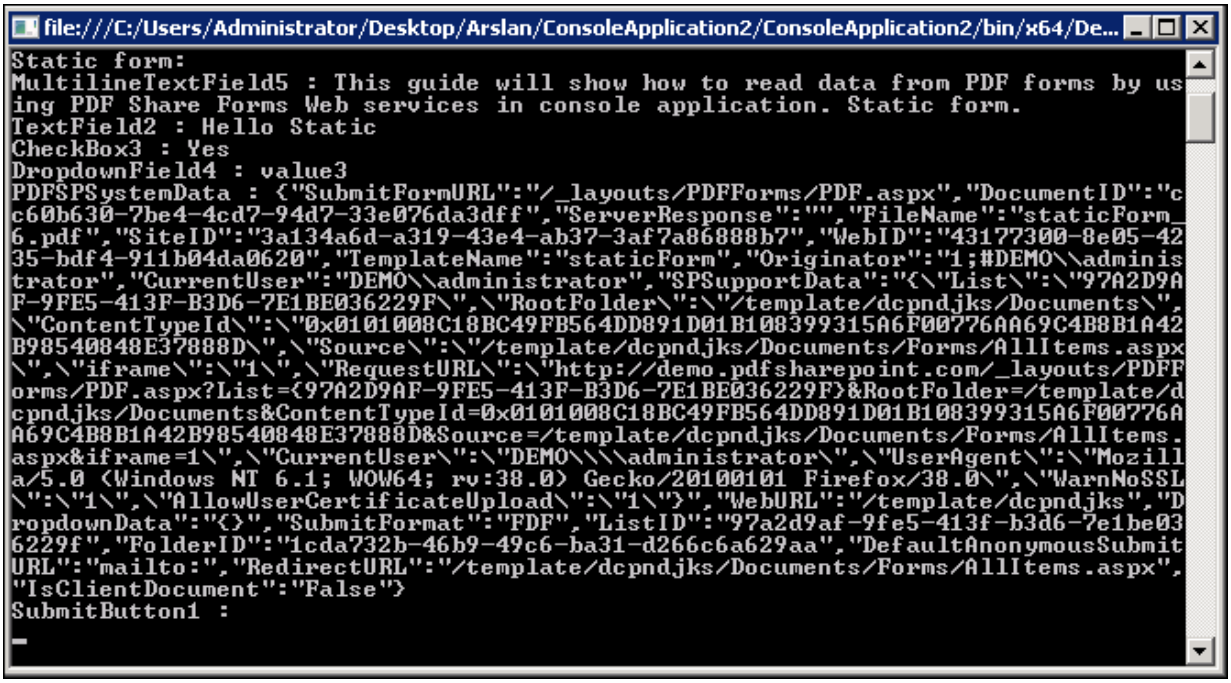

Figure6. Static form fields data as it is seen from a console

As it can be seen in Figure 6 - values and field names are extracted, including all controls from the static form. This explains why following controls are seen:

”SubmitButton1” – submit button which does not have any value;

“PDFSFSystemData” – system field which is used for storing information about the form itself.

780 Roosevelt, Irvine, CA 92620 • Tel +1 888 792 7871 • Fax +1 949 861 6447 • info@pdfshareforms.com • www.pdfshareforms.com

Whole console application code

```
using System;
using System.Collections.Generic;
using System.Text;
using System.ServiceModel;
using ConsoleApplication2.ServiceReference1;
using System.Xml.Linq;
using Microsoft.SharePoint;

namespace ConsoleApplication1
{
 class Program
 {
 static void Main(string[] args)
 {
 BasicHttpBinding binding = new BasicHttpBinding();
 binding.ReaderQuotas.MaxArrayLength = int.MaxValue;
 binding.ReaderQuotas.MaxStringContentLength = int.MaxValue;
 binding.ReaderQuotas.MaxDepth = 128;
 binding.MaxReceivedMessageSize = int.MaxValue;
 binding.Security.Mode = BasicHttpSecurityMode.TransportCredentialOnly;
 binding.Security.Transport.ClientCredentialType = HttpClientCredentialType.Ntlm;
 EndpointAddress endpoint = new EndpointAddress("http://yourSiteName/_vti_bin/PDFForms/PDFForms.svc");
 PDFFormsExternalServiceClient client = new PDFFormsExternalServiceClient(binding, endpoint);

client.ClientCredentials.Windows.AllowedImpersonationLevel = System.Security.Principal.TokenImpersonationLeve
l.Impersonation;

 string xmlString = client.GetXFAData("http://yourSiteName/", "Documents", 4);
 XDocument document = XDocument.Parse(xmlString);
 Console.WriteLine("Dynamic form:");
 foreach (XElement node in document.Root.Elements())
 {
 Console.WriteLine(node.Name + " : " + node.Value);
 }
 Console.Read();

 using (SPSite oSite = new SPSite("http://yourSiteName/"))
 {
 using (SPWeb oWeb = oSite.OpenWeb())
 {
 SPList list = oWeb.Lists["Documents"];
 Guid gd = list.ID;
 Dictionary<object, object> data = new Dictionary<object, object>();
 data = client.GetAcroFormData("http://yourSiteName/", gd, 6);
 Console.WriteLine("Static form:");
 foreach (KeyValuePair<object, object> pair in data)
 Console.WriteLine(pair.Key +" : "+ pair.Value);
 Console.Read();
 }
 }
 }
 }
}
```


780 Roosevelt, Irvine, CA 92620 • Tel +1 888 792 7871 • Fax +1 949 861 6447 • info@pdfshareforms.com • www.pdfshareforms.com