

State Machine Workflow mit InfoPath Formularen für SharePoint 2010 – Teil 8

Abstract:

Im achten und letzten Teil der Workflow Serie rollen wir den Workflow aus und testen ihn. Zusätzlich gibt es noch etliche Hinweise für die Fehlersuche.

[Teil 1: Einführung und grundlegende Funktionsweise des Workflow](#)

[Teil 2: Visual Studio Projekt anlegen und Test des minimalen Workflows](#)

[Teil 3: Zuweisungs und Start-Formular für den Workflow erstellen](#)

[Teil 4: Design des Workflows erweitern](#)

[Teil 5: InfoPath Formulare erstellen](#)

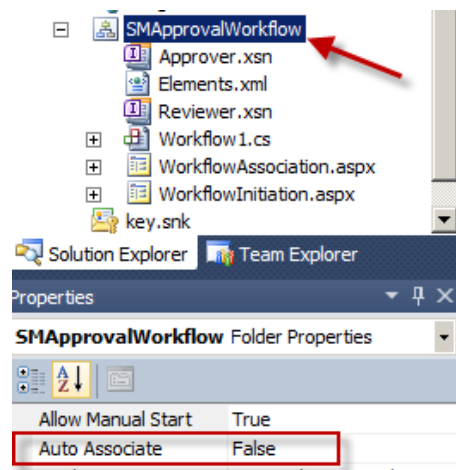
[Teil 6: InfoPath Formulare mit dem Workflow verbinden](#)

[Teil 7: Workflow mit Code erweitern](#)

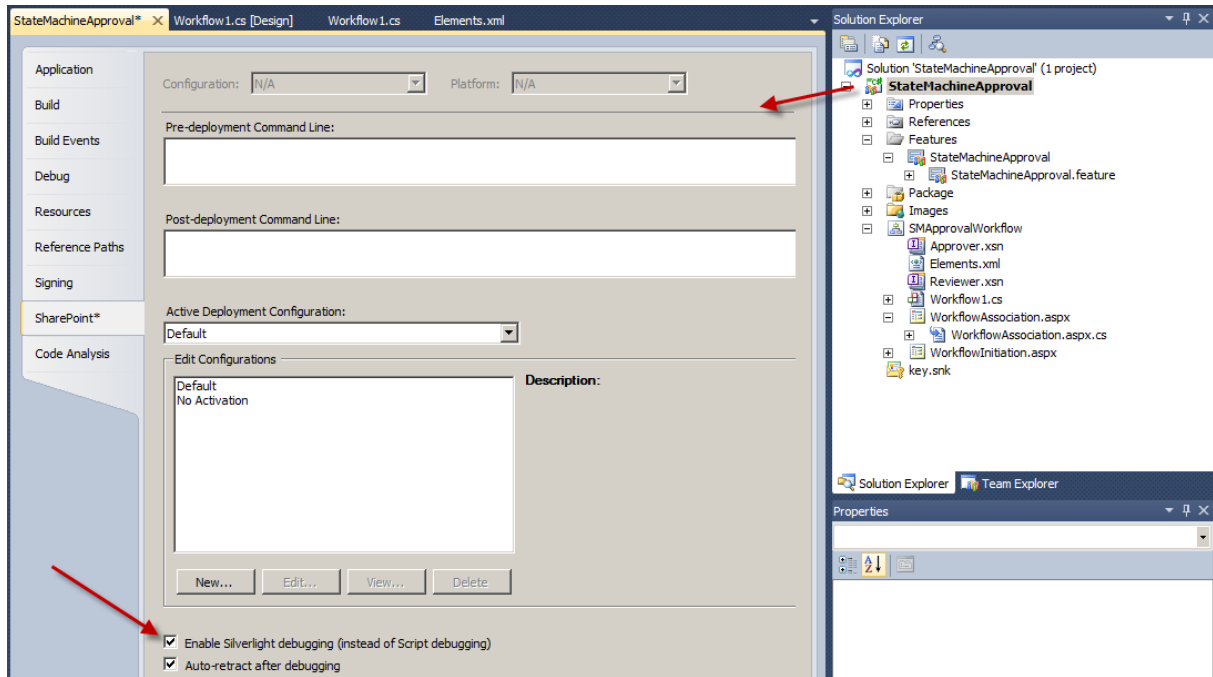
Teil 8: Workflow ausrollen, Testen und Fehlersuche

Nachdem wir im [letzten Teil](#) den Code fertig gestellt haben, können wir nun den Workflow ausrollen den testen. Zusätzlich möchte ich noch ein paar Hinweise bei Fehlern und deren Behebung geben.

Bevor wir den Workflow ausrollen und debuggen, nehmen wir noch eine Einstellung am Workflow Projekt in Visual Studio vor, damit dieser nicht automatisch einer Bibliothek zugewiesen wird. Denn genau dies haben wir ja schließlich implementiert und wollen dies auch testen. Die Einstellung kann man durch Klick auf den Workflow Ordner (SMAApprovalWorkflow) in der Eigenschaft Auto Associate ändern, indem man diese auf False stellte:



Zusätzlich schalte ich in den Projekteinstellungen das Debugging auf Silverlight Debugging um, damit Visual Studio die JavaScript Dateien von SharePoint nicht ständig nachlädt und wir dadurch Performance Einbußen beim Debugging hinnehmen müssten:



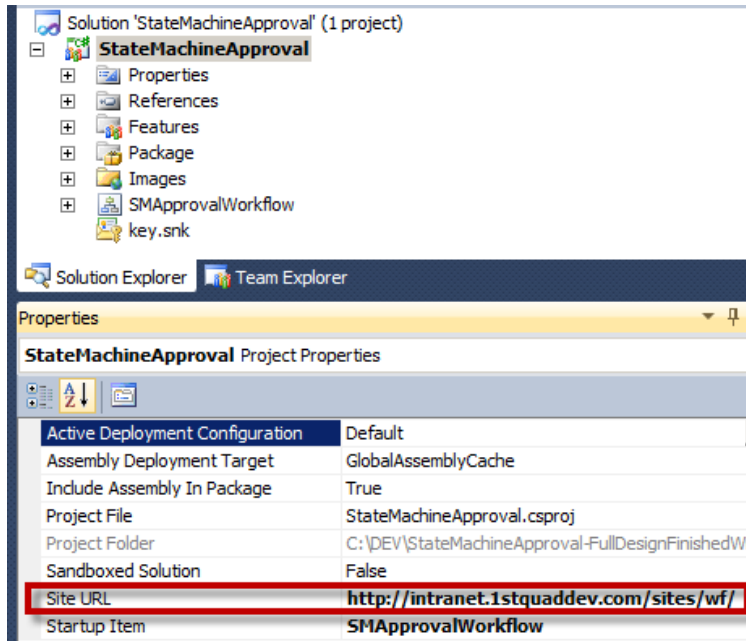
Damit sich der Workflow besser nachverfolgen lässt, kann man einen Breakpoint in der Methode `OnWorkflowActivated` setzen. Es ist in unserem ja nicht mehr notwendig, dass wir die Zuweisung und den Start des Workflow debuggen, da wir dies ja schon im [3. Teil](#) getestet haben:

```

private void OnWorkflowActivated(object sender, ExternalDataEventArgs e)
{
    try
    {
        XmlDocument retData = new XmlDocument();
        retData.LoadXml(workflowProperties.InitiationData);
        XmlNode approverNode = retData.SelectSingleNode("//approver");
        if (null != approverNode)
        {

```

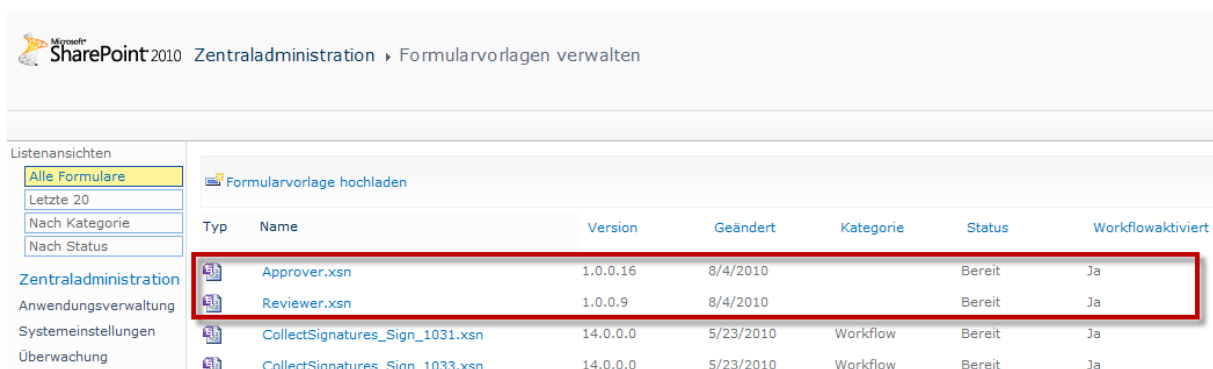
Wir können nun unseren Workflow durch Drücken der Taste F5 starten. Dadurch wird automatisch ein Browserfenster mit der Website geöffnet, die wir in den Projekteigenschaften angegeben haben:



Zusätzlich werden installiert unsere InfoPath Formulare im Hintergrund anhand des Feature Receivers installiert uns als Workflow Formulare gekennzeichnet. Man kann in der Zentraladministration nachschauen, ob dies auch tatsächlich geklappt hat:

Zentraladministration -> Allgemeine Anwendungseinstellungen -> Formularvorlagen verwalten

Dort sollten unsere InfoPath Formulare mit der Kennzeichnung als Workflow Formulare zu sehen sein:

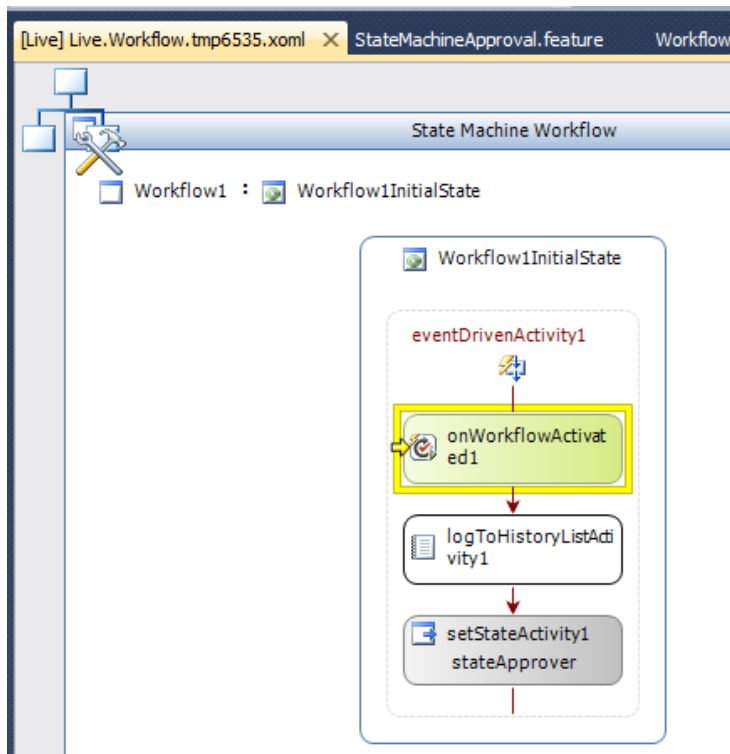


Als erstes muss nun der Workflow einer Dokumentbibliothek zugewiesen werden. Ich mache dies in meinem Fall bei den Freigegebenen Dokumenten meiner Testwebsite. Ich beschreibe die eigentliche Zuweisung und den Start des Workflows an dieser Stelle nicht mehr, da ich dies bereits in [Teil 3](#) gemacht habe.

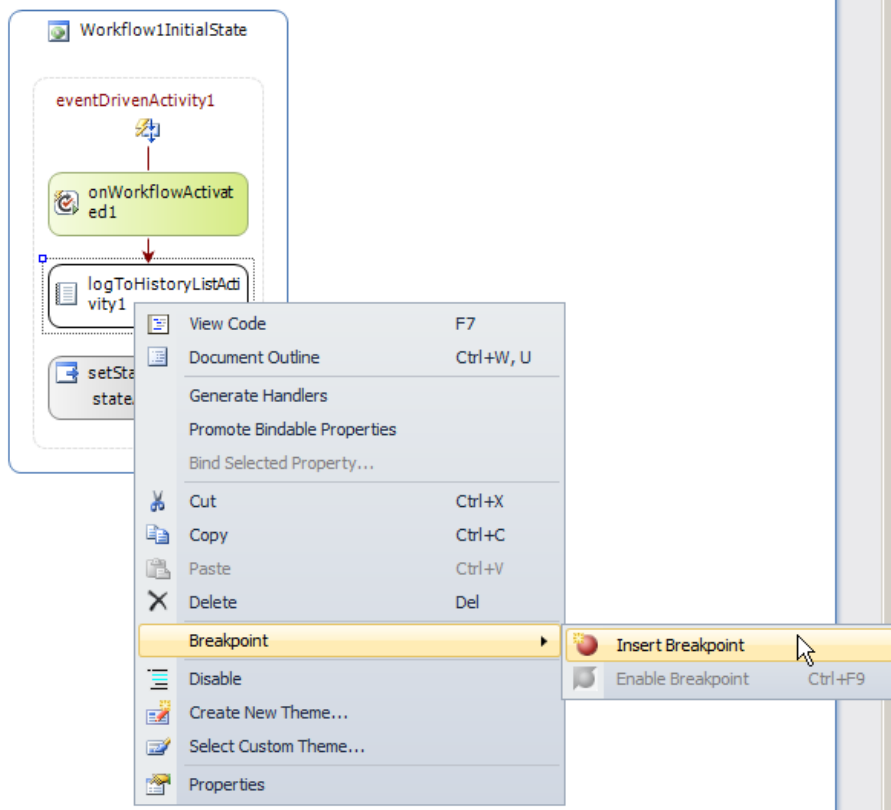
Sobald wir den Workflow Starten, zieht unser Breakpoint in der Methode **OnWorkflowActivated**

```
private void OnWorkflowActivated(object sender, ExternalDataEventArgs e)
{
    try
    {
        XmlDocument retData = new XmlDocument();
        retData.LoadXml(workflowProperties.InitiationData);
        XmlNode approverNode = retData.SelectSingleNode("//approver");
        if (null != approverNode)
        {
            _approverID = approverNode.InnerText;
        }
    }
}
```

Wir können jetzt im Einzelschritt (F10) durch den Code gehen. Irgendwann gelangen wir in den Workflow Designer und können dort ebenfalls im Einzelschritt weitergehen:



Ein Visual Studio lässt sich somit sowohl im Code, als auch im Designer debuggen. Auch im Designer lässt sich übrigens ein Breakpoint setzen. Dies erreicht man durch Klick mit der rechten Maustaste auf die Activity und Auswahl von **Breakpoint -> Insert Breakpoint**:



Der Workflow sollte nun so ablaufen, wie im [ersten Teil der Serie](#) beschrieben. Lediglich die Formulare sehen etwas anders aus, als dort beschrieben.

Fehlersuche

Bei einer derart guten Anleitung, wie dieser ;-)) kann man leicht auf die Idee kommen, dass alles mehr oder weniger reibungslos abläuft bei der Umsetzung von Visual Studio Workflows für SharePoint 2010. Die Wirklichkeit holt einem aber dann doch immer wieder sehr schnell ein und merkt, dass gerade das Gegenteil der Fall ist. Oftmals verbringt genauso viel Zeit mit Fehlersuche zu, als mit der eigentlichen Umsetzung. Und meistens hängt es nur an Kleinigkeiten.

Gerade bei der doch recht komplexen Entwicklung von Visual Studio Workflows und den vielen Fehlerquellen (nicht funktionierende Workflowumgebung, falsche Einstellungen im Designer, inkorrekt Code) sind einige grundlegende Dinge zum Troubleshooting sicherlich sehr hilfreich.

Fehler: Workflow wird im Workflow Designer nicht korrekt angezeigt

Ursache finden: Ist halt so. Kann jeden treffen. Bei komplexeren Designs öfters, als bei einfacheren

Korrigieren: Nochmal probieren. Meistens klappt es dann doch mit der Anzeige des Workflows in einem 2., 3. oder weiteren Versuch, bis man nicht mehr daran glaubt, dass das nochmal etwas werden könnte.

Beschreibung: Manchmal kommt es vor, dass der Workflow im Designer von Visual Studio nicht angezeigt wird und statt dessen eine Fehlermeldung erscheint. In den meisten Fällen hilft es einfach

nochmal zu probieren (ggf. mehrfach). Anscheinend ist Visual Studio an dieser Stelle noch etwas fehlerhaft. Sollte es trotz mehrfachem Versuch nicht funktionieren, sollte man sich überlegen, ob man den vom Designer erzeugten Code manuell geändert hat. Falls ja, sollte man die Änderungen nochmal rückgängig machen (Sourcecode Verwaltung sei Dank ;-) und dann nochmal probieren. Falls dies nicht der Fall ist bzw. nichts bringt, kann man auch versuchen, Visual Studio neu zu starten bzw. im schlimmsten Fall den Rechner bzw. das virtuelle Image. Hilft das auch nicht, bleibt wahrscheinlich nur eine Internet Recherche oder der Microsoft Support.

Workflow bricht mit allgemeinem Fehler ab

Ursache finden: SharePoint LOG

Korrigieren: Visual Studio

Beschreibung: Bricht der Workflow bereits vor dem eigentlichen Start oder direkt nach dem Startformular ab, hat man zumeist keine Chance mit Visual Studio zu debuggen. In diesem Fall passiert der Fehler innerhalb des sog. Workflow Host (in unserem Fall SharePoint) und die einzige Change, mehr über den Fehler zu erfahren, sind die LOG Dateien im LOG Verzeichnis von SharePoint (standardmäßig im Verzeichnis **C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\LOGS**).

Fehler beim Anzeigen des Aufgaben-Formulars

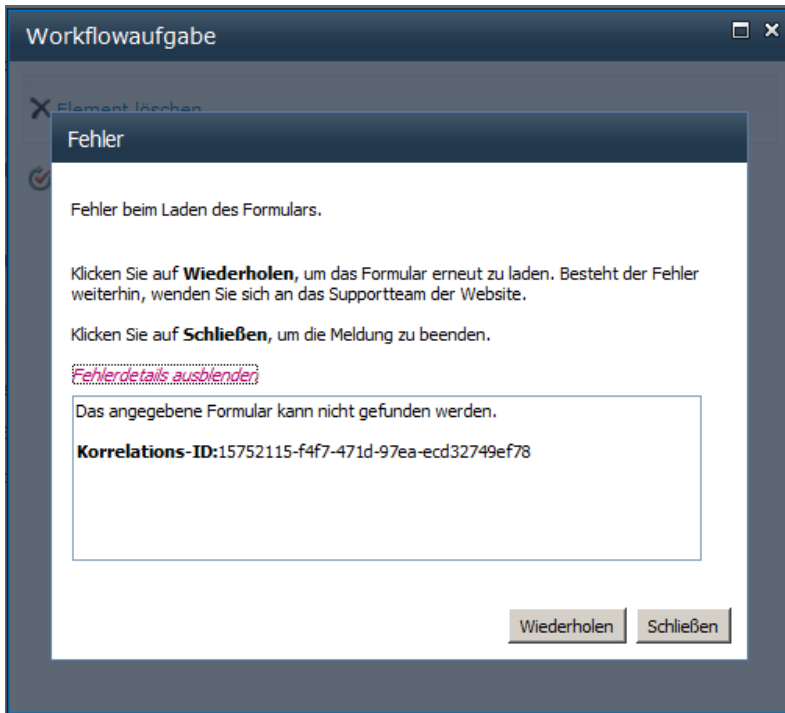
Ursache finden: SharePoint LOG

Korrigieren: Visual Studio

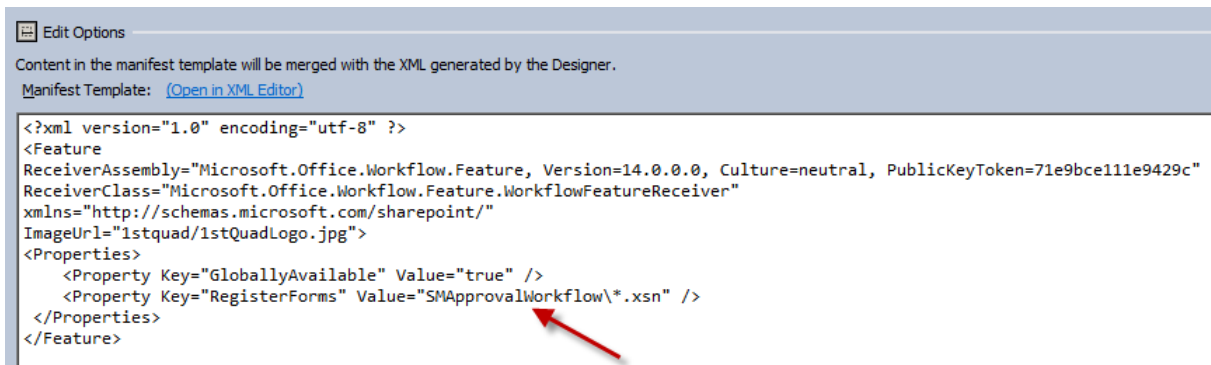
Ursachen:

- Formular ist nicht als **ElementFile** gekennzeichnet (siehe [Teil 7](#))
- Falscher Pfad in den Edit Options des Features stimmt nicht (siehe [Teil 7](#))

Beschreibung: Statt der Anzeige des Aufgabenformulars wird eine Fehlermeldung angezeigt.



Als erstes sollte man in der Zetraladministration nachsehen, ob die InfoPath Formulare für die Aufgaben auch korrekt installiert und als Workflow Formular gekennzeichnet sind (siehe [Anfang dieses Teils](#)). Sind die Formulare nicht vorhanden, sollte man nochmal die Visual Studio Einstellungen für die beiden Formulare kontrollieren, ob diese als **ElementFile** gekennzeichnet sind. Zusätzlich sollte man den Pfad auf die Formulare im Manifest des Workflow Features in den sog. Edit Options kontrollieren:



Vielleicht hat man die Formulare ja nochmal in ein Unterverzeichnis geschoben, oder der Workflow heißt im eigenen Projekt anders und man muss somit das Verzeichnis anpassen.

Sind die Aufgaben Formulare vorhanden und für die Verwendung in Workflows gekennzeichnet, bleibt zumeist nur noch das ULS LOG (siehe vorige Fehlerbeschreibung) anhand der **Correlation ID** die Stelle suchen.

Falsches Workflow Formular wird angezeigt

Ursache finden: Im Code des Workflows beim Anlegen einer Aufgabe

Korrigieren: Den TaskType korrekt setzen

Beschreibung: Wird das falsche Formular angezeigt, kann es sein, dass beim Anlegen der Aufgabe im Code der falsche **TaskType** gesetzt wird:

```
TaskProperties_Approver.TaskType = 0; // Nr. Des Aufgaben Formulars
```

Das jeweilige Formular ist ja in der Datei Elements.xml, wie im [Teil 6](#) beschrieben eingetragen. Der Zusammenhang ist dabei wie folgt:

```
Code:
TaskProperties_Approver.TaskType = 0;

Elements.xml
<MetaData>
  <AssociationCategories>List</AssociationCategories>
  <Task0_FormURN>urn:schemas-microsoft-com:office:infopath:Approve
  <Task1_FormURN>urn:schemas-microsoft-com:office:infopath:Review
  <StatusPageUrl>_layouts/WrkStat.aspx</StatusPageUrl>
</MetaData>
```

Workflow bricht beim Erstellen/Bearbeiten/Abschließen einer Aufgabe ab

Ursache finden: SharePoint LOG

Korrigieren: Im Workflow Designer/Code beim Setzen der TaskID

Beschreibung: TaskID ist evtl. falsch oder passt nicht zusammen mit bereits angelegter Aufgabe oder es wird die gleiche TaskID verwendet, wie für den Task zuvor.

Workflow bricht ab beim Aufruf des nächsten Zustandes

Ursache finden: SharePoint LOG

Korrigieren: Im Workflow Designer/Code beim Setzen des Correlation Token

Beschreibung: In einem State Machine Workflow muss jeder Zustand einen eigenen Correlation Token haben und dieser muss als Scope den Zustand haben und nicht den Workflow. Wird der Scope auf den Workflow gesetzt, kann der Zustand genau einmal aufgerufen werden. Ruft man ihn erneut auf, bekommt man eine Fehlermeldung im SharePoint LOG:

```
System.InvalidOperationException: Correlation value on declaration "ApproverToken" is already initialized. at
System.Workflow.Runtime.CorrelationToken.Initialize(Activity activity,
ICollection`1 propertyValues) at
Sys-
tem.Workflow.Activities.CorrelationService.InvalidateCorrelationToken(Activity activity, Type interfaceType, String methodName, Object[] messageArgs)
at
Sys-
tem.Workflow.Activities.CallExternalMethodActivity.Execute(ActivityExecutionContext executionContext) at
System.Workflow.ComponentModel.ActivityExecutor`1.Execute(T activity, ActivityExecutionContext executionContext) at
```

```
System.Workflow.ComponentModel.ActivityExecutor`1.Execute(Activity activity, ActivityExecutionContext executionContext) at System.Workflow.ComponentModel.ActivityExecutorOperation.Run(IWorkflowCoreRuntime workflowCoreRuntime) at System.Workflow.Runtime.Scheduler.Run()
```

Daten aus dem Workflow werden im Formular nicht angezeigt

Ursache finden: InfoPath Formular oder Visual Studio

Korrigieren: InfoPath oder Visual Studio

Beschreibung: Werden die Daten, die man in den ExtendedProperties des Workflows setzt, nicht im Aufgabenformular angezeigt, hat dies zumeist eine von 3 Ursachen:

- Der Feldname in der sekundären Datenquelle stimmt nicht mit dem verwendeten Namen in den ExtendedProperties überein (z.B. wenn in den Extended Properties der Name instructions übergeben wird, muss in der sekundären Datenquelle das Attribut **ows_instructions** verwendet werden:

```
TaskProperties_Approver.ExtendedProperties["instructions"] = _instructions;
```



```
<z:row xmlns:z="#RowsetSchema" ows_instructions="" ows_approver="" ows_comment="" />
```

- Es ist keine externe Datenquelle eingerichtet oder die Werte aus der Datenquelle werden nicht den entsprechenden Feldern zugewiesen. Hier muss man wie in [Teil 6](#) beschrieben, eine XML-Datei erstellen und diese als sekundäre Datenquelle einbinden.
- Man hat vergessen, das Feld aus der sekundären Datenquelle an den Wert eines Feldes der primären Datenquelle zu übergeben (siehe hierzu auch [Teil 6](#)).

Funktioniert andersherum der Abruf eines Wertes aus dem Formular nicht, hat dies zumeist die Ursache, dass die Angabe in den AfterProperties nicht mit dem Feldnamen im Formular übereinstimmt:

```
private void OnTaskChanged_Approver(object sender, ExternalDataEventArgs e)
{
    try
    {
        _status = AfterProperties_Approver.ExtendedProperties["status"].ToString();
        _comment = AfterProperties_Approver.ExtendedProperties["comment"].ToString();
    }
    catch (Exception ex) { }
}
```

Feld im InfoPath Formular muss **status** heißen

Feld im InfoPath Formular muss **comment** heißen

Damit ist unsere kleine Workflow Serie erst mal beendet. Ich hoffe, dass ich die Vorgehensweise anhand eines einfachen Workflows zeigen konnte.

Noch ein paar Hinweise in eigener Sache:

Wer noch tiefer in die Workflow Programmierung einsteigen will, der kann bei 1stQuad auch eine [spezielle Workflow Schulung zur Entwicklung von SharePoint 2010 Workflows mit Visual Studio 2010](#) besuchen. Wem das doch etwas zu viel Programmierung ist, für den bieten wir auch eine [Schulung für die Erstellung von No Code Workflows Hilfe von Visio, SharePoint Designer und InfoPath](#) an.