

# Drehbuch Azure-Vortrag Big>Days

---

## Vorbereitung vor dem Vortrag

1. Kopieren der Lösung aus Verzeichnis `Template Solution` in einen neuen Ordner
  - a. Falls kein lokaler SQL Server mit Defaultinstanz verwendet wird, müssen Connection Strings in `Web.config` und `App.config` angepasst werden
2. Lokaler SQL Server
  - a. Falls die DB lokal noch nicht existiert: Mit `Script Database Create Script.sql` aus `Verzeichnis Template Solution` anlegen.
  - b. Falls die DB lokal bereits existiert: Tabelle `Mission` entleeren:  
`truncate table Mission`
3. SQL Azure
  - a. Falls die DB in SQL Azure noch nicht existiert: Mit `Script Database Create Script.sql` aus `Verzeichnis Template Solution` anlegen.
  - b. Falls die DB in SQL Azure bereits existiert: Tabelle `Mission` entleeren:  
`truncate table Mission`
4. Local Storage
  - a. Ordner `c:\temp\azuretests` anlegen
  - b. Musterdatei `Anmeldebestaetigung.docx` muss in dem Ordner enthalten sein (Vorlage siehe `Verzeichnis Template Solution`)
  - c. Alle anderen Dateien aus Container entfernen
5. DevStorage (Zugriff mit *Azure Blob Storage Explorer*)
  - a. Container `templates` muss existieren
  - b. Musterdatei `Anmeldebestaetigung.docx` muss in dem Container enthalten sein (Vorlage siehe `Verzeichnis Template Solution`)
  - c. Alle anderen Dateien aus Container entfernen
6. Windows Azure Storage (Zugriff mit *CloudBerry Explorer for Azure Blob Storage*)
  - a. Container `templates` muss existieren
  - b. Musterdatei `Anmeldebestaetigung.docx` muss in dem Container enthalten sein (Vorlage siehe `Verzeichnis Template Solution`)
  - c. Alle anderen Dateien aus Container entfernen
7. Entwicklungsumgebung
  - a. Cloud DB Server in SQL Management Studio als *Registered Server* eintragen:  
`rx1ygpma6.database.windows.net`, UID=`HelfenHelfen`,  
PWD=`Helfen@He!fen`
  - b. DevFabrik starten
  - c. DevStorage starten
  - d. Verzeichnis `Snippets\Azure Workshop` in *Snippets Manager* für C# hinzufügen

## Vorstellung der Anwendung

Ziel: Ausgangspunkt zeigen.

1. Visual Studio 2010 *als Admin* starten
2. `Events.sln` in vorbereitetem Verzeichnis (siehe oben) öffnen.
3. Im Debugger starten
4. Hinweis auf *Development Webserver*, noch keine Cloud
5. Anmelden zu Demand 1
6. Generiertes Wordfile in `c:\temp\azuretests` zeigen
7. Eintrag in Tabelle `Mission` zeigen
8. Nochmals mit gleicher Emailadresse zu Demand 1 anmelden (Back-Button im Browser)
9. Hinweis auf Fehlermeldung

## Betrieb der Anwendung in der DevFabrik

Ziel: Unveränderte ASP.NET Anwendung in *DevFabrik* statt im *Development Webserver* betreiben.

1. Neues *Azure Cloud Service (C#)* Projekt namens `Events.Cloud` zur Solution hinzufügen
2. Sicher gehen, dass neues Projekt *Startup Project* ist.
3. Keine Rollen im Wizard hinzufügen!
4. Bestehendes Web-Projekt als *Web-Role* einfügen
5. Im Debugger starten
6. Hinweis auf *DevFabrik* statt *Development Webserver*
7. Anmelden zu Demand 2
8. Generiertes Wordfile in `c:\temp\azuretests` zeigen
9. Eintrag in Tabelle `Mission` zeigen

## Umstellen auf SQL Azure

Ziel: *DevFabrik* als Webserver, *SQL Azure* als Datenspeicher.

1. Connection String aus *Web.config* in *cscfg/csdef* Files kopieren. Eintrag muss `ConnectionString` heißen.
  - a. **Achtung:** Verwendet man das VS UI zum Kopieren des Connection Strings, wird `&quot;` durch `&amp;quot` ersetzt → Fehler, muss in *cscfg/csdef* Files korrigiert werden!
2. `Default.aspx.cs` öffnen und zeigen, wie EF-ObjectContext mit `new` erzeugt wird.
3. `Events.Data/EventModelProcs.cs` öffnen
4. Factorymethode `Create` anlegen. Snippet `AzureContextCreate` verwenden!
5. Nach `new HelfenHelfenEntities()` suchen
6. Vorkommen in den ASP.NET Code Behind Dateien durch Aufruf der Factorymethode ersetzen:
  - a. `Default.aspx.cs`:

```
this.context = HelfenHelfenEntities.Create();
```
  - b. `RegisterForEvent.aspx.cs`:

```
using (var context = HelfenHelfenEntities.Create())
```

7. Breakpoint in Factorymethode setzen, Debugger starten und verwendete Azure SDK Methoden erklären.
8. Connection String durch Connection String auf SQL Azure ersetzen.
  - a. Connection String aus Datei `Connection String.txt` kopieren
9. Im Debugger starten
10. Anmelden zu Demand 3
11. Verbindung zu SQL Azure in *SQL Management Studio* zeigen.
12. Eintrag in Tabelle `Mission` in *SQL Azure* zeigen

## Filesystem und Blob Storage

Ziel: Lokales Dateisystem durch Blob Storage ersetzen.

### Filesystem

1. Mit neuem *Navigate To* Feature von VS 2010 (Strg + ,) zu Methode `CreateRegistrationConfirmation` navigieren.
2. Zeigen wie dort ein temporäres File zum Generieren der Anmeldebestätigung verwendet wird
  - a. Hinweis nicht vergessen, dass das auch mit Memory Streams möglich wäre und nicht unbedingt temporäres File notwendig ist. Hier zu Demozwecken mit temporärem File.
3. In Rolleneigenschaften *Local Storage* von 10MB namens `TemporaryFiles` anfordern.
4. Die Zeile `tempPath = @"C:\temp\azuretests";` durch folgenden Codeteil ersetzen (kein Snippet weil so kurz):
 

```
if (RoleEnvironment.IsAvailable)
{
    tempPath =
RoleEnvironment.GetLocalResource("TemporaryFiles").RootPath;
}
else
{
    tempPath = @"C:\temp\azuretests";
}
```
5. Erklären, warum lokales Filesystem nur für temporäre Dateien Sinn macht.

### BLOB Storage

1. Mit neuem *Navigate To* Feature von VS 2010 (Strg + ,) zur Datei `LocalTemplateManager` navigieren.
2. Zeigen, wie dort die Musterdatei mit Dateioperationen geladen wird.
3. Neue Klasse namens `CloudTemplateManager` zum `Events.Data` Projekt hinzufügen.
4. Gesamte Datei mit Snippet `AzureCloudTemplateManager` ersetzen (implementiert die Klasse)
5. In Rolleneigenschaften BLOB Connection String namens `BlobStorageConnectionString` hinzufügen. Zeigt auf *DevStorage*.
  - a. Im *Azure Blob Storage Explorer* zeigen, dass in *DevStorage* Word-Vorlage liegt.
6. BLOB Zugriffsfunktionen erklären.
7. Mit F12 (*Go To Definition*) Basisklasse von `CloudTemplateManager` öffnen.

8. Factorymethode **Create** durch folgenden Codeteil ersetzen (kein Snippet weil so kurz):
 

```
if (RoleEnvironment.IsAvailable)
{
    return new CloudTemplateManager();
}
else
{
    return new LocalTemplateManager();
}
```
9. Im Debugger starten
10. Anmelden zu einem Event
11. Im *Azure Blob Storage Explorer* generierte Datei zeigen.
12. BLOB Connection String durch Connection String auf Windows Azure ersetzen.
  - a. Connection String aus Datei **Connection String.txt** kopieren
13. Im Debugger starten
14. Anmelden zu einem Event
15. Im *CloudBerry Explorer for Azure Blob Storage* generierte Datei zeigen.

## Worker Role einfügen

Ziel: Generierung der Anmeldebestätigung von UI entkoppeln.

1. Neues Worker Role Projekt namens **Events.Worker** zu Solution hinzufügen.
2. **Referenzen** in Worker Role Projekt einfügen:
  - a. Auf Events.Data
  - b. Auf System.Data.Entity
  - c. Auf Microsoft.WindowsAzure.StorageClient
  - d. Auf DocumentFormat.OpenXml (mit Copy Local = true!)
3. **Konfigurationseinstellungen** in *cscfg/csddef* Files von Web Role in die Worker Role kopieren.
4. *Worker Role Loop* in WorkerRole.cs durch Snippet **AzureWorkerRole** ersetzen.
5. Usings hinzufügen (inkl. using Microsoft.WindowsAzure.StorageClient).
6. Mit neuem *Navigate To* Feature von VS 2010 (Strg + ,) zur Datei **CloudTemplateManager** navigieren.
7. Funktionen zum Zugriff auf die Queue mit Snippet **AzureQueue** einfügen.
8. *Queue Client* in **Konstruktor** zuweisen:
 

```
this.cloudQueueClient =
cloudStorageAccount.CreateCloudQueueClient();
```
9. Feld für cloudQueueClient mit **Strg + .** anlegen (neue VS 2010 Funktion)
10. In *EventManagerProcs.cs* Funktion einfügen (kein Snippet weil so kurz):
 

```
public void QueueRegistrationConfirmation(Guid missionId)
{
    TemplateManager.Create().QueueRegistration(missionId);
}
```
11. In *RegisterForEvent.aspx.cs* die Zeile
 

```
context.SendRegistrationConfirmation(missionId)
```

 durch folgenden Code ersetzen (kein Snippet weil so kurz):
 

```
if (RoleEnvironment.IsAvailable)
```

```
{
    context.QueueRegistrationConfirmation(missionId);
}
else
{
    context.SendRegistrationConfirmation(missionId);
}
```

12. Im Debugger starten

13. Anmelden zu einem Event

14. In *DevFabrik UI* zeigen, wie auf die Message reagiert wird

15. Im *CloudBerry Explorer for Azure Blob Storage* generierte Datei zeigen.

## Ab in die Cloud

Deploy...