

Entwicklung eines Event-Managers mit ASP.NET 1.1 und C#

Alexander Zeitler

MVP Visual Developer ASP.NET

<http://alexonasp.net>

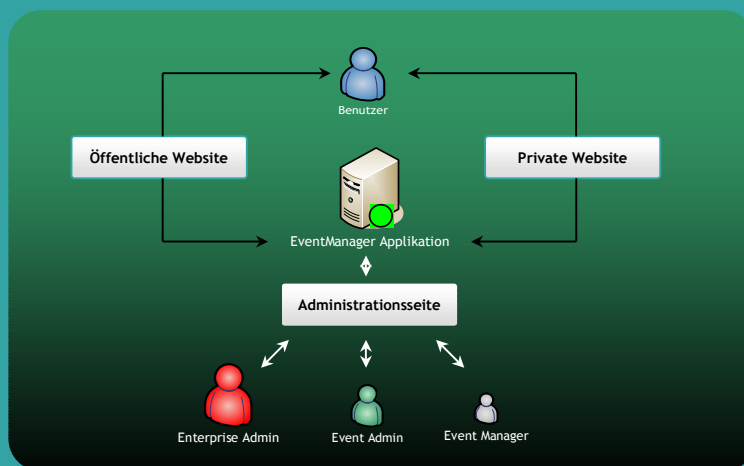
- Anforderungen funktionell
- Anforderungen technisch
- Aufbau der Applikation
- Datenbankstruktur
- Datenbankschicht
- Geschäftslogik
- Präsentationsschicht

- Verwalten von Events (+ Sessions etc.)
 - hinzufügen
 - bearbeiten
 - löschen
 - rollenbasiert

Demo: Administration

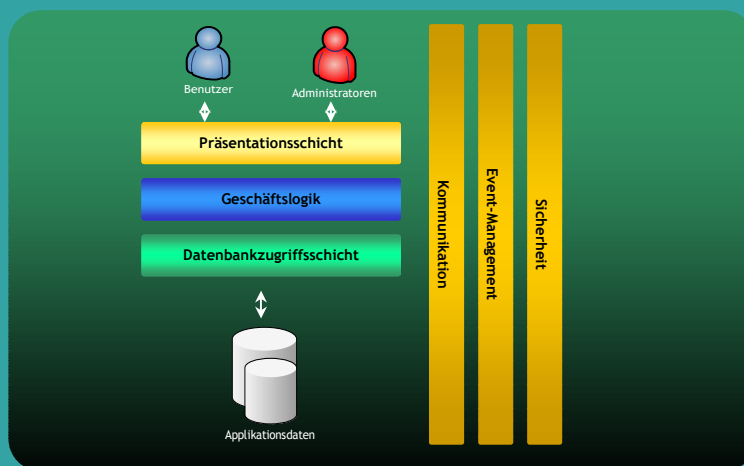


- Besucherregistrierung und Anmeldung
 - Login-geschützt
 - (Selbst-)Verwaltung der Besucherdaten
 - Profil
 - Passwort
 - An- / Abmeldung zu den Events

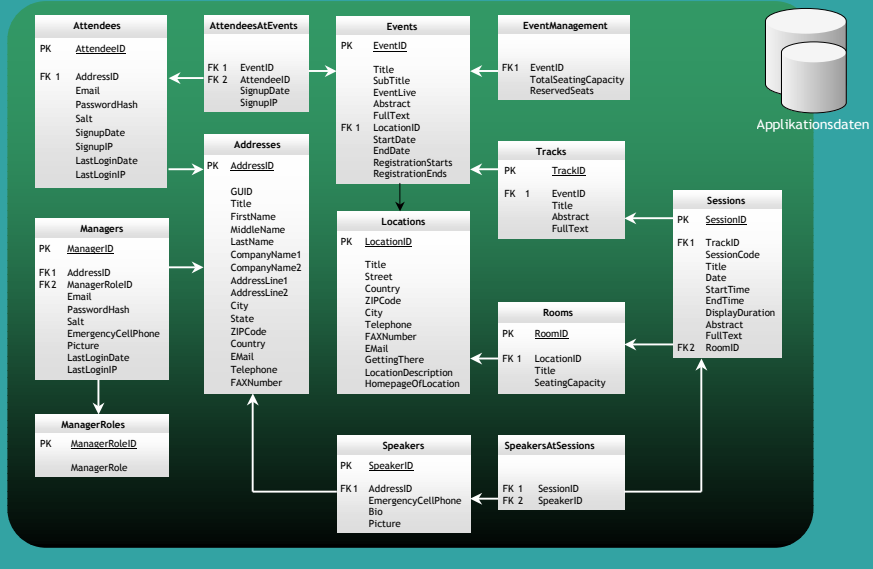




- Webbasiert
- Browserunabhängig
- Flexibel in der Layoutgestaltung
- Skalierbar
- Datenbankunabhängig



.NET Community Conferences
in D, A und CH
von .NET-Entwicklern für .NET-Entwickler



.NET Community Conferences
in D, A und CH
von .NET-Entwicklern für .NET-Entwickler



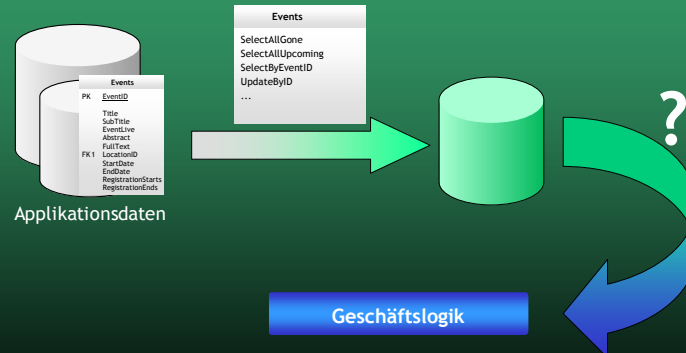
Attendees SelectAllAtEventByEventID SelectByID UpdateByID UpdateLogin GetByEmail ...	Addresses AddNew UpdateByID DeleteByID ...	Events SelectAllGone SelectAllUpcoming SelectByEventID UpdateByID ...
Sessions SelectBySessionID SelectBySpeakerID SelectByTrackID UpdateByID ...	Speakers AddNew SpeakerAtSessionAddNew SelectByEventID SelectBySessionID SelectBySpeakerID

```

SELECT
    Speakers.SpeakerID, Speakers....,
    Addresses.GUID, Addresses....
FROM
    Addresses
INNER JOIN
    Speakers
ON
    Addresses.AddressID = Speakers.AddressID
WHERE
    (( (Speakers.SpeakerID)=[@SpeakerID] ) );
    
```



Wie gelangen die Daten von der Datenbank in die Applikation?



Klassischer Ansatz mit ADO.NET

```
// Connection erzeugen
OleDbConnection MyNWConn =
    new
OleDbConnection("PROVIDER=Microsoft.Jet.OLEDB.4.0;Data
Source=" + Server.MapPath("~/db/Events.mdb"));

// DataSet erzeugen
DataSet MyDataSet = new DataSet();

// OleDbDataAdapter und OleDbCommand erzeugen
OleDbDataAdapter oCommand = new OleDbDataAdapter();
OleDbCommand oledbcmd = new OleDbCommand();

// Stored Procedure definieren
oledbcmd.CommandType = CommandType.StoredProcedure;
oledbcmd.CommandText = "EventsSelectAllUpcoming";

// Connection und Stored Procedure zuweisen
oledbcmd.Connection = MyNWConn;
oCommand.SelectCommand = oledbcmd;

// DataSet befüllen
oCommand.Fill(MyDataSet, "Events");

// Connection schließen
MyNWConn.Close();
```

Klassischer Ansatz mit ADO.NET - Nachteile

Je Funktion nur ein Datenbanktyp erreichbar:

- OleDb ODER
- Odbc ODER
- SqlServer

Das bedeutet:

- für jede Datenbank neue Zugriffsmethoden schreiben
- wiederholen von bereits erzeugtem Code (Redundanz!)

Die logische Frage: Geht das nicht einfacher?

Die Antwort: ja - mit DAAB!

Was bedeutet DAAB?

DAAB = Data Access Application Block

Woher kommt DAAB?

DAAB wird von Microsoft bereitgestellt

Was ist DAAB?

- Ausgereifte, kostenlose .NET-Komponente (aktuell v3.x)
- Liefert ein einfaches Modell zum Datenzugriff via ADO.NET
SqlClient, OleDb und Odbc
- Unterstützt SqlServer UND Access-Datenbanken
- Erweiterbar um weitere Datenbanken (Oracle, etc.)
- Liefert Methoden für die gängigen Datenbankzugriffe
- Performance-optimiert
- Unterstützt SQL-Text-Abfragen und Stored Procedures

DAAB - Features

- Liefert viele Datenformate zurück:
 - DataSets
 - DataReader
 - Scalars
 - XmlReader
- Führt Abfragen ohne Rückgabewerte aus
- Alles in einer einzigen Codezeile!

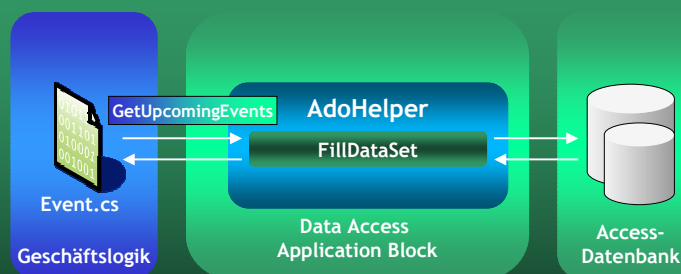


DAAB - Features

- Vereinfachter Aufruf über verschiedene Datenquellen:
 - Connections
 - ConnectionStrings
 - SQL Transaktionen
- Unterstützt strongly Typed DataSets via FillDataSet-Methode
- Unterstützt das Schreiben von DataSet-Updates in Datenbank
- Bietet zusätzliche Helper-Methoden für DataRow Parameter



DAAB - Beispiel



```
AdoHelper.FillDataset(  
    ConnectionString,  
    CommandType.StoredProcedure,  
    "EventsSelectAllUpcoming",  
    ds,  
    new string[] {"Events"});
```

DAAB - leider nur die halbe Miete

- DAAB stellt nur Methoden für verschiedene Provider bereit - nicht aber den Handler zur Auswahl des gewünschten Providers
- Entwicklung eines eigenen Layers zur Erzeugung der providerspezifischen DAAB-Objekt-Instanzen:
 - Connections
 - DAAB-Helper (=AdoHelper)
 - Abfrage-Parameter

DALBase - Factory

- Liest aus der `web.config` den eingestellten `DataProviderType` (Access, SqlServer, Odbc)
- Erzeugt den entsprechenden `AdoHelper`
- Liest den zugehörigen `ConnectionString` aus der `web.config` und stellt ihn als allgemeinen `ConnectionString` bereit



DALBase.cs

```
public abstract class DALBase
{
    protected static string ConnectionString = string.Empty;
    protected static AdoHelper DALHelper;

    static DALBase()
    {
        // Strings für Provider Assembly und Type erzeugen
        string assembly = string.Empty;
        string type = string.Empty;

        // Bestimmung des ProviderTypes
        switch( Configuration.ProviderType ) {
            case DataProviderType.OleDb:
                assembly = Configuration.OleDbHelperAssembly;
                type = Configuration.OleDbHelperType;
                ConnectionString = Configuration.OleDbConnectionString;
                break;
            case DataProviderType.SqlServer:
                // weitere ProviderTypes für SqlServer und Odbc
                ...
        }

        // DALHelper erzeugen, d.h. entsprechenden AdoHelper instanzieren
        DALHelper = AdoHelper.CreateHelper( assembly, type );
    }
}
```



Verwenden von DALBase.cs

Beispiel: Zukünftige Events aus der Datenbank lesen

```
public class DALEvents : DotNetGerman.Data.DALBase
{
    public static EventsDataSet GetEventByID(int EventID)
    {
        // neues Typed DataSet erzeugen
        EventsDataSet ds = new EventsDataSet();

        // DataSet per AdoHelper befüllen
        DALHelper.FillDataSet(
            ConnectionString,
            CommandType.StoredProcedure,
            "EventsSelectByEventID",
            ds,
            new string[] { "Events" },
            DALParameter.Create("@EventID", EventID));

        // befülltes DataSet zurückgeben
        return ds;
    }
}
```

Verwenden von Abfrage-Parametern mit DALBase.cs

Ähnlicher Lösungsansatz wie bei DALBase

DALParameter.cs:

```
public class DALParameter
{
    public static IDataParameter Create(
        string ParameterName, object Value)
    {
        // welcher ProviderType?
        switch(Configuration.ProviderType) {
            case DataProviderType.OleDb:
                // entsprechenden Parameter-Typ erzeugen
                return new OleDbParameter(ParameterName, Value);

            case DataProviderType.Odbc:
                return new OdbcParameter(ParameterName, Value);
            case DataProviderType.SqlServer:
                return new SqlParameter(ParameterName, Value);
        }
        return null; // ungültiger Parametertyp
    }
}
```

Da war doch noch was?

```
public class DALEvents : DotNetGerman.Data.DALBase
{
    public static EventsDataSet GetEventByID(int EventID)
    {
        // neues strongly Typed DataSet erzeugen
        EventsDataSet ds = new EventsDataSet();

        // DataSet per AdoHelper befüllen
        DALHelper.FillDataset(
            ConnectionString,
            CommandType.StoredProcedure,
            "EventsSelectByEventID",
            ds,
            new string[] { "Events" },
            DALParameter.Create("@EventID", EventID));

        // befülltes DataSet zurückgeben
        return ds;
    }
}
```

Typed DataSets

■ Nachteile normaler DataSets

■ Keine Typsicherheit

- `DataSet ds = new DataSet();`
- `DataTable dt = ds.Tables[„Events“];`
- `DataRow dr = dt.Rows[„EventID“];`

■ Anfällig gegen Tippfehler - erst zur Laufzeit feststellbar

■ Keine Unterstützung durch IntelliSense in VisualStudio.NET

■ Typed DataSets lösen diese Probleme!

■ typsicher

■ Fehler zeigen sich bereits beim Kompilieren

Demo: Typed DataSet erzeugen

Demo: Implementierung der Geschäftslogik

Überprüfung unserer Anforderungen an die Applikation

- Datenbankunabhängig ✓
- Skalierbar ✓
- Webbasiert ✗
- Browserunabhängig ✗
- Flexibel in der Layoutgestaltung ✗



Zum letzten Mal Theorie: Page Templates (Master Pages)

■ Vorteile

- Eine (oder mehrere) Vorlagen für mehrere layoutgleiche Seiten
- Implementierung gleichartiger Funktionen (z.B. Menüs) nur einmal nötig

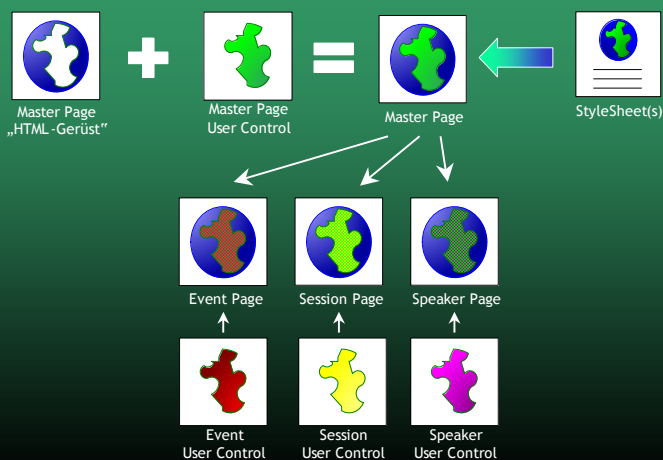
■ Nachteil

- Es existiert (aktuell) kein einheitlicher Lösungsansatz für Master Pages
 - Page-Klasse, von der abgeleitet wird
 - Xml-basierte Ansätze
 - Implementierung via User Controls
- Keine richtige Unterstützung durch VisualStudio.NET

Lösungsansatz EventManager: Page Templates mittels User Controls



Schema: Page Templates mit User Controls



Demo: Implementierung der Präsentationsschicht

Fragen?



Guten Appetit ;-)