WIE MAN MIT LOCALDB, FAKES UND ANDEREN ZUTATEN TESTBARERE .NET SYSTEME BEKOMMT

Entwicklertag 2014

von Andreas Bräsen

ZIEL DES VORTRAGES

Es soll gezeigt werden, wie man ein auf .NET basiertes System so erweitert, dass es besser testbar wird….

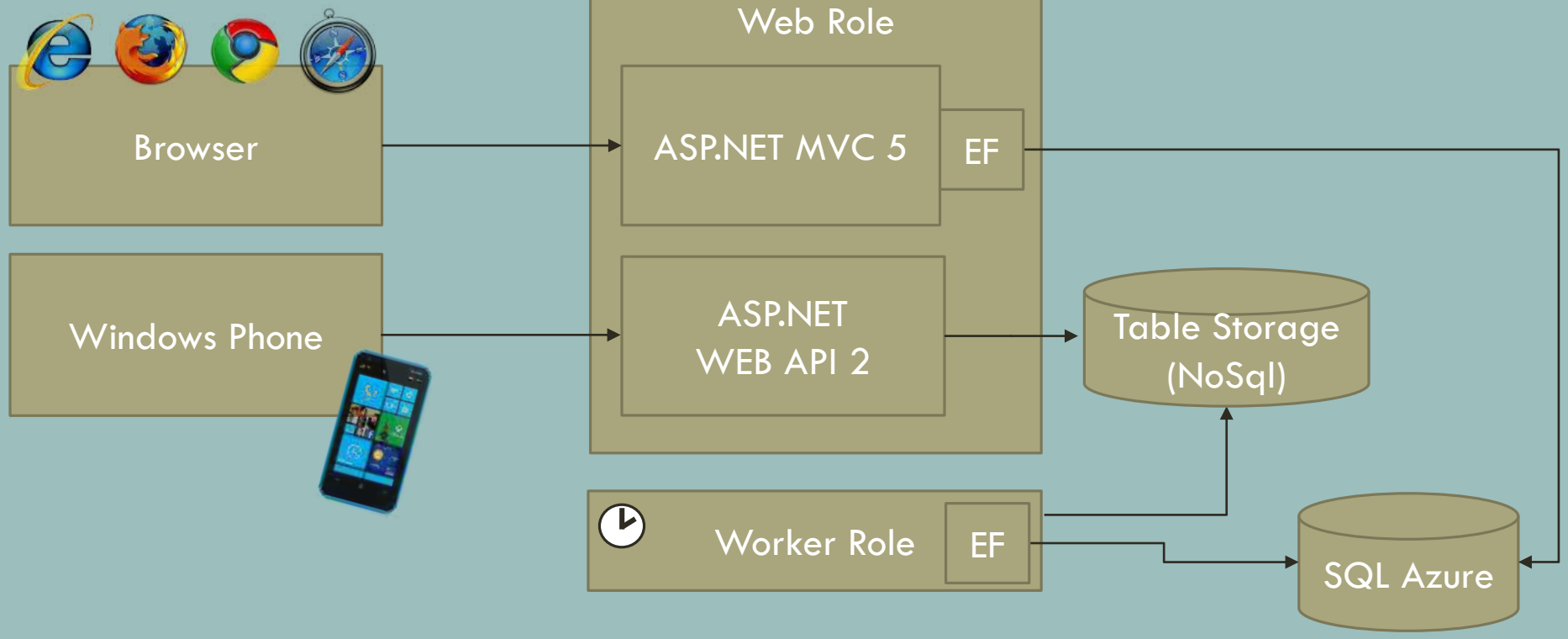
AGENDA

Das Beispiel-System

Die erforderlichen Schritte / Zutaten…

- IOC
- Fakes
- Local DB
- Self Host
- UI Tests

DAS BEISPIEL-SYSTEM



STAND BEIM TESTEN…

Aktuell kann ich nur die Tests schreiben, welche das System mit den externen Systemen (TableStorage und SQL Azure) zusammen testen kann.

=> Blöd! Dann kann ich irgendwie nur manuell testen

GEFÜHLTE VERTEILUNG ÜBER DIE ZEIT…

FRAGE

Wie kann ich jetzt das System testbar machen, so dass es z.B. durch automatische Unit Tests nach dem Build getestet werden kann?

ANTWORT

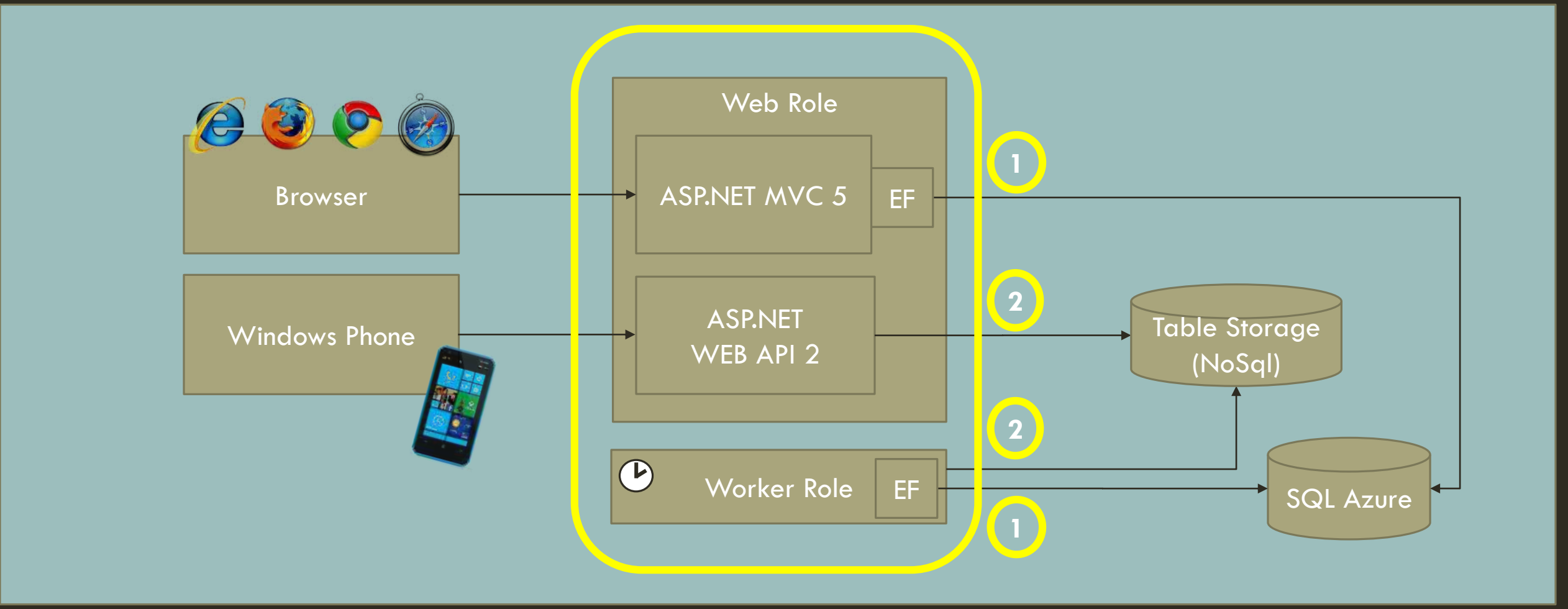
Es sind relative wenige Schritte/Zutaten erforderlich….

SCHRITT 1

Identifiziere die benutzten externen Systeme.

=> Werde unabhängig von externen Systemen, sonst wird das Testen wie ein Shooting on a moving Target

DAS BEISPIEL-SYSTEM



FRAGE

Wie werde ich externe Systeme los?

ANTWORT

Ziehe für jedes externe System ein Interface ein und abstrahiere somit das externe System….

Zugriff auf den Tablestorage

```
[Route("api/v1/AddGame")]
public bool AddGame(JObject gameData, Guid commandId) {
 …
 var tableStorageConnectionString = CloudConfigurationManager.GetSetting(„TSCS");

 var cloudStorageAccount = CloudStorageAccount.Parse(tableStorageConnectionString);

 var cloudTableClient = cloudStorageAccount.CreateCloudTableClient();

 var tableReference = cloudTableClient.GetTableReference("EventStore");

 tableReference.CreateIfNotExists();

 var commandStoreEntry = new CommandStoreTableEntry{ … }

 var insertOrReplaceTableOperation = TableOperation.InsertOrReplace(commandStoreTableEntry);

 var tableResult = tableReference.Execute(insertOrReplaceTableOperation);

 return tableResult == null || (tableResult.HttpStatusCode >= 200 &&
                                tableResult.HttpStatusCode <= 299);
 }
```

INoSqlStorage

```
public interface INoSqlStorage
{
    int InsertOrReplace(CommandStoreEntry commandStoreEntry);
}
```

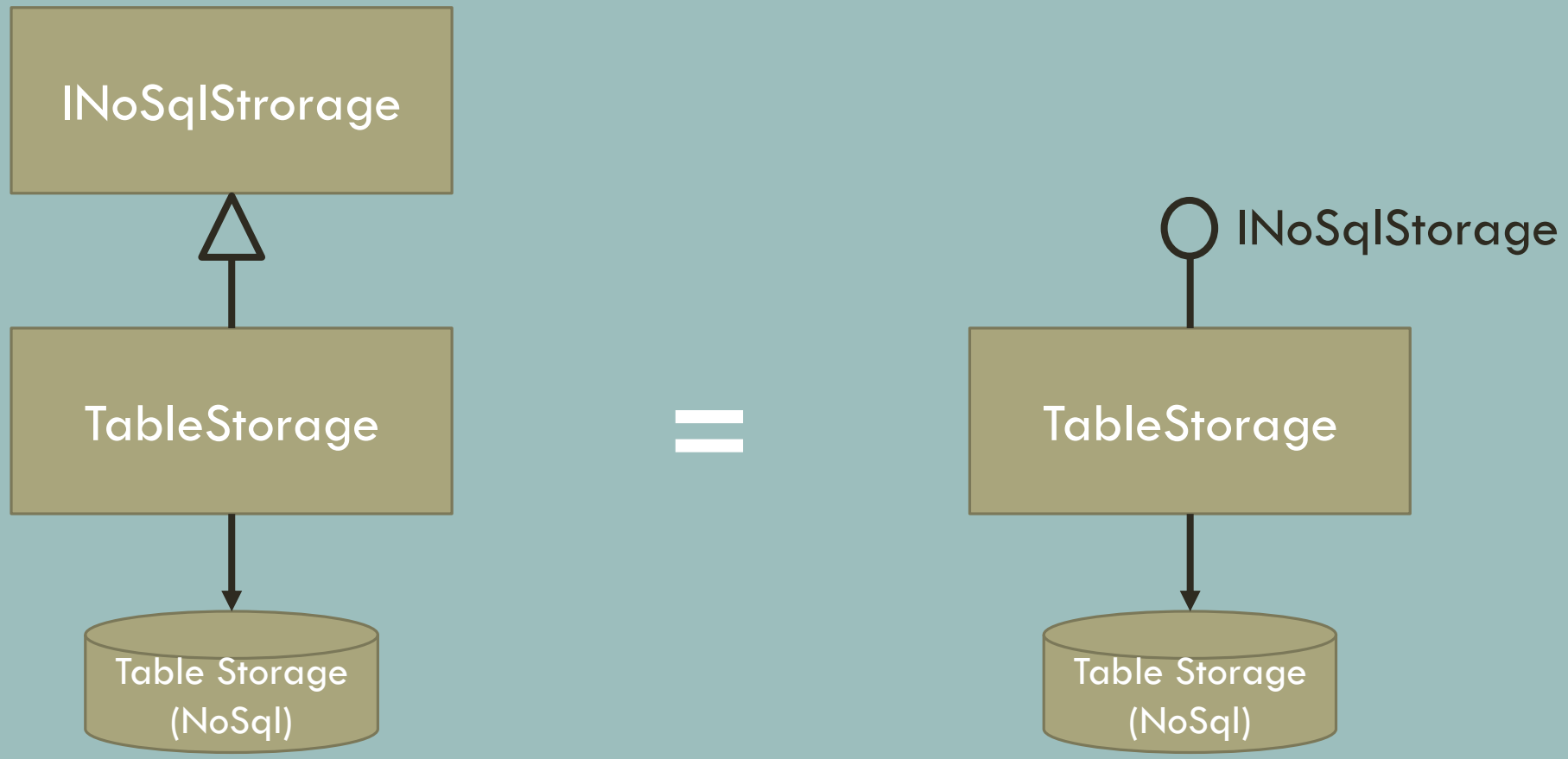
TableStorage : INoSqlStorage

```
internal class TableStorage : INoSqlStorage{
 public int InsertOrReplace(CommandStoreEntry commandStoreEntry){
  var tableStorageConnectionString = CloudConfigurationManager.GetSetting("TSCS");
  var cloudStorageAccount = CloudStorageAccount.Parse(tableStorageConnectionString);
  var cloudTableClient = cloudStorageAccount.CreateCloudTableClient();
  var tableReference = cloudTableClient.GetTableReference("EventStore");
  tableReference.CreateIfNotExists();
  var commandStoreTableEntry = CreateCommandStoreEntry(commandStoreEntry);
  var insertOrReplaceTableOperation =
     TableOperation.InsertOrReplace(commandStoreTableEntry);
  var tableResult = tableReference.Execute(insertOrReplaceTableOperation);
  return tableResult == null ? -1 : tableResult.HttpStatusCode;
  }
}
```

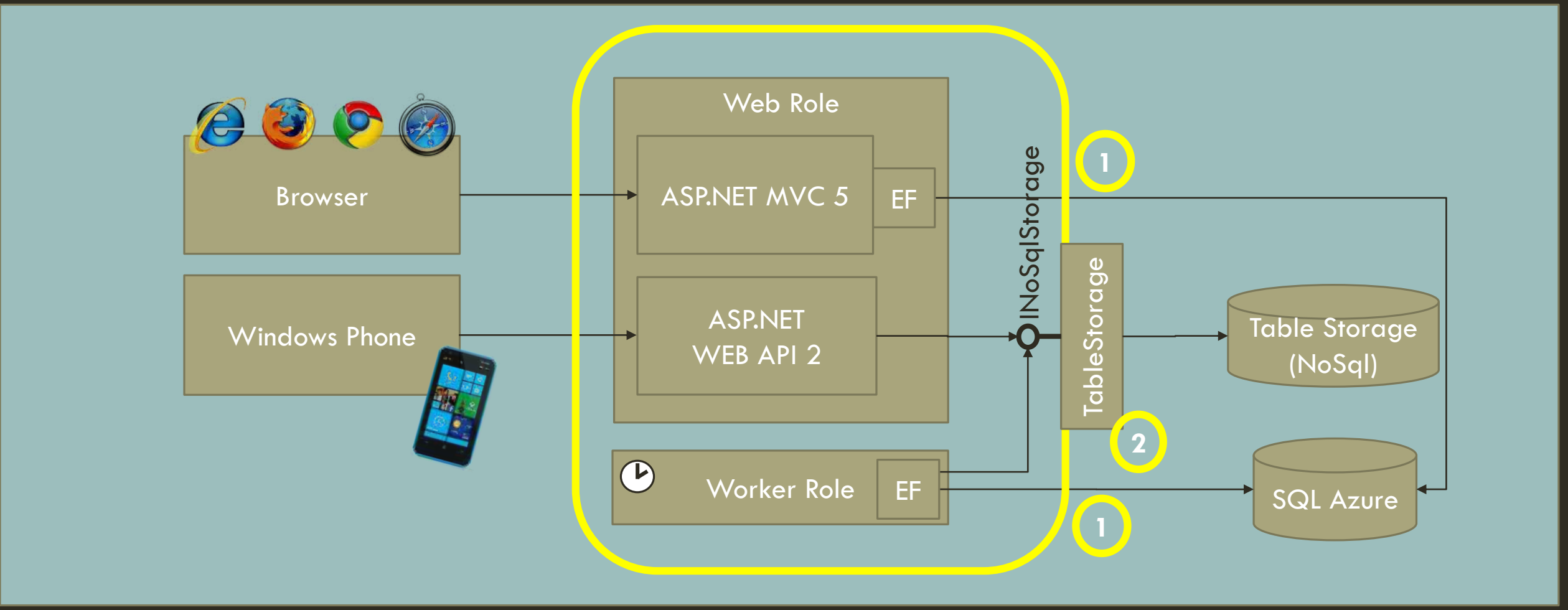
Zugriff auf den Tablestorage über INoSqlStorage

```
[Route("api/v1/AddGame")]
public bool AddGame(JObject gameData, Guid commandId) {

  INoSqlStorage noSqlStorage = new TableStorage();

  var commandStoreEntry = new CommandStoreEntry{ … }

  var httpStatusCode = noSqlStorage.InsertOrReplace(commandStoreEntry);

  return (httpStatusCode >= 200 && httpStatusCode <= 299);

}
```

WAS WURDE GEMACHT…



DAS BEISPIEL-SYSTEM



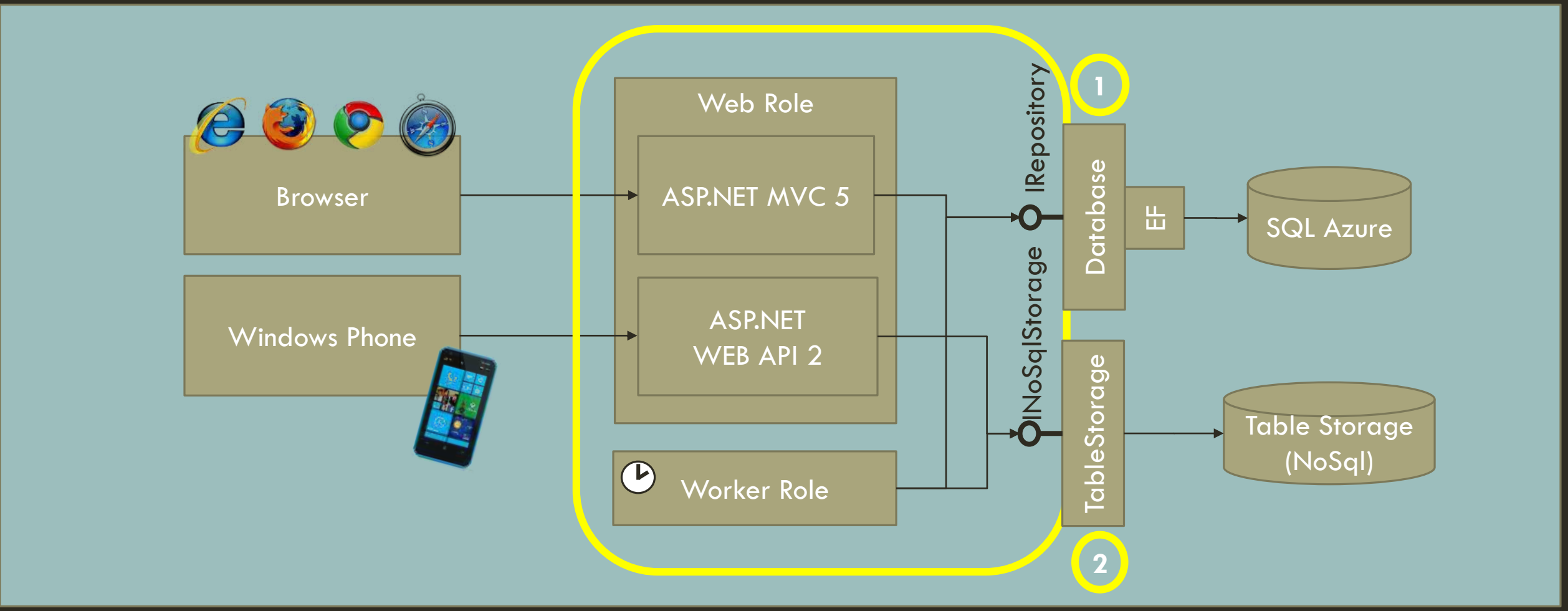
FRAGE

Was mache ich mit der Datenbank (SQL Azure)

ANTWORT

Auch hier abstrahiert wir das Funktionalität und führen ein IRepository ein.

DAS BEISPIEL-SYSTEM



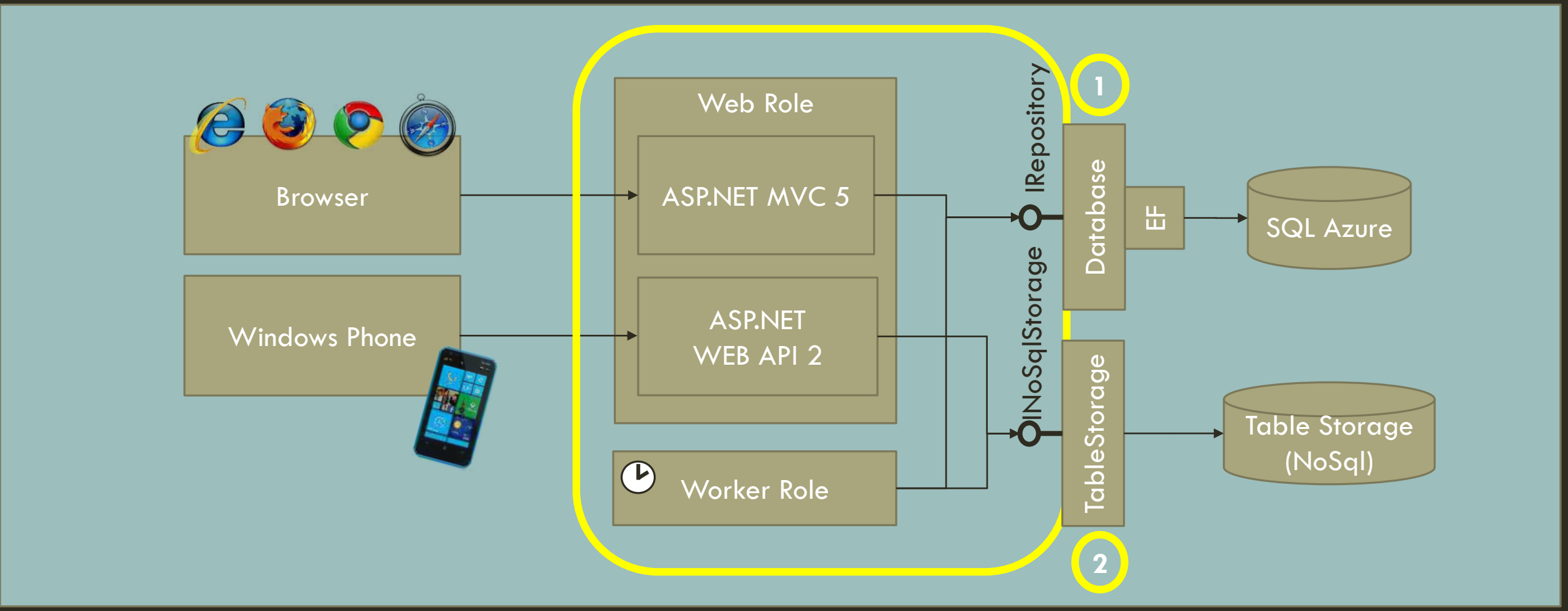
Entity Framework und Linq

```
var foundSpiele =
  (from i in _bomContainer.SpielSet
   where i.Name == spieleName
   select i).ToList();
```

IRepository

```
public interface IRepository{
  …
   List<Spiel> SearchSpiele(string spieleName);
  …
}
```

DAS BEISPIEL-SYSTEM



SCHRITT 2

Abstrahiere die externen Systeme führe für jedes ein Interface ein.

STAND BEIM TESTEN…

Ich bin jetzt theoretisch in der Lage autoamtisierte Tests zu schreiben, die ohne die externen Systeme auskommen …. Aber es fehlt noch ein wenig.

=> Oh man aey….

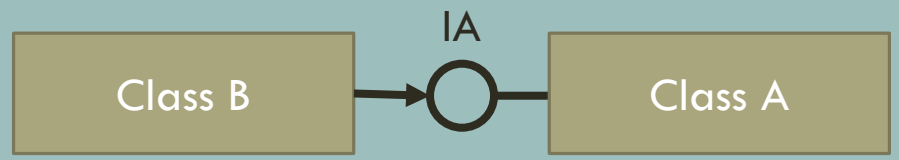
FRAGE

Wie bekomme ich die Instanz zu einem Interface in die Klasse, welche die jeweilige Funktionalität benutzt ?

ANTWORT

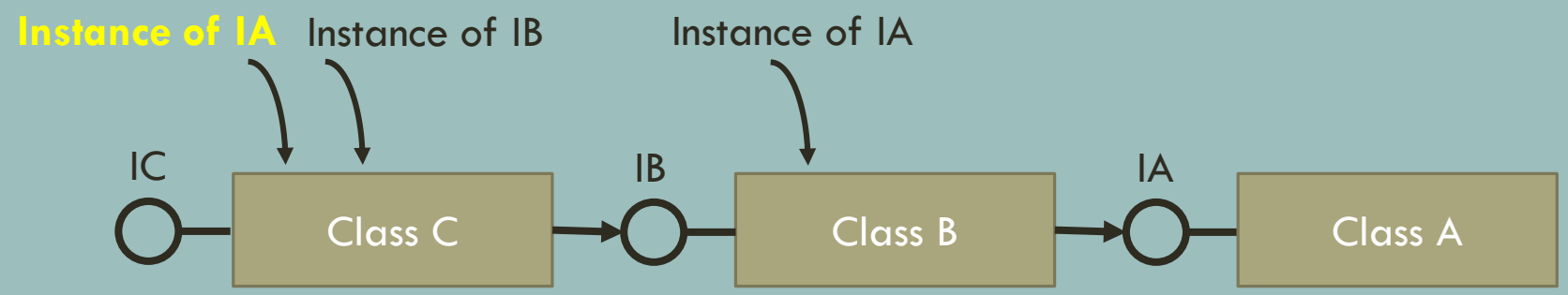
Man gibt eine Instanz, welche das Interface implementiert über den Constructor mit rein.

CONSTRUCTOR INJECTION




```
public class B{
   private IA _instanceOfA;

   public B(IA instanceOfA){
      _instanceOfA = instanceOfA;
   }
   …
}
```

DAS PROBLEM MIT DER "CONSTRUCTOR INJECTION"



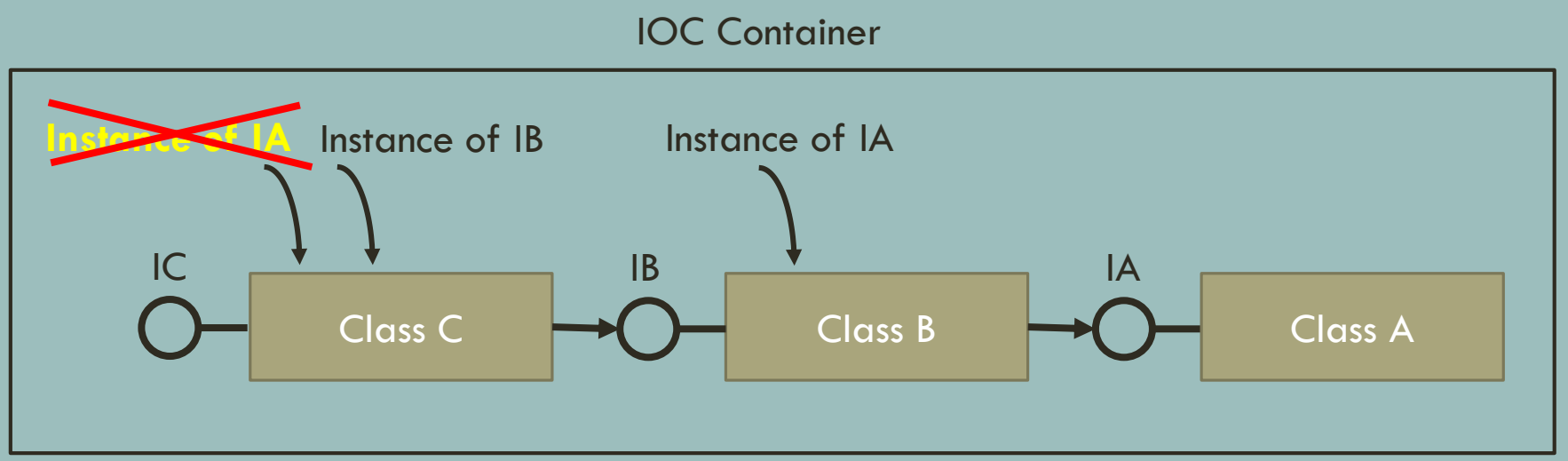
DIE LÖSUNG ZU DEM PROBLEM... IOC CONTAINER

Inversion **O**f **C**ontrol Container

z.B. Unity von Microsoft Pattern & Practice Group,

Ninject, Castle Windsor, …

IOC CONTAINER



Configuration
IA => class A
IB => class B
IC => class C

CONSTRUCTOR INJECTION

Configuration
IA => class A
IB => class B
IC => class C

```
var unityContainer = new UnityContainer();
unityContainer.RegisterType<IA,A>();
unityContainer.RegisterType<IB,B>();
unityContainer.RegisterType<IC,C>();
```

ASP.NET MVC / WEB API VS IOC CONTAINER

ASP.NET MVC / WEB API ist dafür schon vorbeteitet. Das Schlüsselwort dazu ist der DependencyResolver.

```
var unityContainer = new UnityContainer();

…

// Set the Web API dependency
Resolver.GlobalConfiguration.Configuration.DependencyResolver =
   new Unity.WebApi.UnityDependencyResolver(unityContainer);

// Set the MVC Dependency Resolver
DependencyResolver.SetResolver(
   new Unity.Mvc5.UnityDependencyResolver(unityContainer));
```

SCHRITT 3

Einführung eines IOC Containers.

STAND BEIM TESTEN…

Ich kann jetzt schon mal alles zusammenstecken, aber bringt mich das wirklich beim Testen weiter?

=> Sind wir immer noch nicht da?

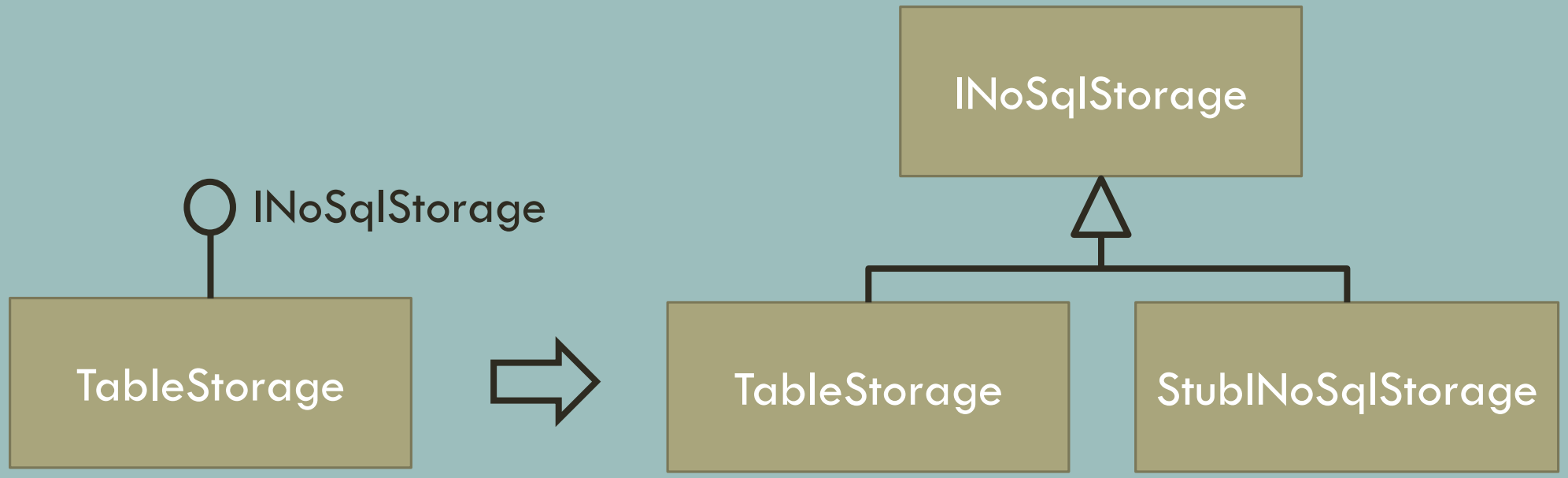
FRAGE

Durch was ersetze ich sowas wie TableStorage und Database?

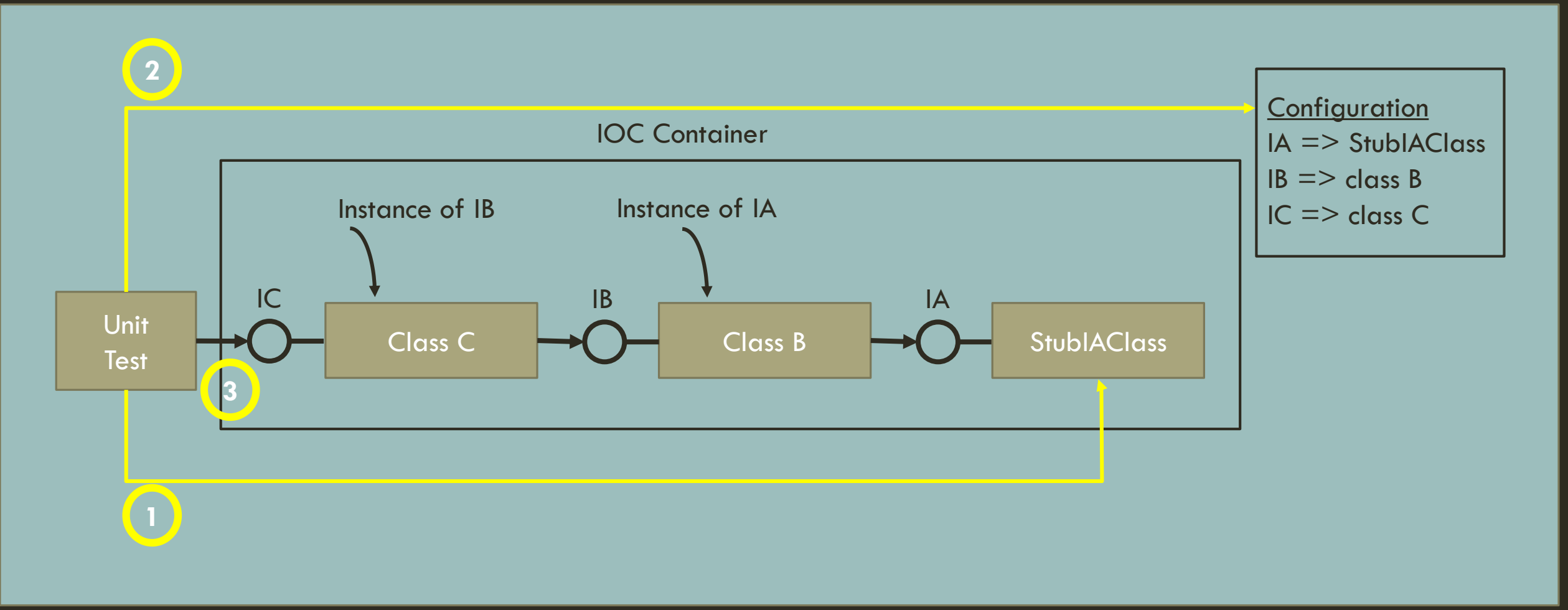
ANTWORT

Stubs

STUBS


INoSqlStorage
TableStorage
INoSqlStorage
TableStorage
StubINoSqlStorage

IOC CONTAINER



STUBS – HINTER DER KULISSEN

```
public class StubINoSqlStorage : INoSqlStorage{
    public Func<CommandStoreEnrty,bool> InsertOrReplaceFunc {get;set;}
    public bool InsertOrReplace(CommandStoreEntry commandStoreEntry){
        return InsertOrReplaceFunc(commandStoreEntry);
    }
}
```

Stubs im Test

```
var noSqlStorage = new StubITableStorage();
noSqlStorage.InsertOrReplaceFunc = c => return 200;
var sut = new ServiceController((INoSqlStorage)noSqlStorage);
var result = sut.AddGame(…);
Assert.isTrue(result);
```

STAND BEIM TESTEN…

Wir sind jetzt in der Lage einzelne Komponenten bis hin zum gesamten Systems (ohne externe Komponenten) automatisiert zu testen.

=> Puh, schon cool!

SCHRITT 4

Stubs einfügen.

FRAGE

Muss ich jetzt für jedes Interface die ganzen Stubs selber schreiben?

ANTWORT

Nein, auch da gibt es was von Microsoft… Fakes

WAS IST FAKES

Fakes ist ein Generator von Stubs und Shimes, gesteuert durch eine Konfiguration…

Aktueller Nachteil:

Nur verfügbar in Visual Studio Ultimate/Premium

FAKES CONFIGURATION

```
<Fakes xmlns="http://schemas.microsoft.com/fakes/2011/" Diagnostic="true">
  <Assembly Name="BRuKEware.ETS.WebRole"/>
  <StubGeneration>
    <Clear/>
    <Add FullName="BRuKEware.ETS.WebRole.BusinessLogic.INoSqlStorage"/>
    <Add FullName="BRuKEware.ETS.WebRole.BusinessLogic.IRepository"/>
  </StubGeneration>
  <ShimGeneration Disable="true">
  </ShimGeneration>
  <Compilation DisableCodeContracts="true">
    <Property Name="PlatformTarget">anycpu</Property>
  </Compilation>
</Fakes>
```

FRAGE

Wie testen wir jetzt aber die Daten aus der Datenbank ?

ANTWORT

SQL Server als LocalDB

WAS IST LocalDB

LokalDB ist der Nachfolger von SQL Express. Eine SQL Server DB, die nicht im Netzwerk auftaucht, als Datei vorliegt und für Entwickler gedacht ist.

DAS PROBLEM MIT DER TEST DATENBANK

Jeder Entwickler hätte gerne eine Test Datenbank mit diversen Daten Konstellationen, gegen den er seine Tests ablaufen lassen kann….

=> In der Realität gibt es diese aber meinst nicht. Aber….

LocalDB

Als eine als File vorliegende SQL Server DB, muss diese nur irgendwo hin kopiert werden. Diese Location muss dann nur im Connection String angegeben werden und dann kann diese benutzt werden. Es ist keine expliziter Attach DB notwendig.

=> Create => Copy => Configure => Use

FRAGE

Aber wir macht man das mit dem Connection String?

ANTWORT

Hole den Connection String nicht aus App-/Web-/Role-Config direkt sondern abstrahiere den Zugriff => Erzeuge eine ISettings Interface.

SCHRITT 5

Konfiguration abstrahieren.

FRAGE

Wie macht man das mit dem Connection String im Kontext des Entity Frameworks, da dieser per default aus der Web-/App-Config kommt ?

ANTWORT

T4 Code Generierung des EF erweitern und den Connection String über eine Instanz des ISettings holen.

UNIT TESTS MIT LocalDB

- StubISettings erzeugen und konfigurieren
- Andere Stub's erzeugen und konfigurieren
- IOC Container erzeugen und konfigurieren
- LocalDB Datenbank kopieren und mit Testdaten befüllen
- SUT erzeugen über den IOC Container
- Test ausführen
- Asserts durchführen

STAND BEIM TESTEN…

Ich bin jetzt in der Lage das System inklusive Datenbank automatisch zu testen.

=> Wow, das hat schon was.

ZUSAMMENFASSUNG DER SCHRITTE

1. Identifiziere die benutzten externen Systeme.
2. Abstrahiere die externen Systeme führe für jedes ein Interface ein.
3. Einführung eines IOC Containers.
4. Stubs einfügen.
5. Konfiguration abstrahieren.
6. …

. . . MAN KANN ES NOCH WEITER TREIBEN

- Externe Systeme durch Self Hosted Systeme ersetzen und im Test Context hochfahren (z.B. HTTP Self Host, WCF Self Host, XML RPC Self Host)
- UI Tests (Coded UI Test)

FAZIT

Ein System tastbar zu machen ist gar nicht so schwer, wenn man einmal verstanden hat, wie es geht.

=> Der Aufwand lohnt sich um das zu erreichen…

GEFÜHLTE VERTEILUNG ÜBER DIE ZEIT…

Q & A

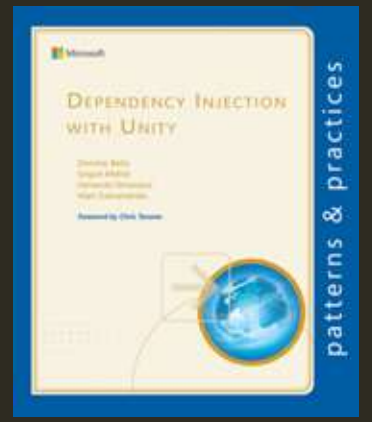
EIN PAAR LINKS

Dependency Injection with Unity

Dependency Injection for Web API Controllers

Better Unit Testing with Microsoft Fakes

SQL Server 2014 Express LocalDB



.NET OPEN SPACE SÜD KARLSRUHE 2014

19/20-JULI-2014
WWW.NOSSUED.DE

WER BIN ICH...

- Andreas Bräsen
- abraesen@bruke.de
- Dipl. Ing. der technischen Informatik (FH)
- Freiberuflicher Software Entwickler mit Schwerpunkt auf .NET basierter Software Entwicklung und Leidenschaftlicher Software Entwickler
- Twitter:@abraesen

www.bruke.de

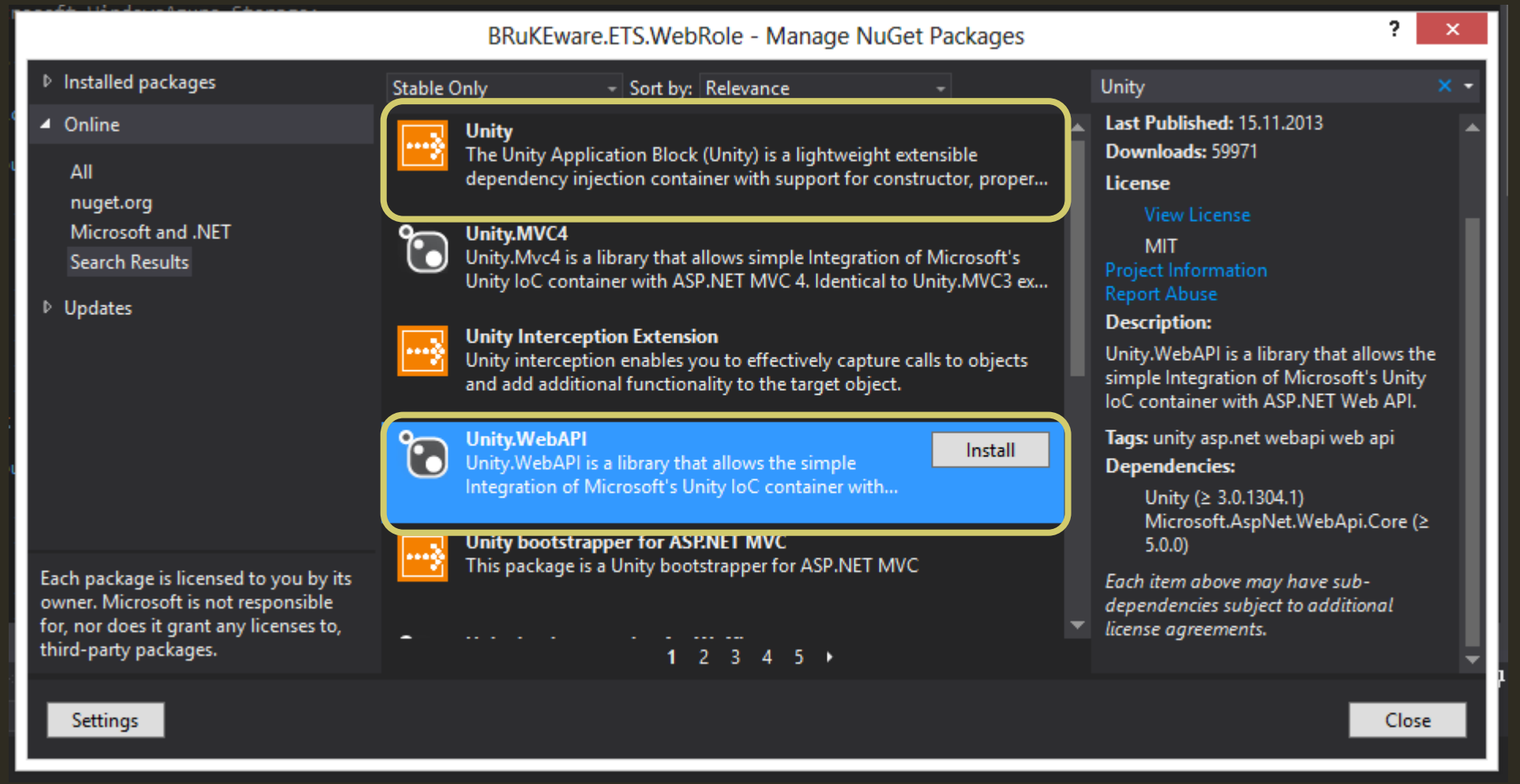
BEGRIFFSDEFINITION 1/2

- Dummy
  - Ein Objekt, das im Code weitergereicht, aber nicht verwendet wird. Werden eingesetzt um Parameter mit Werten zu befüllen.
- Fake
  - Ein Objekt mit Implementierung. Die Implementierung ist dabei jedoch eingeschränkt, wodurch ein Einsatz in der Produktionsumgebung nicht möglich ist. Ein typisches Beispiel für ein Fake ist eine Datenbank, die Daten nur temporär im Speicher hält.
- Stub
  - Ein Objekt, welches beim Aufruf einer bestimmten Methode unabhängig von der Eingabe die gleiche Ausgabe liefert.
- Mock
  - Ein Objekt, das bei vorher bestimmten Funktionsaufrufen mit bestimmten übergebenen Werten eine definierte Rückgabe liefert. Zur Erstellung des Mock-Objektes verwendet man üblicherweise ein Mocking Framework.
- Spy
  - Ein Objekt, welches Aufrufe und übergebene Werte protokolliert und bei Bedarf zurückliefert. Dabei werden Fake-, Stub- oder Mock-Objekte zu einem Spy erweitert. Alternativ kann ein Decorator eingesetzt werden.
- Shim, Shiv
  - Eine Bibliothek, welche die Anfrage an eine Programmierschnittstelle abfängt und selbst behandelt (z. B. mittels eines Fake-, Stub- oder Mock-Objekts), die übergebenen Parameter verändert oder die Anfrage umleitet.
- Quelle: http://de.wikipedia.org/wiki/Modultest

BEGRIFFSDEFINITION 2/2

- Komponententest[Bearbeiten]
  - Der Modultest, auch Komponententest oder Unittest genannt, ist ein Test auf der Ebene der einzelnen Module der Software. Testgegenstand ist die Funktionalität innerhalb einzelner abgrenzbarer Teile der Software (Module, Programme oder Unterprogramme, Units oder Klassen). Testziel dieser häufig durch den Softwareentwickler selbst durchgeführten Tests ist der Nachweis der technischen Lauffähigkeit und korrekter fachlicher (Teil-) Ergebnisse.
- Integrationstest[Bearbeiten]
  - Der Integrationstest bzw. Interaktionstest testet die Zusammenarbeit voneinander abhängiger Komponenten. Der Testschwerpunkt liegt auf den Schnittstellen der beteiligten Komponenten und soll korrekte Ergebnisse über komplette Abläufe hinweg nachweisen.
- Systemtest[Bearbeiten]
  - Der Systemtest ist die Teststufe, bei der das gesamte System gegen die gesamten Anforderungen (funktionale und nicht-funktionale Anforderungen) getestet wird. Gewöhnlich findet der Test auf einer Testumgebung statt und wird mit Testdaten durchgeführt. Die Testumgebung soll die Produktivumgebung des Kunden simulieren, d. h. ihr möglichst ähnlich sein. In der Regel wird der Systemtest durch die realisierende Organisation durchgeführt.
- Abnahmetest[Bearbeiten]
  - Ein Abnahmetest, Verfahrenstest, Akzeptanztest oder auch User Acceptance Test (UAT) ist das Testen der gelieferten Software durch den Kunden bzw. Auftraggeber. Der erfolgreiche Abschluss dieser Teststufe ist meist Voraussetzung für die rechtswirksame Übernahme der Software und deren Bezahlung. Dieser Test kann unter Umständen (z. B. bei neuen Anwendungen) bereits auf der Produktionsumgebung mit Kopien aus Echtdaten durchgeführt werden.
- Besonders für System- und Abnahmetests wird das Blackbox-Verfahren angewendet, d. h. der Test orientiert sich nicht am Code der Software, sondern nur am Verhalten der Software bei spezifizierten Situationen/Handlungen (Eingaben des Benutzers, Grenzwerte bei der Datenerfassung, etc.).
- Quelle: http://de.wikipedia.org/wiki/Softwaretest#Teststufen

1/3 - UNITY ZU ASP.NET MVC WEB API HINZUFÜGEN



Readme...

readme.txt | WorkerRole.cs | Your ASP.NET application | Source Control Explorer

```
Getting started with Unity.WebAPI
---------------------------------

To get started, just add a call to UnityConfig.RegisterComponents() in the Application_Start method of Global.asax.cs
and the Web API framework will then use the Unity.WebAPI DependencyResolver to resolve your components.

e.g.

public class WebApiApplication : System.Web.HttpApplication
{
  protected void Application_Start()
  {
    AreaRegistration.RegisterAllAreas();
    UnityConfig.RegisterComponents();                    // <----- Add this line
    GlobalConfiguration.Configure(WebApiConfig.Register);
    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);
    BundleConfig.RegisterBundles(BundleTable.Bundles);
  }
}

Add your Unity registrations in the RegisterComponents method of the UnityConfig class. All components that implement IDisposable
registered with the HierarchicalLifetimeManager to ensure that they are properly disposed at the end of the request.
```

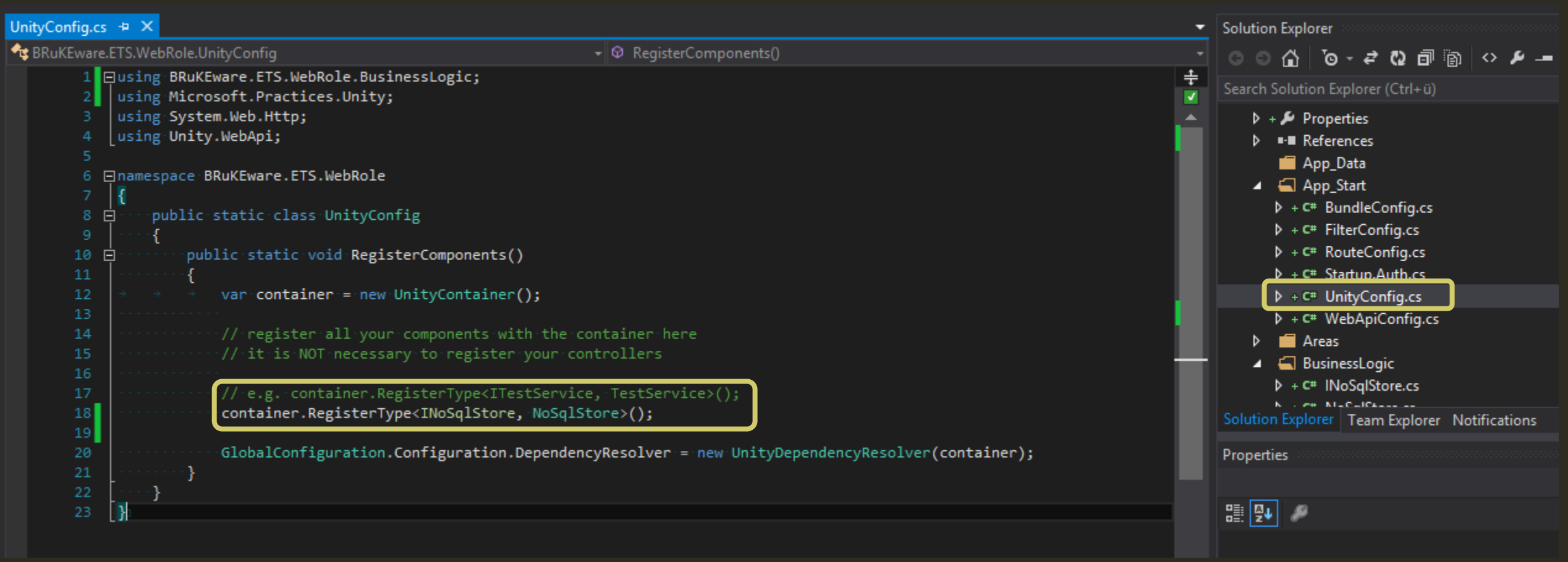
2/3- Global.asax.cs ANPASSEN

Global.asax.cs

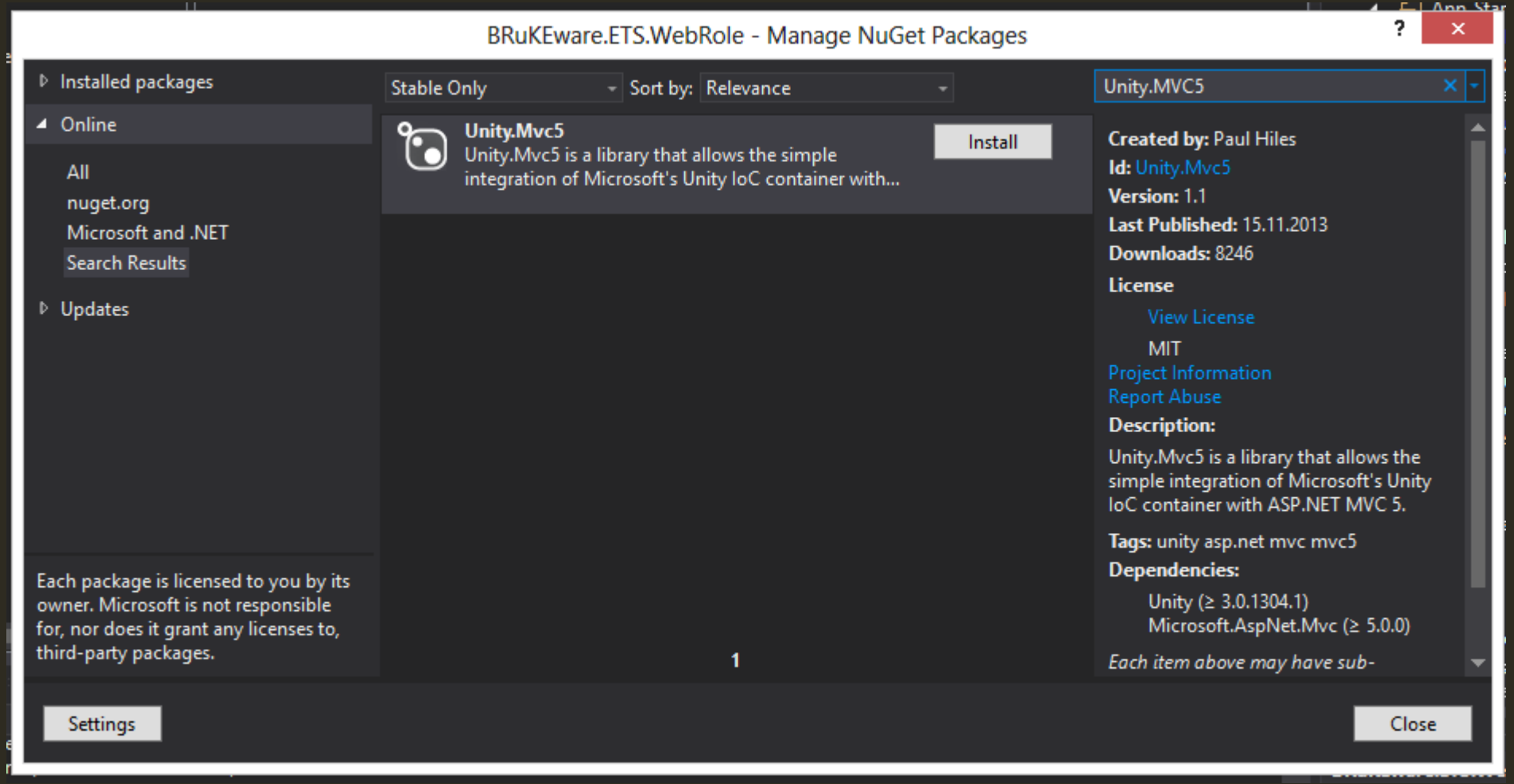
BRuKEware.ETS.WebRole.WebApiApplication — Application_Start()

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.Http;
6  using System.Web.Mvc;
7  using System.Web.Optimization;
8  using System.Web.Routing;
9
10 namespace BRuKEware.ETS.WebRole
11 {
12     public class WebApiApplication : System.Web.HttpApplication
13     {
14         protected void Application_Start()
15         {
16             AreaRegistration.RegisterAllAreas();
17             // Web API Unity registration.
18             UnityConfig.RegisterComponents();
19             GlobalConfiguration.Configure(WebApiConfig.Register);
20             FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
21             RouteConfig.RegisterRoutes(RouteTable.Routes);
22             BundleConfig.RegisterBundles(BundleTable.Bundles);
23         }
24     }
25 }
26
```

3/3 - UNITY IN IN DER SOLUTION



1/3 - UNITY ZU ASP.NET MVC 5 HINZUFÜGEN

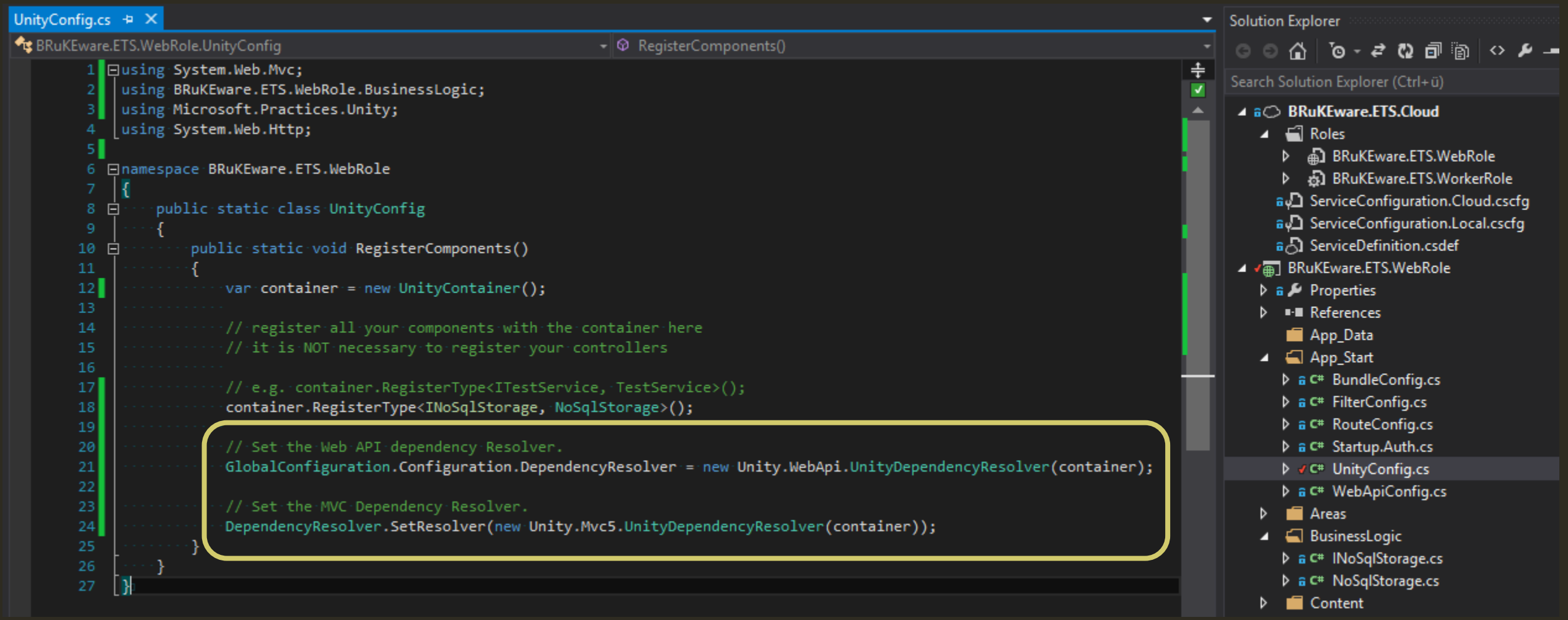


Readme...

readme.txt | WorkerRole.cs | Your ASP.NET application | Source Control Explorer

```
Getting started with Unity.WebAPI
---------------------------------

To get started, just add a call to UnityConfig.RegisterComponents() in the Application_Start method of Global.asax.cs
and the Web API framework will then use the Unity.WebAPI DependencyResolver to resolve your components.

e.g.

public class WebApiApplication : System.Web.HttpApplication
{
  protected void Application_Start()
  {
    AreaRegistration.RegisterAllAreas();
    UnityConfig.RegisterComponents();                    // <----- Add this line
    GlobalConfiguration.Configure(WebApiConfig.Register);
    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);
    BundleConfig.RegisterBundles(BundleTable.Bundles);
  }
}

Add your Unity registrations in the RegisterComponents method of the UnityConfig class. All components that implement IDisposable
registered with the HierarchicalLifetimeManager to ensure that they are properly disposed at the end of the request.
```

2/3 - UnityConfig.cs ANPASSEN



3/3 – WEB API HelpPage CONTROLLER ANPASSEN

```
public HelpController() : this(GlobalConfiguration.Configuration){}


protected HelpController(HttpConfiguration config)
{
   Configuration = config;
}
```