

Work with InterSystems.  
Not separate systems.



## The .NET Connection

Steve Pisani

INTERSYSTEMS

### Agenda



- The .NET Framework.
- Various ways to use Caché with .NET
- Caché Managed Provider for .NET
  - Overview
  - ADO.NET
  - Object Binding
- More...

INTERSYSTEMS

## What is the .NET Framework ?



### .NET Framework key features and benefits

- An extended object-oriented framework for many languages.
  - Applications can be developed using a mixture of languages
- A simpler deployment model
- Automatic memory management
- A standardized way to deal with software version conflicts
- A richer code security model
- A simple way to organize system and user code in namespaces
- A “near-seamless” interoperability with COM components
- A disconnected architecture for data access via the Internet with ADO.NET.

INTERSYSTEMS

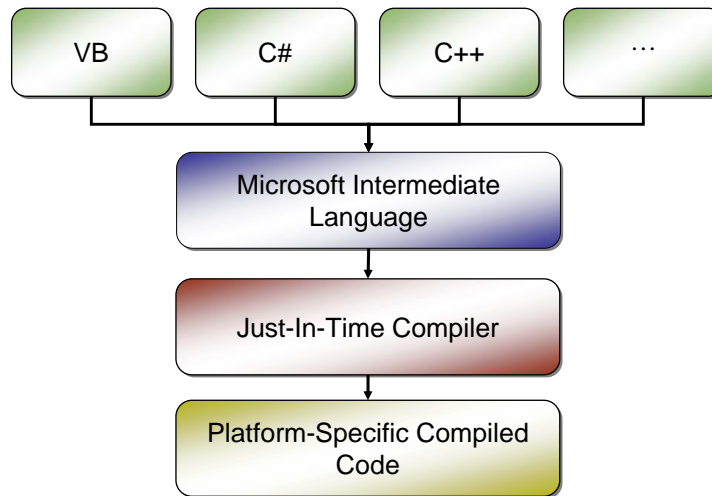
## .NET Framework



- The .NET Framework has two main components:
  - the .NET Framework class library.
  - the Common Language Runtime and
- The Common Language Runtime (CLR) is the foundation of the .NET Framework.

INTERSYSTEMS

## Common Language Runtime (CLR)



INTERSYSTEMS

## Language Independent Platform



- A language compiler targets the .NET platform by translating source code to Intermediate Language
  - Similar to Java byte code.
- At run time, the Common Language Runtime loads and executes the .NET application.
- Just-In-Time compilation is used to translate the Intermediate Language to native machine code.
- Code management by the Common Language Runtime provides an extra layer that decouples the application from the operating system.

INTERSYSTEMS

## Developing for the .Net Platform



- There are .NET compilers available for a variety of languages:
  - C#, Visual Basic .NET, JScript .NET, COBOL, Perl, Python, Eiffel, APL, and others.
  - You can also use the managed extensions for Visual C++ to write .NET applications.
- .NET supports these languages by supporting none directly. Instead, .NET understands only one language, Microsoft Intermediate Language.

INTERSYSTEMS

## Managed Code vs. Unmanaged Code



### Managed Code

- Runs within the .NET CLR
- Benefits from new .NET features, such as the CLR's new garbage collection, code versioning, and security policies
- "Platform-independent" runtime

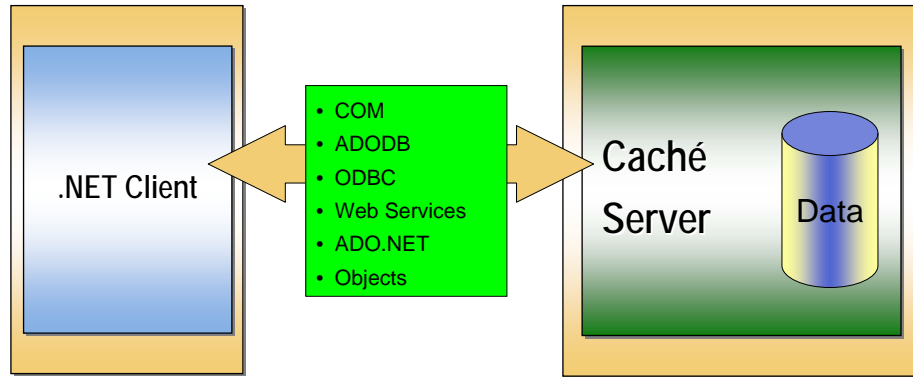
### Unmanaged Code

- Pre-compiled for a specific architecture
- Only runs on the intended platform
- More prone to memory leaks, version mismatch, and other issues

**Managed and unmanaged code can interoperate in .NET.**

INTERSYSTEMS

## Using Cache with .NET



INTERSYSTEMS

## COM: Run-time Callable Wrappers (RCW)



The RCW translates between .NET and COM so that:

- The COM object looks like a native .NET object
- The .NET client looks like a standard COM client
- Data types look the same

INTERSYSTEMS

## Caché support for COM under .NET



### Three flavors of COM support

- Caché Factory (CacheObject.DLL)
- Caché Direct (VisM.OCX)
- Caché Activate

INTERSYSTEMS

## ODBC



### ADO 2.7 is still supported by .NET and by Caché:

- ODBC (Microsoft ODBC DataProvider)
- Mostly For Legacy Applications

INTERSYSTEMS

## Caché Managed Provider for .NET



- What is *ADO*.NET ?
- ADO.NET architecture
- Caché Managed Provider for .NET
  - ADO.NET interface
  - OBJECTS interface

INTERSYSTEMS

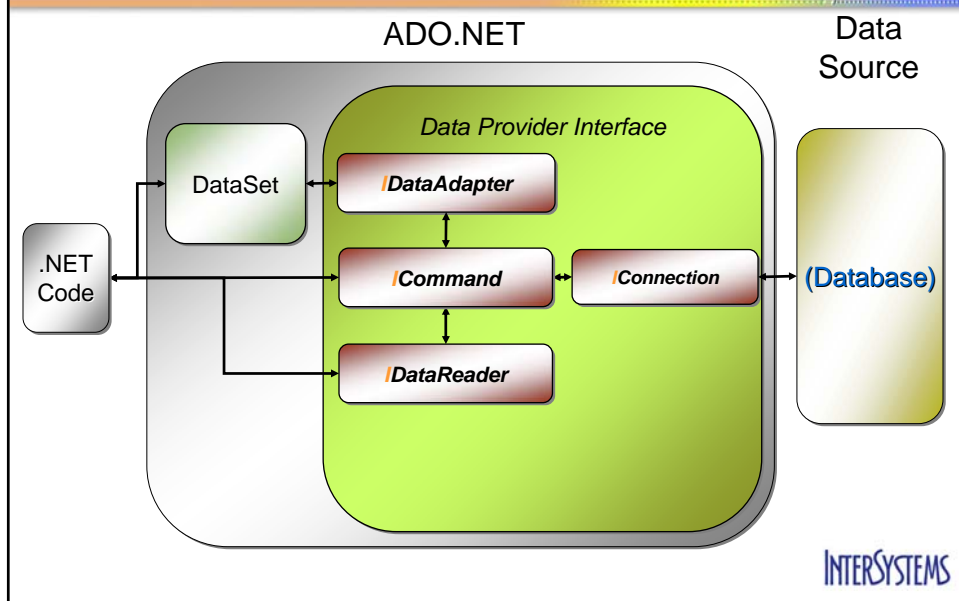
## What is ADO.NET ?



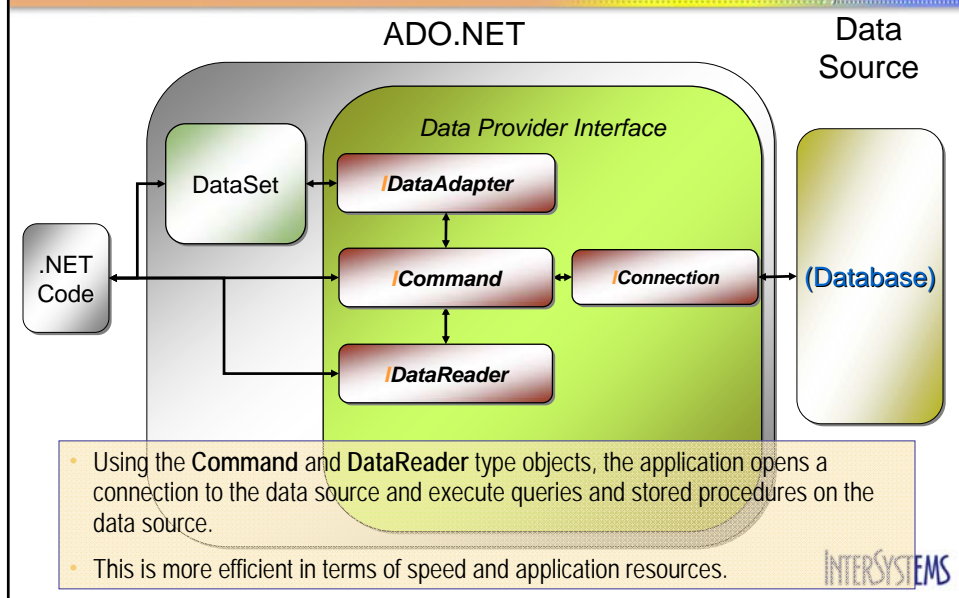
- The .NET Framework's set of classes for accessing and manipulating data
- Provides a link between a .NET programmer and a data store
- .NET applications can be somewhat database-independent
- ADO.NET provides two different styles of data access
  - Direct
  - Disconnected

INTERSYSTEMS

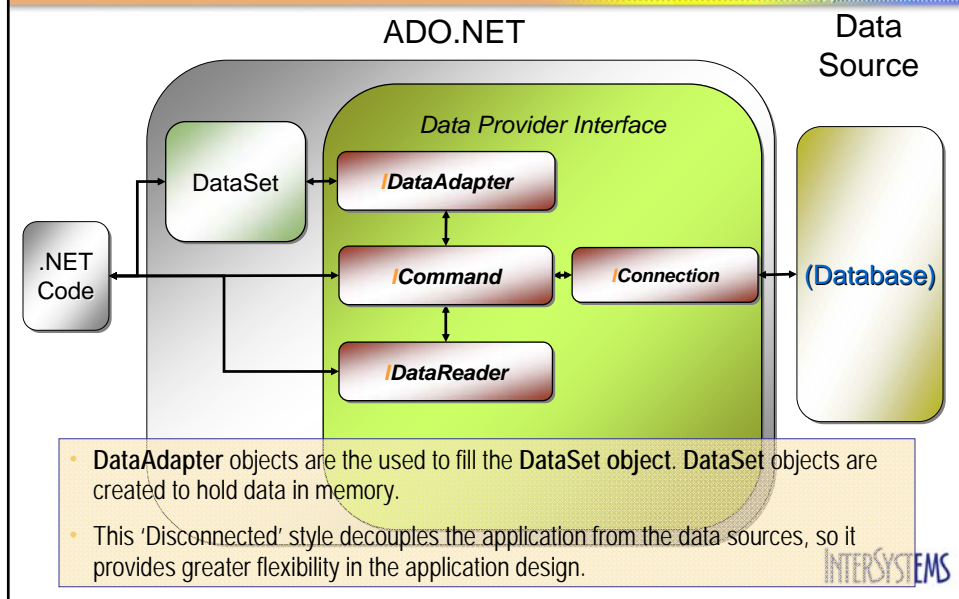
## ADO.NET Architecture



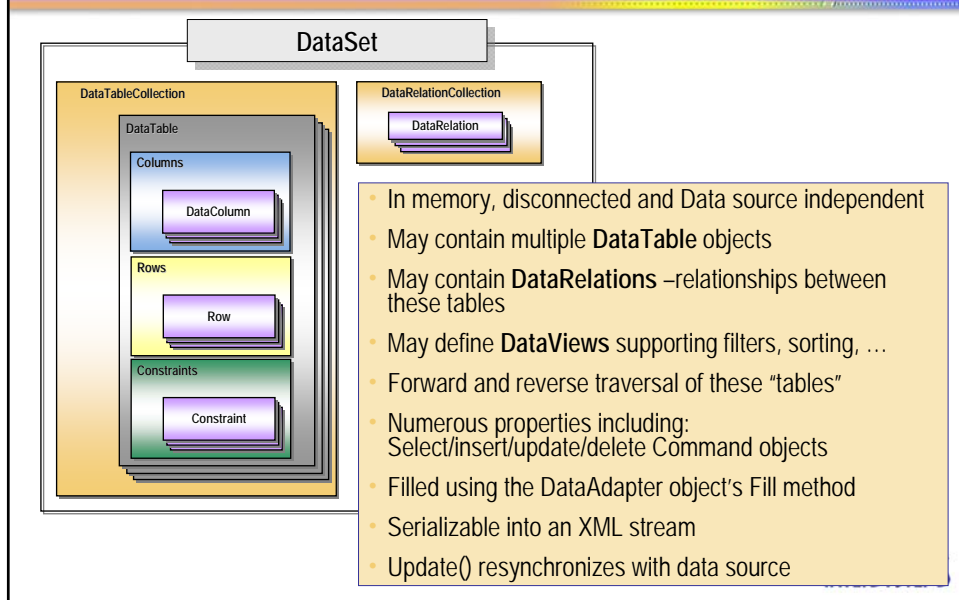
## Using ADO.NET – Direct DB Access



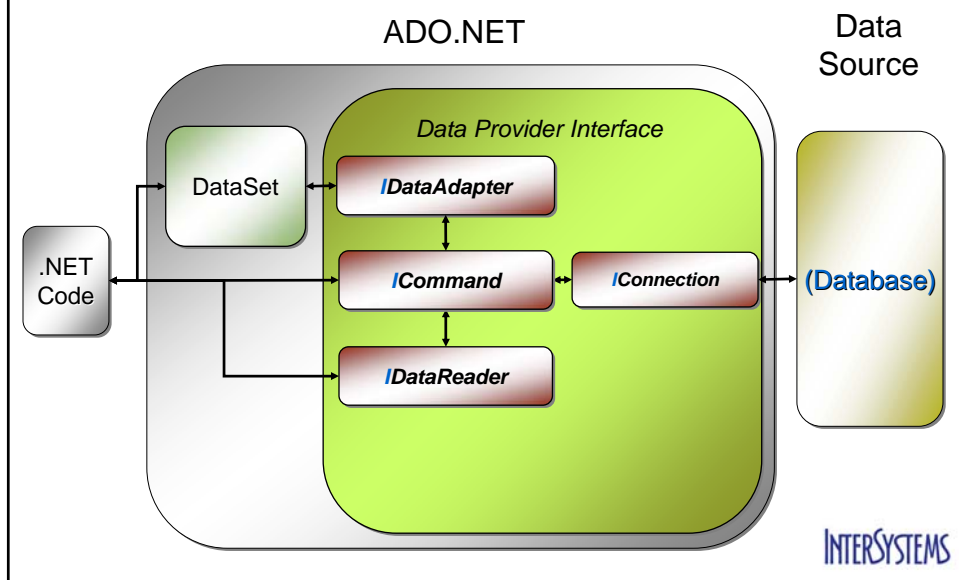
## Using ADO.NET – Direct DB Access



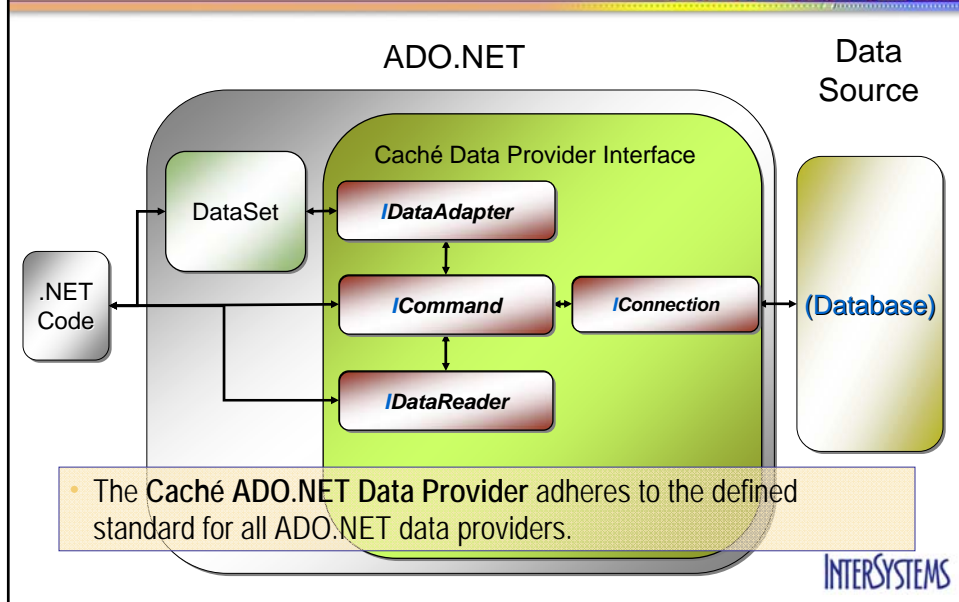
## Using ADO.NET: DataSets in more detail.



## Using ADO.NET with Caché



## Using ADO.NET with Caché



## CacheCommand



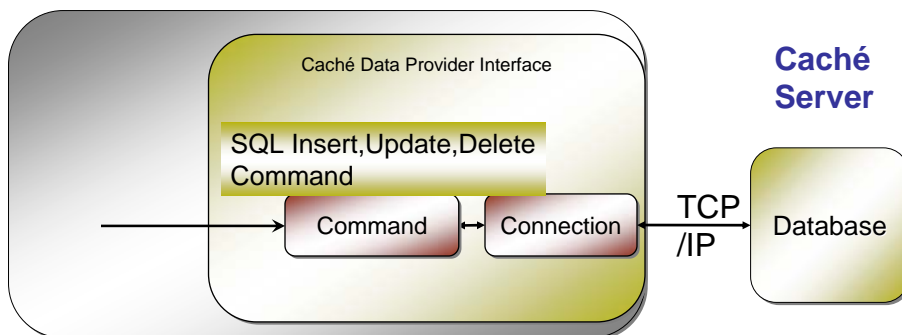
- Represents an SQL statement or stored procedure to execute against a data source
- Main Methods:
  - CreateParameter
  - ExecuteNonQuery
  - ExecuteReader
  - ExecuteScalar: Executes the query, and returns the first column of the first row in the resultset returned by the query.
  - Prepare

INTERSYSTEMS

## CacheCommand



.NET Project



(connect)  
ExecuteCommand  
(disconnect)

INTERSYSTEMS

## CacheDataReader



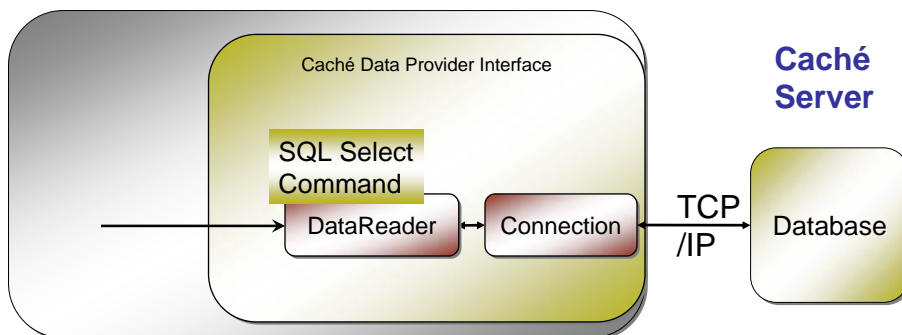
- Reads a forward-only stream of data rows from a data source.
- Main methods:
  - GetName, GetOrdinal, GetValue, Read
- While the CacheDataReader is in use, no other operations can be performed on the associated CacheConnection other than closing.

INTERSYSTEMS

## CacheDataReader



.NET Project



(connect)  
While datareader.Read() {process record}  
(disconnect)

INTERSYSTEMS

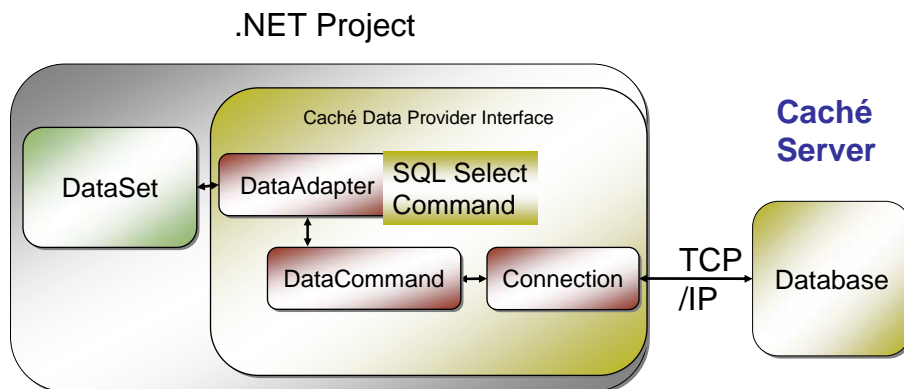
## CacheDataAdapter



- Allows full random read/write access to data.
- A set of data commands and a connection to a data source that are used to fill the DataSet and update the data source
- The CacheDataAdapter serves as a bridge between a DataSet and data source for retrieving and saving data.

INTERSYSTEMS

## CacheDataAdapter



Fill DataSet, (disconnect), Modify, (connect), Update()

INTERSYSTEMS

## Caché ADO.NET classes interface



- Demo ... *InterSystems.Cache.DataClient.dll*
  - Minimal ADO.NET project
  - Using CacheConnection, CacheCommand, CacheReader....
  - CacheDataAdapter

INTERSYSTEMS

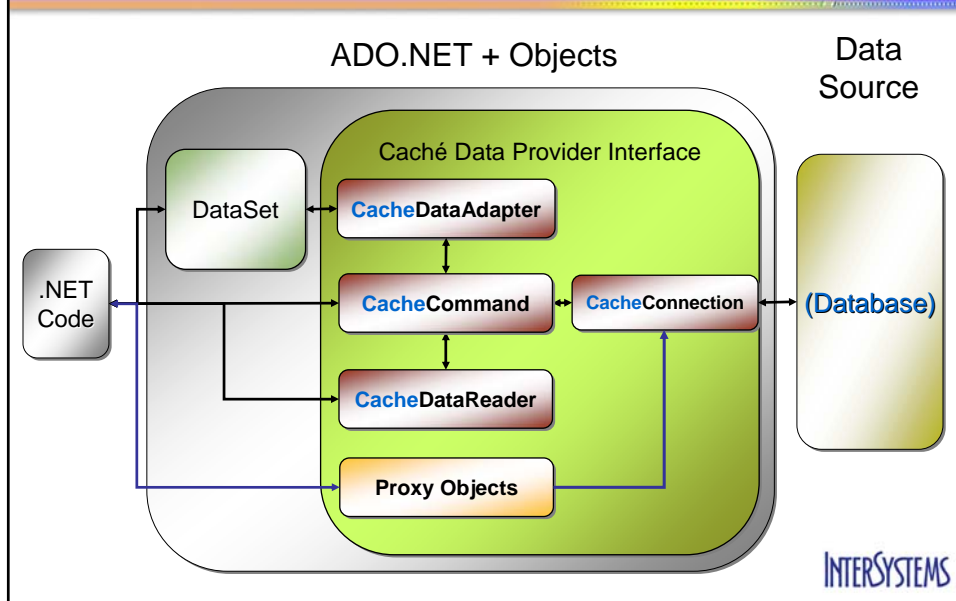
## Caché Managed Provider - Object Interface



- Implemented using Caché proxy objects
- Instances of .NET classes are generated from classes defined in Caché
- Each proxy object communicates with a corresponding object on the Caché server
- These proxy classes use a set of helper classes, contained in the *InterSystems.Data.CacheClient.dll* assembly

INTERSYSTEMS

## Caché Managed Provider - Object Interface



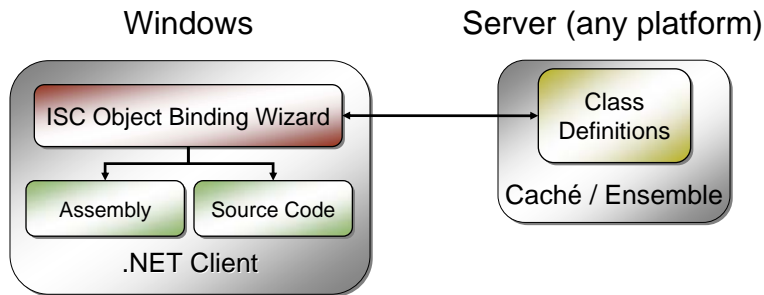
## Caché .NET Binding Workflow



- 1** • Define classes within Caché.
  - Either persistent objects stored in Caché or transient objects running in a Caché server.
- 2** • Use the **Caché Object Binding Wizard** to create .NET classes for the Caché classes
  - .NET Classes contain stub methods and properties corresponding to Caché object methods and properties.
- 3** • At Design time, included the generated .NET classes into the .NET project.
  - These 'Proxy' objects are the same as any other .NET objects, and are used in the same way too.
- 4** • At runtime, the .NET application connects to a Caché server and creates instances of .NET objects that correspond to objects within the Caché server.
  - Caché manages all communications and client-side data caching.

INTERSYSTEMS

## Generating .NET Classes



### Using the Object Binding Wizard

- Can be executed standalone or as a Visual Studio External Tool
- Form text can be easily localized
- Generated code is used directly in .NET project

INTERSYSTEMS

## Assemblies vs. Source Code



### Assemblies

- Usable by any .NET code in any source language
- Code is compiled once, not repeatedly with each project build
- Caché code is neatly separated from user-defined code

### Source Code

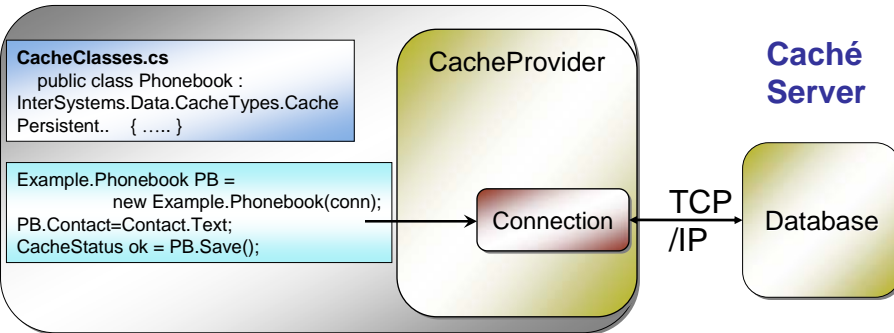
- Generated code can be stepped through in the Visual Studio debugger
- Generated code can be sub-classed

INTERSYSTEMS

## Object Binding



### .NET Project



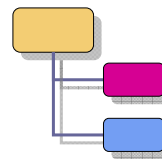
Add wizard-generated classes  
obj.Open(), obj.Name="x", obj.Save()

INTERSYSTEMS

## Proxy class methods



- **Id**
  - Returns the Caché object's object ID as type **string**.
- **ExistsId**
  - Returns a **boolean** value.
- **OpenId**
  - Opens the Caché object with the specified object ID on the Caché server and opens a corresponding .NET proxy object on the .NET client.
- **DeletId**
  - Deletes the Caché instance with the specified object ID from the Caché server.
- **Save**
  - Saves the state of the Caché object to the database. The method returns a **CacheStatus** object that contains information about the success or failure of the operation.

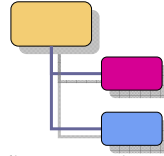


INTERSYSTEMS

## Projections from Caché classes.



- **Methods** - proxy class contains both the class and instance methods of the corresponding Caché class. Executing the proxy method causes the corresponding Caché method to execute.
- **Properties** - proxy class has a property corresponding to each public property in the corresponding Caché class.
- **Collection Properties** - proxy class contains projections of the collections in the corresponding Caché class.
- **Relationships** - proxy class contains properties representing the properties of a Caché relationship.
- **Queries** - proxy class projects Caché class queries as ADO.NET CacheCommand objects.



INTERSYSTEMS

## Caché ADO.NET Object interface



- Demo

INTERSYSTEMS

## More...



- **Caché hosted Web Service can return .NET DataSet.**
  - Extend %XML.DataSet
- **Output/Input redirection**
  - Server side Write statements in Class methods can be 'captured' and sent to a nominated method in your project (then handled as required).
  - The same applies for Input re-direction.
- **The IgnoreNull flag:**
  - Setting a *class.property* = TextBox.Text, (where this textbox is empty), will stick a \$CHAR(0) in Cache once *class* is saved.  
Use this flag to implement the expected behavior and avoid having to translate null("") and empty strings (\$c(0)) when handling object properties.
- **Scripting the Cache .NET Wizard**
  - Manually generating proxy classes, using Native Adapter methods, or, the dotnet\_generator.exe tool.
- **More Resources:** See /dev/dotnet and E-Learning

INTERSYSTEMS

Work with InterSystems.  
Not separate systems.



## The .NET Connection

Steve Pisani

INTERSYSTEMS