Czech University of Life Science Prague

Faculty of Economics and Management

Department of Information Engineering

DIPLOMA THESIS

Object-oriented database in application programming

Author: Gofur Halmuratov

Supervisor: doc. Ing. Vojtěch Merunka, Ph.D.

DECLARATION

I declare that the Diploma Thesis „Object-oriented database in application programming" is my own work and all the sources I cited in it are listed in References.

Prague, 31.03.2014

Signature:

ACKNOWLEDGEMENTS

I take this opportunity to express my deep gratitude and regards to my supervisor *Associate Professor Ing. Vojtěch Merunka, Ph.D.* for the help in the preparation of my thesis. His serious and rigorous scientific approach mastering science deeply infected and cheered me. From the choice of thesis topic to the final completion of the thesis, it always gave me a sensitive leadership and tireless support. My supervisor gave me great help in the research, not only helped me to correct errors, but also gave me valuable advice. I would like to take this opportunity to express my heartfelt gratitude and deepest respect.

I thank almighty, my father *Soriboy Halmuratov*, my mother *Rahima Kuramboeva*, my wife *Ro'za*, brothers and sisters for their great support and help without which this assignment would not be possible.

My sincere appreciation and gratitude are for all of them and many others.

SUMMARY

The goal of this research is to implement an application which uses one of Object-oriented database as database. For this purpose, this research mainly focused to study InterSystems Caché. Caché is a high-performance object-oriented database management system nowadays. It uses newer approach towards to data modeling. Data is stored as objects, interfaces of objects are created as classes in object-oriented database. It supports all paradigms of object oriented programming such as encapsulation, inheritance and polymorphism. I have chosen a kind of library called *czuLibrary.NET* application as an example, but it is intended only for small scientific group. It is a good choice to show an example of implementation of InterSystems Caché as database and evaluate it from the perspective of differences between pure object and relational technology.

My summaries about InterSystems Caché are:

- Easy to use
- Faster database as no joins required
- complex data types
- High performance can be achieved
- Object extensions for SQL script to handle object identity and relationships
- Multiple inheritance
- supports for object relationships

Object-oriented approach models the real world more completely than a relational approach. Complex objects are inefficient to model in a RDBMS. When application behavior is separated from database behavior, mapping between the relational and the object world has to be made.

Keywords: OODBMS, DBMS, RDBMS, SQL, Intersystems Caché, Caché Object script, Object-oriented database, XML, Object-oriented programming, relational databases, UML, class diagram, Windows presentation framework

Table of Contents

1. Introduction

An object-oriented database or an object database is a database in which data is represented as objects used in object oriented programming (OOP). Each object consists of attributes and methods. Object databases are different from Relational databases management systems (RDBMS) because of data representation.

At the beginning of the design process of new software, developers must decide their approach to data modeling. For most, this means a choice between the traditional relational data model and the new modeling approach as objects. Faced with the need to manage complex data, many developers believe that data modeling with objects is a more efficient approach. Because the data are represented as objects used in object-oriented programming as relational databases that are focused on the table.

Object-oriented database management systems (OODBMS) have features of OOP language which are abstract data types, encapsulation, inheritance and polymorphism. OODBMS let OOP programmers to develop project, store them represented as objects, and replicate, edit existing objects or create new objects in the OODBMS. Because Object database is integrated with the programming language, the developer can maintain consistency within the environment, in that both the object-oriented database management system and the programming language uses the same representation model.

Object-oriented approach models the real world more completely than a relational approach. Complex objects are inefficient to model in a RDBMS. When application behavior is separated from database behavior, mapping between the relational and the object world has to be made.

In this research, I have focused to study one of object-oriented database which is called InterSystems Caché. It has features of SQL and object access to the database, as well as letting direct manipulation of InterSystems Caché's underlying data structures.

InterSystems claims that InterSystems Caché is the fastest object database of the world.

2. Aims of the thesis

The Aim of this research is to implement an example object database in the environment of the Caché object-oriented database and evaluate it from the perspective of differences between pure object and relational. Documenting analysis and design and estimate the benefit of new technology on software quality and complexity

For this purpose, I have chosen *czuLibrary.NET* application to implement as an example. It is a kind of library, but it is intended only to small scientific group. It has features of managing Items, Academic Staff, Authors, Categories, and Publishers as well as borrowing Items and Recording information about borrowing. This application is only for studying and evaluating how InterSystems Caché works.

czuLibrary.NET is a good choice, because I can use most features of object-oriented databases such as encapsulation, inheritance, polymorphism and accessing databases as objects and SQL during designing and implementing of the *czuLibrary.NET* application.

3. Research Methods

During doing this research and developing of the *czuLibrary.NET* application, I have tried to follow agile methodology principles as much as possible and also I have used following methods.

1. *UML Class diagramming* is for modeling our *czuLibrary.NET* application. It helped me to overview object classes.
2. *Object-oriented programming* is for accessing Caché's database objects and managing data.
3. *SVN repository* is for versioning source code and revision control.
4. *Software engineering techniques of software quality and complexity estimation* are used for evaluation of database performance and workload of developers.

I. LITERATURE REVIEW

4. Introduction to Object-oriented databases

„An object-oriented database management system (OODBMS) is a database management system that supports the creation and modeling of data as objects. OODBMS also includes support for classes of objects and the inheritance of class properties, and incorporates methods, subclasses and their objects. Most of the object databases also offer some kind of query language, permitting objects to be found through a declarative programming approach."(Techopedia, 2014)

Currently, the most popular approach to organizing information systems - is a client - server applications, which are based on the interaction with the database. The majority of data storage used relational databases, and applications that provide an interface to a database implemented on the basis of object-oriented programming. In this case there is a discrepancy related to the fact that the data model used in programming models differ from DBMS, which entails the need to maintain relations between tuples and harmonies database objects and programming. The need for such approvals, or impedance, was accepted as payment for the performance provided by the relational data organization. But if we can find another way of developing the system in which such payment will be avoided.

Object-oriented databases are the result of combining the principles of object -oriented programming principles and database management. In one system, combined the concepts of encapsulation, polymorphism, inheritance of object-oriented programming and the atomicity, consistency, isolation from databases. As a result, we are able to manage large amounts of information using an object- oriented approach.

Object-oriented database (OODB) is capable of storing objects in the same form in which they will be available for the programming language. This is ensured by the fact that the objects in OODBs belong to the class, having in its structure a set of attributes that can be expressed by simple data types or other classes. The classes, the rules of inheritance that carry all the benefits of relying polymorphism, overriding inherited methods and the possibility of dynamic binding.

Each class object is an object identifier that is used to uniquely identify this object in the system. Identifier assigned by the system and does not depend on the state of the object.

In the object model, storing references to other objects looks comfortable enough, but there may be other problems associated with referential integrity. For example, when an object is deleted, other objects may be linked to the identifier. Because the control system of object-oriented databases (OODBMS) should represent not only the system -oriented software development environment, but also a data management system. OODBMS shares many features characteristic of relational databases, such as transaction codes, identification and resolution of deadlocks, data recovery mechanisms

Objects, in which most modern client/server applications as well as other complex software are programmed, do not fit perfectly into relational tables. More often than not, the only solution is to create a separate table for each distinct class that exists in the particular database model being developed. Once the number of classes reaches a point where data needs to be taken from many tables to complete most queries, the overhead will be increasingly noticeable. The SQL method of pooling together data from more than one table is by using the JOIN operation, an intensive algebraic computation when working with large data.

To overcome the performance limitations of traditional relational data bases applications - from those running on a single machine to large interconnected networks - often use object-oriented databases to speed up access to data. Although the basics of object-oriented and product data caching increase throughput, they suffer from a number of limitations, including the lack of support for large data sets, excessive hardware requirements, and limitations scalability.

Objects consist primarily of the following:

1. *Attributes* - Attributes are data that defines the characteristics of an object. These data can be from very simply data type to a reference to a complex object as classes.
2. *Methods* - Methods specify the behaviour of an object, they are mostly called procedures or functions.

Therefore database objects contain data and executable code. There are some other properties of the objects, for example, whether the data from the outside of the object or process can be used.

5. Comparison of object-oriented database and relational database

„Relational databases still make sense for many kinds of applications that lend themselves to expression as simple sets of simple tables. However, to build applications that reflect the real world, you will have to find a way to take another two steps forward this time to a post relational world as object-oriented database."(Kirsten, 2003)

For comparison, relational and object -oriented databases is necessary to analyze their features. The need to address the shortcomings of hierarchical and network models have led to the emergence of a new, relational data model, which was developed by Edgar Frank Codd (Codd) in 1970. It lacked clear pointers to ancestors and descendants, and all data were presented as simple tables, divided into rows and columns. The practical definition of the term „relational database" was much vaguer than precise mathematical definition given to this term by Codd in 1970.

In the first relational DBMS were not implemented some of the key parts of the model, and this gap was filled only later.

In response to the misuse of the term „relational" Codd in 1985 wrote an article in which he formulated 12 rules that must be satisfied by any database that pretend to be relational . Since then, twelve rules considered Codd defined the relational DBMS.

However, we can formulate a simpler definition: called relational database in which all the data available to the user, are organized in the form of tables, and all operations on the data are reduced to operations on these tables.

Now, consider Object-Oriented database. Object- oriented databases are used from late 1980s to provide database management, applications built in accordance with the concept of object-oriented programming. Object technology extends the traditional application development methodology with new data modeling and programming techniques. For code, reuse and improve data integrity in object programming code and data processing are organized into objects. Thus, virtually no limit to the types of data.

If the data consists of a short, simple fixed-length fields (name, address, bank account balance), then the best solution is to use a relational database.

If, however, the data contains a nested structure, dynamically resizable, user-defined arbitrary structure (multimedia, for example), their representation in tabular form is at least difficult. At the same time in OODBMS each user-defined structure - an object directly controlled database.

In relational database connection controlled by the user by creating a foreign key. Then, to find relations dynamically at runtime system looks at two (or more) tables comparing foreign keys to achieve compliance.

This process, called the union (join), is the weakness of relational technology. If more than two or three levels of associations - a signal to look for a better solution. In OODBMS user simply declares the linkage, and automatically generates a database management techniques, dynamically creating, deleting and crossing links. Direct links with no need for viewing and comparison, or even search index, which can greatly affect performance. Thus, the application object model is preferable for databases with a large number of complex relationships: a cross-reference, the links between multiple objects with several (many-to-many relationships) bidirectional links.

Unlike relational, OODBMS fully supports object - oriented programming languages.

Developers using C++, .NET or Smalltalk, deal with one set of rules (allowing the use of the advantages of object technology as inheritance, encapsulation and polymorphism).

Developer does not need to resort to a broadcast object model into a relational and back. Application programs access and operate with objects stored in the database that uses the standard object-oriented language semantics and operations.

In contrast, relational database requires developers to broadcast to the object model supported by the data model and included routines to provide this mapping at runtime. Result in an additional effort in the development and reduction of efficiency.

Despite the enormous popularity of the paradigm of object- oriented programming technology in the development of databases that paradigm while not particularly popular. And there are both objective and subjective reasons.

Based on relational databases were created many great products that must be maintained and developed. These products have already invested a lot of money and customers more willing to invest in their development. In contrast, using OODBs developed relatively few serious commercial products, there are few powerful OODBMS. One of them is Intersystems Caché.

Even back in 1986, the first standard SQL86 was introduced, which determined the fate of all relational databases. After the adoption of the standard relational database, all developers had to follow it. There is OQL for object -oriented database query language standard.

Relational database at the time Edgar Codd laid the foundations of the mathematical apparatus of relational algebra. This explains the mathematical apparatus, should be performed as basic operations on relations in a database proves their optimality (or out of it to see where it is necessary to optimize). On the other hand for OODBs is no such system, even though that work in this area have been conducted since the 80s. Thus, in OODBs is no strict terms such as Cartesian product, attitude, etc.

In relational database stores only the bare details. That they will make an application depends upon the application. In OODB, in contrast items should be kept, and the object is a collection of its properties (parameters of the object) and methods (interface object). Here too there is no consensus as OODBs must carry out the storage objects and how these objects should the developer to develop and design. Here arises the problem of storing the object hierarchy, storage abstract classes, etc.

Thus, the active use of object -oriented database system requires further research and practical study.

As we mentioned, Relational databases store data in tables which are two-dimensional tables. Tables of relational databases are „normalized” so that data are not repeated more often than necessary. All columns in the table depend on the primary key (required unique value in the column) to determine the column. Once the specific column is determined, data from one or more associated with the column lines may be acquired or changed.

To put objects in relational databases, they must be represented in terms of data types such as string integer or real numeric. For example in the case of an aircraft. The wing can be placed in an array with rows and columns describing its size and characteristics. The body may be in another table, the helix in another table, the tires, and so on.

Breaking complex information in a simple data takes time and hard work. The code must be written to carry out this task. Data is stored on a permanent storage device in Relational databases.

The application can handle the transitional time and persistence in the object database. Object must be used where complex information and could be complex data relationships. This could include a many to many relationships. Object Databases should not be used when there would be something to join the tables.

“In relational databases, data is stored as attributes in tables linked with keys (primary and foreign keys), data is identified with unique keys assigned at their creation and remains during the whole lifetime regardless their states. One of the issues or questions about OODB is whether code should be stored in the database this question divided the community of OODB; many developers thought that using this approach affects significantly performances, while others argued that if code was not stored in DB a well-known semantic anomalies might happen.”(Norrie, 2009).

The weakness of this implementation is that the data access layer is tightly developed based on object-oriented design, if you change the definition of objects, database schema and access methods need to be changed too. Whereas, if we choose to store data in the form of objects in an OODB, seems to hide or circumvent this difficulty level.

Benefits of Relational database

1. SQL and standards
2. Easy to model
3. Can only use standard types and suppliers
4. Referential integrity (theory mathematically solid relational sets)
5. Many tools and implementations of databases
6. Data separate program
7. the high-end infrastructure support and storage management

Disadvantages of Relational database

1. No custom type
2. No extensible data types
3. Differential impedance
4. Cannot express the nested relationship
5. Do not use complex entities as a single unit such as classes
6. Need to define keys and various types of relationships such 1:1, 1:n, n:1, n:n in the data model
7. The writing procedures for versioning, operations if necessary

Benefits of object-oriented databases

1. High performance
2. Versioning mechanism inherent
3. The navigation interface for operations (such as graphic crossing)
4. Object Query Language retrieve declarative objects
5. Complex Data Types
6. Identity of the object i.e. equals () in which the identity of objects is independent of the value and updates
7. Integrated language persistence as ODL
8. Support for relationships
9. Facilitates the sharing of objects
10. Classes and hierarchies (inheritance and encapsulation)
11. Support for atomicity
12. Integrated language persistence as ODL

13. Support for nested relationships
14. Semantic Modelling

Disadvantages of object-oriented databases

1. Disadvantages of object orientation
2. The persistence difficult for complex structures, data must be transient
3. The object-oriented modelling data
4. Powerful query language
5. SQL types and rich extended
6. Support for legacy RTUs
7. Basics of object-relational data
8. Different approaches (OO, relational DB or OODB) may be required for different applications
9. Support for complex data types such as collecting, multi etc.

6. Object-Oriented Databases in InterSystems Caché

6.1. Introduction

„InterSystems Caché is an advanced database management system and rapid application development environment. With Caché, you'll make breakthroughs in processing and analyzing complex Big Data, and developing Web and mobile applications. Caché uniquely offers lightning-fast performance, massive scalability, and robust reliability – with minimal maintenance and hardware requirements."(InterSystems, 2014)

InterSystems Caché is an object-oriented database, which provides huge opportunities for developing Web-based solutions and client-server applications. Caché is designed for transaction processing systems with extra-large databases and a virtually huge number of concurrent users.

The unique quality of Caché is equitable and effective support of three ways to work with the data:

1. Powerful object access
2. Relational SQL- access
3. Flexible and high-performance direct access to multidimensional data

InterSystems Caché is a commercial object-oriented database management system (ODBMS). It has features of object and SQL access to the database, as well as allowing direct manipulation of Caché's underlying data structures. Caché runs on many operation systems such as Windows, Linux, Solaris, AIX, HP-UX, Tru64 UNIX, OpenVMS platforms and Mac OS X.

According to InterSystems Caché's web site InterSystems, *„Caché - a high-performance object database management system" and „world's fastest database"(InterSystems, 2014).*

"Caché has evaluation version also which is a fully functional, non-expiring, single-user version of the InterSystems Caché database. It includes all of the development and deployment capabilities of larger Caché implementations except the ability to network with other Caché

systems and to connect terminals, instruments, or similar external input devices. Although it is limited to a single user, the Caché evaluation version can run twelve concurrent processes."(InterSystems, 2014)

6.2. Installation Caché

Caché has an evaluation version, it is free to install on only one machine. There is an evaluation version of Caché on their website (http://www.intersystems.com/library/software-downloads/). We need to register first to get a free evaluation version.

6.3. The Caché Utilities

6.3.1. Caché Cube

Caché has a set of tools to manage and create databases. You can invoke management tools that control important Caché functions directly from the Caché Cube menu.

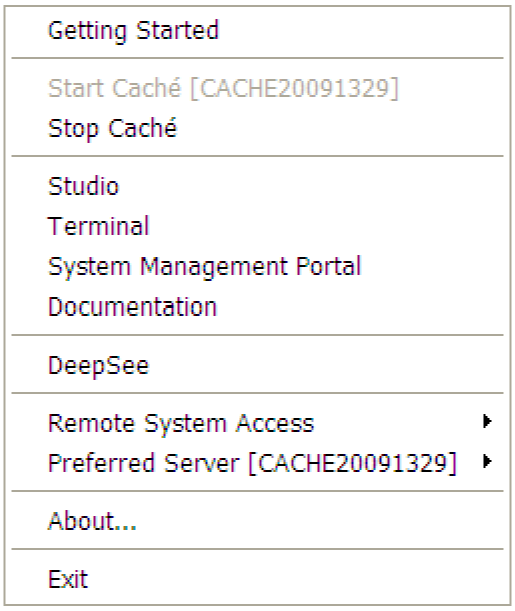


Figure 1. Caché Cube tray menu. [Source: InterSystems]

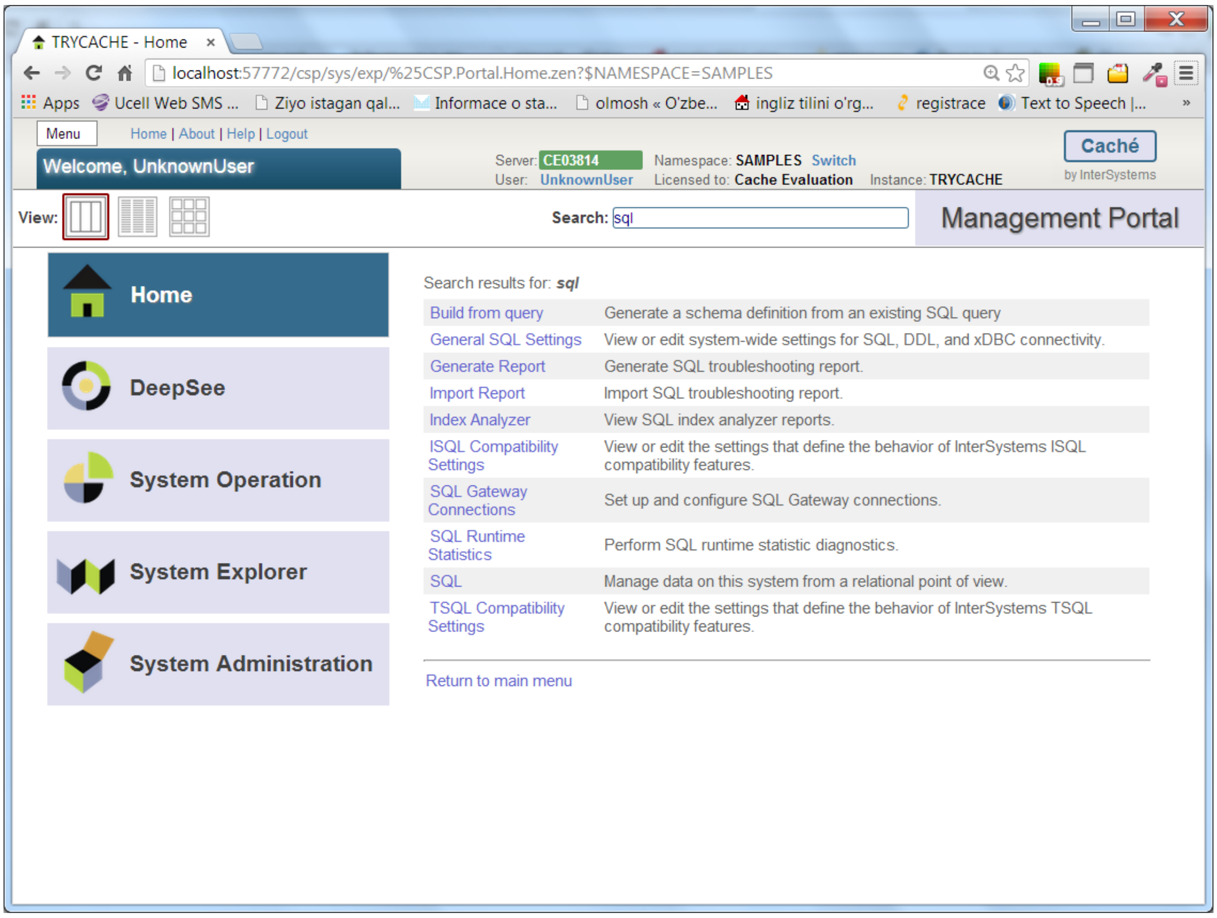
The following describes the Caché Cube menu commands.

1. *Getting Started* - shows links to documentations, release notes and other important information.
2. *Start Caché* - Starts Caché server.
3. *Stop Caché* - Shuts down or restarts Caché server.
4. *Caché Studio* – for managing Caché class definitions, CSP (Caché Server Pages) pages, Caché Basic routines, and Caché ObjectScript routines.
5. *Terminal* - Caché command line tool.
6. *System Management Portal* - Performs common system management tasks.
7. *Documentation* - opens Caché online documentation.
8. *Preferred Server[server name]* - lists remote servers.
9. *About* – shows information about Caché.
10. *Exit* - Removes tray icon.

„Caché ObjectScript is an object programming language designed for rapidly developing complex business applications. Caché ObjectScript source code is compiled into object code that executes within the Caché Virtual Machine. This object code is highly optimized for operations typically found within business applications, including string manipulations and database access. ObjectScript programs are completely portable across all platforms supported by Caché."(InterSystems, 2014)

6.3.2. System management portal

The System Management Portal is a browser-based interface that allows a variety can be made of the most commonly used settings of Caché. Figure 2 shows the home page of the portal. About this portal, for example, new Databases generated or ad hoc queries are made using SQL. The Management of backup or import data using CSV files are on the System Management Portal possible.



Search results for: *sql*

Build from query	Generate a schema definition from an existing SQL query
General SQL Settings	View or edit system-wide settings for SQL, DDL, and xDBC connectivity.
Generate Report	Generate SQL troubleshooting report.
Import Report	Import SQL troubleshooting report.
Index Analyzer	View SQL index analyzer reports.
ISQL Compatibility Settings	View or edit the settings that define the behavior of InterSystems ISQL compatibility features.
SQL Gateway Connections	Set up and configure SQL Gateway connections.
SQL Runtime Statistics	Perform SQL runtime statistic diagnostics.
SQL	Manage data on this system from a relational point of view.
TSQL Compatibility Settings	View or edit the settings that define the behavior of InterSystems TSQL compatibility features.

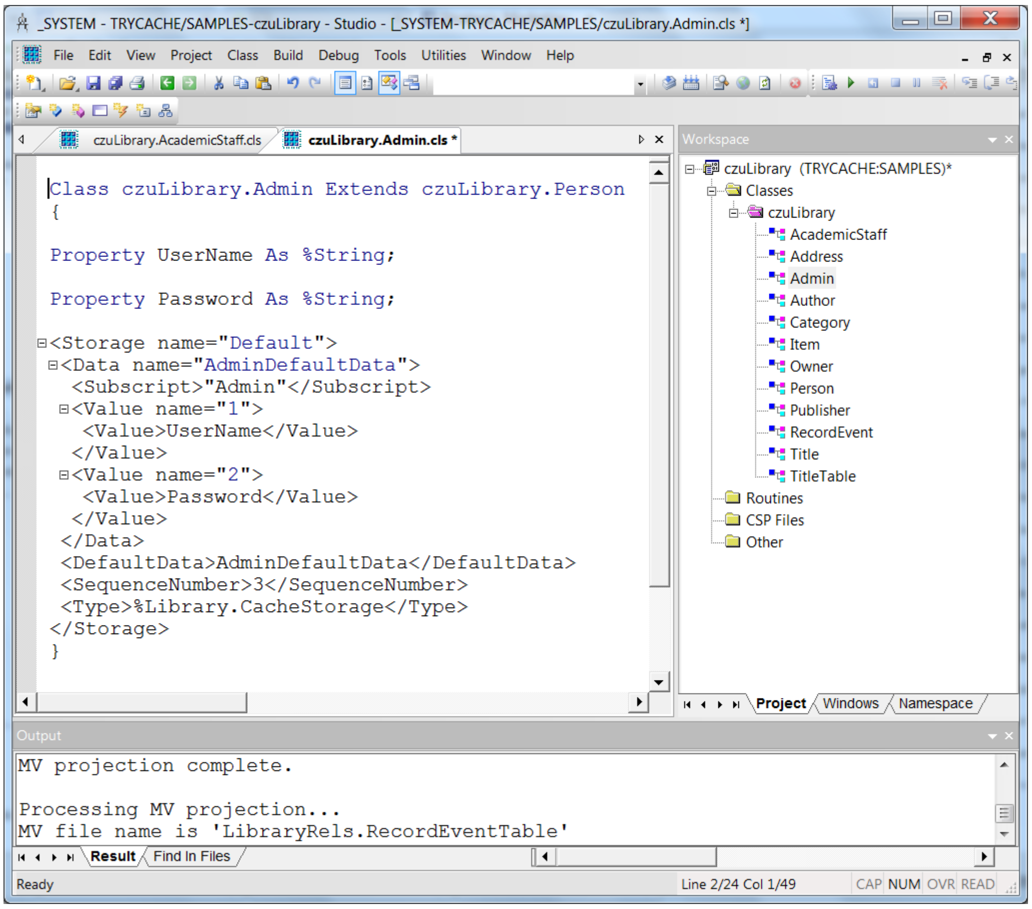
Figure 2. System Management portal query page. [Source: own]

6.3.3. Caché Studio

Caché Studio is a graphical tool for Creating, editing, deleting, and compiling Caché class definitions, CSP (Caché Server Pages) pages, Caché Basic routines, and Caché ObjectScript routines.

We can start Caché Studio to define an object class from Caché Cube in the taskbar. A window appears with the only existing connection, Local, and the namespace User already selected. Click Connect to accept these defaults.

In Caché Studio, the metadata of the database can be edited. Here are e.g. Primary keys, properties and indexes, as well as classes and their attributes managed. Figure 3 shows the Caché Studio with an open class. interesting is that tables that have data definition language commands (DDL) in System Management Portal has been created, the studio as a class and can be edited there. The reverse is also possible: In the Studio defined classes can be viewed via the System Management Portal.



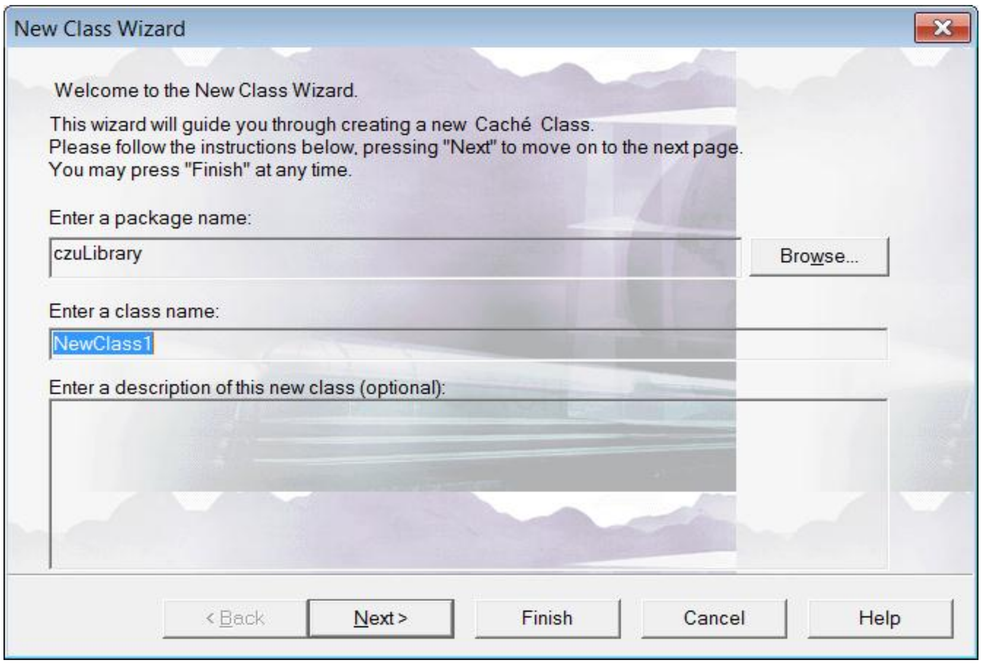
```
Class czuLibrary.Admin Extends czuLibrary.Person
{

Property UserName As %String;

Property Password As %String;

<Storage name="Default">
<Data name="AdminDefaultData">
  <Subscript>"Admin"</Subscript>
  <Value name="1">
    <Value>UserName</Value>
  </Value>
  <Value name="2">
    <Value>Password</Value>
  </Value>
 </Data>
 <DefaultData>AdminDefaultData</DefaultData>
 <SequenceNumber>3</SequenceNumber>
 <Type>%Library.CacheStorage</Type>
</Storage>
}
```

```
MV projection complete.

Processing MV projection...
MV file name is 'LibraryRels.RecordEventTable'
```

Figure 3. Caché Studio. [Source: own]

6.3.4. Creating an object class

Caché provides a wizard for creating new classes. Right-click on Classes folder in the project and select Create New Class in the context menu that opens. The New Class Wizard appears. Now specify the name and description of the new class.


New Class Wizard

Welcome to the New Class Wizard.

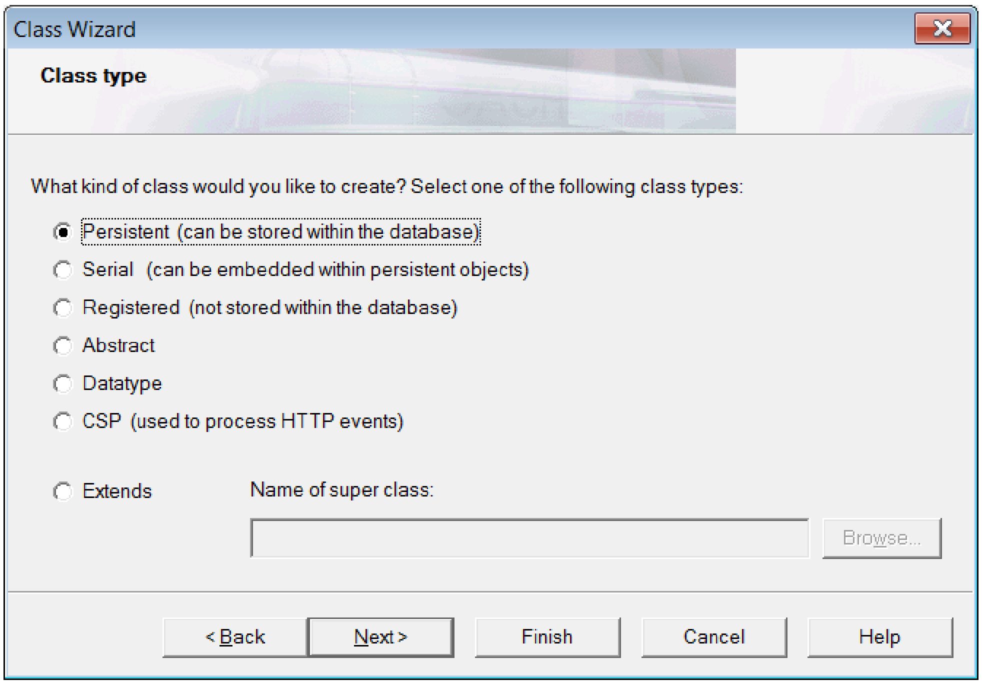
This wizard will guide you through creating a new Caché Class.
Please follow the instructions below, pressing "Next" to move on to the next page.
You may press "Finish" at any time.

Enter a package name: czuLibrary

Enter a class name: NewClass1

Enter a description of this new class (optional):

Figure 4. Caché Studio Wizard for creating classes. [Source: own]


Class Wizard

Class type

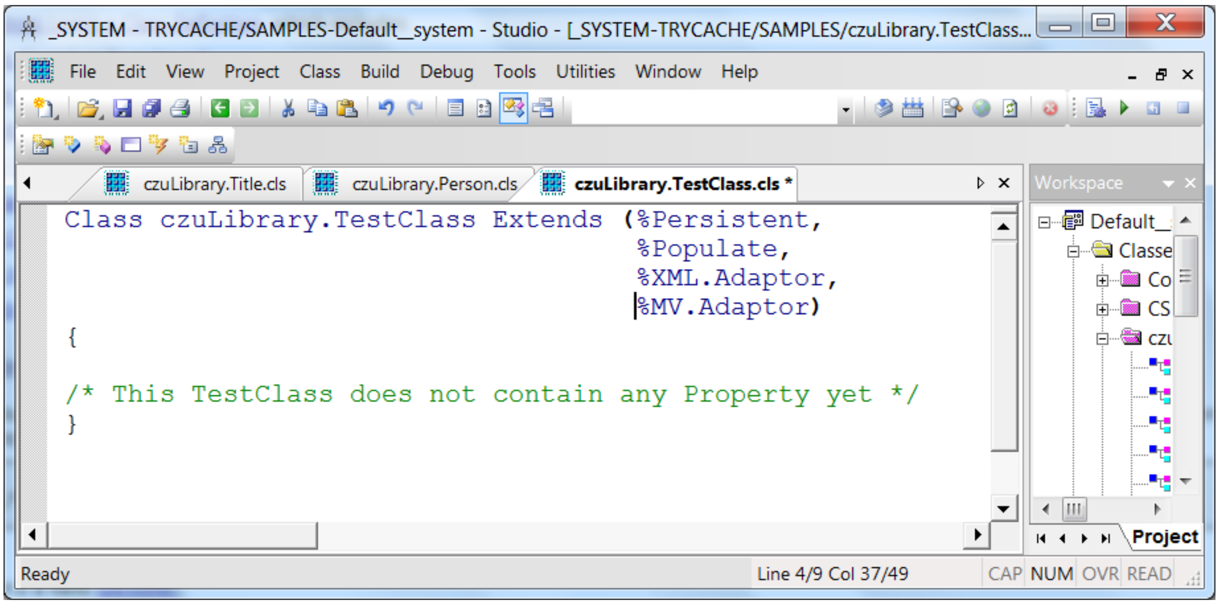
What kind of class would you like to create? Select one of the following class types:

- [✓ Persistent (can be stored within the database)]
- Serial (can be embedded within persistent objects)
- Registered (not stored within the database)
- Abstract
- Datatype
- CSP (used to process HTTP events)
- Extends

Name of super class:

Figure 5. Caché Studio Wizard supports class types. [Source: own]

After choosing needed type of class type, then Caché Studio generates an empty class with chosen name.


_SYSTEM - TRYCACHE/SAMPLES-Default__system - Studio - [_SYSTEM-TRYCACHE/SAMPLES/czuLibrary.TestClass...
File Edit View Project Class Build Debug Tools Utilities Window Help
czuLibrary.Title.cls
czuLibrary.Person.cls
czuLibrary.TestClass.cls *
Workspace
Default_
Classe
Co
CS
czu
Project
Ready
Line 4/9 Col 37/49
CAP NUM OVR READ

```
Class czuLibrary.TestClass Extends (%Persistent,
                                    %Populate,
                                    %XML.Adaptor,
                                    %MV.Adaptor)
{

/* This TestClass does not contain any Property yet */
}
```

Figure 6. An empty Caché class. [Source: own]

6.3.5. Caché Indexes

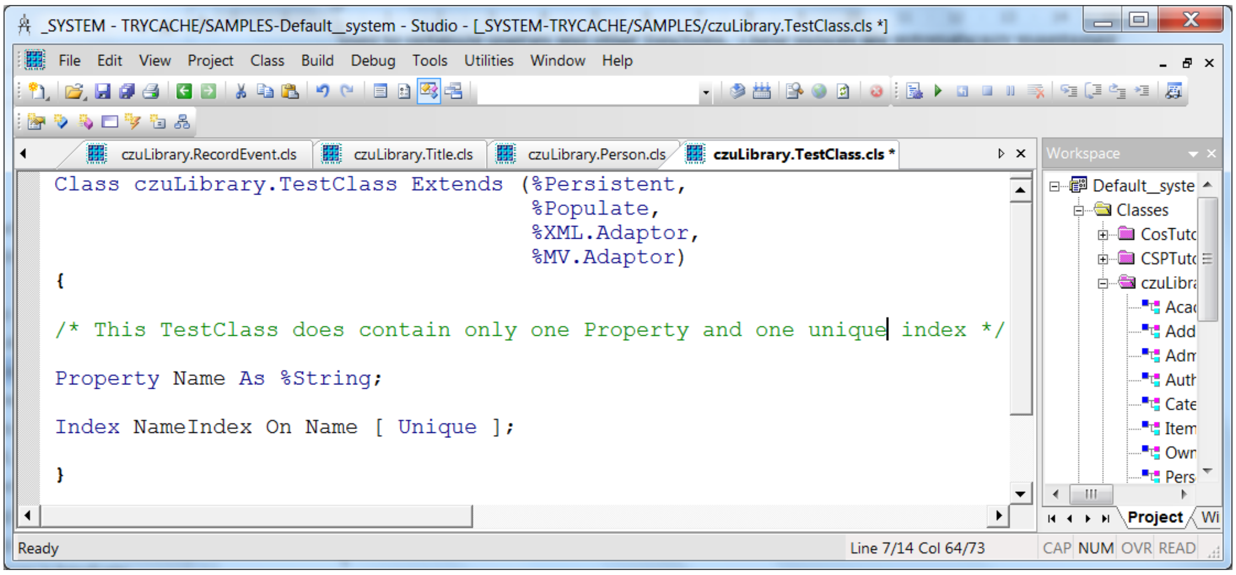
"An index is a structure maintained by a persistent class that is intended to be used to optimize queries and other functions. These indices are automatically maintained whenever DELETE, UPDATE or INSERT SQL-based operations are carried out against the database."(InterSystems, 2014)

Storable (persistent) class can have one or more indexes, which are used to perform operations such as sorting or conditional search. Caché SQL use indexes for query execution. Caché Object and SQL automatically maintains the correct index values during insert, modify and delete.

The SQL Query Processor makes use of available indices when preparing and executing SQL queries. Every index has a unique name. The following is syntax for creating indexes within class definition.

INDEX index_name ON index_property_expression_list [index_keyword_list];

This TestClass contains one property called Name which is set String data type, and one unique index on property Name named NameIndex.


_SYSTEM - TRYCACHE/SAMPLES-Default_system - Studio - [_SYSTEM-TRYCACHE/SAMPLES/czuLibrary.TestClass.cls *]
File Edit View Project Class Build Debug Tools Utilities Window Help
czuLibrary.RecordEvent.cls
czuLibrary.Title.cls
czuLibrary.Person.cls
czuLibrary.TestClass.cls *
Workspace
Default_syste
Classes
Project
Ready
Line 7/14 Col 64/73
CAP NUM OVR READ

```
Class czuLibrary.TestClass Extends (%Persistent,
                                    %Populate,
                                    %XML.Adaptor,
                                    %MV.Adaptor)
{

/* This TestClass does contain only one Property and one unique index */

Property Name As %String;

Index NameIndex On Name [ Unique ];

}
```

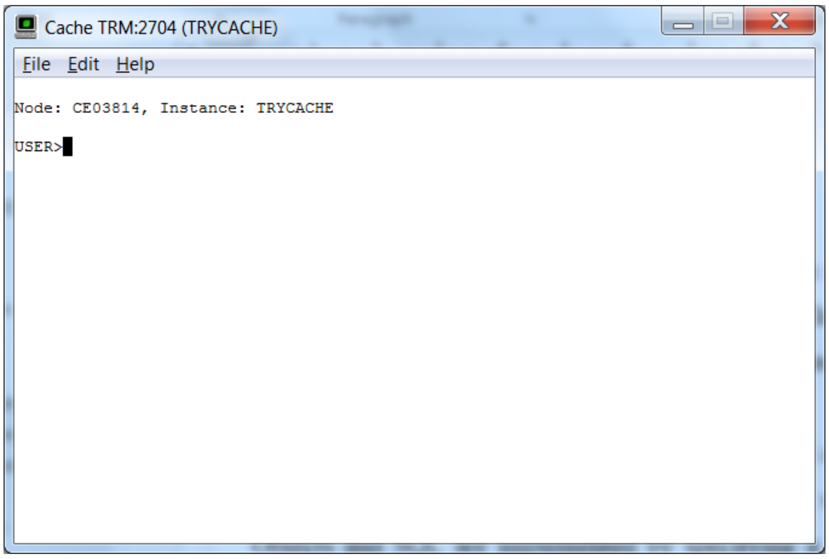
Figure 7. *TestClass* object with *NameIndex* index in Caché Studio. [Source: own]

6.3.6. Caché Terminal

The terminal is a simple console application, but at the same time the most powerful these tools. While the studio and the System Management Portal only contain the most essential functions, and thus are relatively easy to use, can via the terminal very specific settings.

For this used the previously mentioned script language Caché ObjectScript. An example of for a simple ObjectScript command is the command „do $system.SQL.Shell()", of an application to execute SQL commands within the terminal starts.

Caché Terminal provides a terminal emulation screen that you can use to logon in command mode to local or remote Caché systems. It is a Caché command shell, which is for running Caché commands. This is very useful for administrate server, manipulate database and data.



```
Node: CE03814, Instance: TRYCACHE

USER>
```

Figure 8. Caché Terminal command line tool. [Source: own]

6.4. Storing objects

"InterSystems Caché stores data in multidimensional arrays capable of carrying hierarchically structured data. That allows efficient and compact storage of data in a rich data structure. Objects and SQL are implemented by specifying a unified data dictionary that defines the classes and tables and provides a mapping to the multidimensional structures - a mapping that can be automatically generated. InterSystems Caché also allows direct multidimensional data access."(InterSystems, 2014)

InterSystems Caché is a persistent database, which means that data maintained in RAM is written to disk by background processes. So how can InterSystems Caché provide performance that is comparable to in-memory databases, which only periodically write data to some permanent data store?

"InterSystems Caché gives programmers the freedom to store and access data through objects, SQL, or direct access to multidimensional structures. Regardless of the access method, all data in InterSystems Caché's database is stored in InterSystems Caché's multidimensional arrays. Once the data is stored, all three access methods can be simultaneously used on the same data with full concurrency. InterSystems Caché automatically creates a map for how

objects and tables are stored in multidimensional structures, or the programmer can explicitly control the display."(InterSystems, 2014)

Each database InterSystems Caché starts as a special file called CACHE.DAT (the database extensions are CACHE.EXT files). There CACHE.DAT Each file in a separate directory and contains a global directory , global pointer blocks , global data , distribution maps and other information storage InterSystems Caché is necessary to maintain the database.

Caché object model differs from programming languages is that in addition to the properties and methods , you can specify the storage conditions , such as related indexes, constraints , and storage structures .

Storage structure used persistent objects independent of the logical definition of the class and is quite flexible: developers can use the default structure provided by the compiler class or they can tune the structure for specific cases.

For new classes stored Caché automatically uses the standard method of storing - *CacheStorage*. In some cases, you may need to create your own ways of storing (*CacheSQLStorage* or *CustomStorage*).

1. *CacheStorage*- used when compiling the new Caché classes stored. Techniques for using SQL and objects generated by the system.
2. *CacheSQLStorage*- used to work with existing global through SQL. Created by the programmer. In object data access method (%SaveData, %LoadData, %DeleteData) used SQL, respectively, triggers are activated. You must configure SQL projection.
3. *CustomStorage*- used to implement complex logic own object access. Created by the programmer. It is your responsibility to implement the methods %SaveData, %LoadData, %DeleteData. You can configure the SQL projection.

For storage scheme CacheSQLStorage (or CustomStorage) configured SQL view (SQL storage map).

Projection contains the necessary information to the compiler used global, their keys (subscripts), the rules pass on the keys and data location. Based on this information, the compiler generates the necessary data access methods.

6.5. Object to XML projection

“The term projecting an object to XML means *defining how that object can be used as an XML document. To project an object to XML, you add %XML.Adaptor to the superclass list of the class that defines the object.”(InterSystems, 2014)*

Caché generates additional code INT for a class that can be used as instances of XML. This xml code is generated after you compile your class.XML is hierarchical and multidimensional structure and is ideal for Caché multi-dimensional data.

Thanks to the Caché features in the multiple inheritance which allows any class dual interface with XML. This means that Cache class of easily and quickly can be created XML-documents and the description of their structures. Conversely, from the specification and XML documents can be created describing the classes and objects in Caché.

Structure of the internal representation of multidimensional data in Caché well suits to documents in XML. Therefore, developers do not need to manually code mapping for translation between XML and Caché Database. And since these applications have to perform a smaller amount of processing, they work faster.

Export to XML:

1. For compatibility with XML Caché class must be inherited from the system class *%XMLAdaptor*. It provides methods that are needed:
2. Creating a schema class for DTD (Document Type Definition) or XML Schema. Caché automatically generates schema and DTD Schema, but developers can optionally create them yourself in order to change the format of XML;
3. Automatic presentation object data in the XML format as defined DTD or XML Schema.

Import from XML:

1. XML Schema import schemes and automatic creation of the classes Caché;
2. Import data from XML documents into instances (objects) classes Caché, using a simple API;
3. Analysis (parsing) and verification (validation) documents XML, using the built- SAX parser.

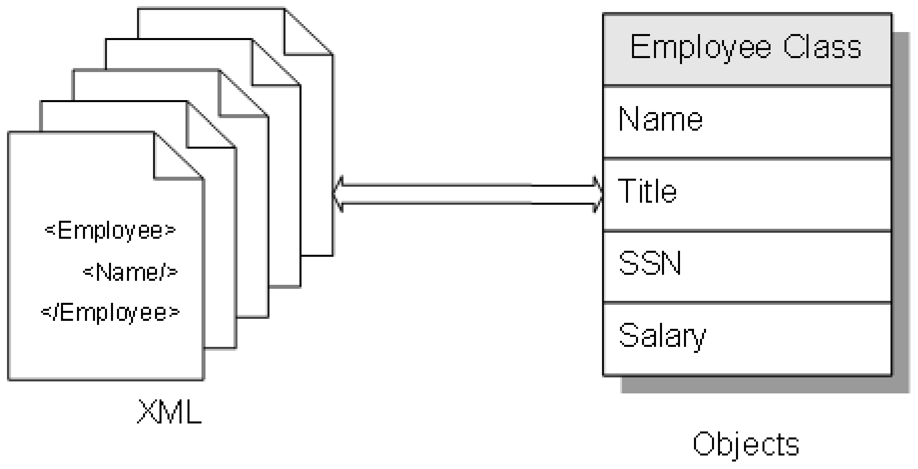


Figure 9. Caché XML projection from object Employee. [Source: InterSystems]

Caché provides additional features and tools for DOM, XPath, and XSLT.

6.6. Querying objects using Caché SQL

"Caché SQL is an extended version of standard SQL. Caché SQL provides uncompromising, standard relational access to data stored within a Caché database."(InterSystems, 2014)

Caché SQL offers the following benefits:

1. Integration with Caché objects technology — Caché SQL is tightly integrated with Caché object technology. You can mix relational and object access to data without sacrificing the performance of either approach.
2. High performance and scalability — Caché SQL offers performance and scalability superior to other relational database products. In addition, Caché SQL runs on a wide

variety of hardware and operating systems; from laptop computers to high-end, multi-CPU systems.

3. Support for standard SQL queries — Caché SQL supports SQL-92 standard syntax and commands. In most cases, you can migrate existing relational applications to Caché with little difficulty and automatically take advantage of the higher performance and object capabilities of Caché.
4. Low maintenance — unlike other relational databases, Caché applications do not require index rebuilding and table compression in deployed applications.
5. Object- and web-based applications — you can use SQL queries within Caché Object and Caché Server Page.
6. You can use Caché SQL for many purposes including:

```
/* item detail query */
select id, ,ifnull(name, ' ', name) as name
,ifnull(author->firstname, ' ', author->firstname) as author_firstname
,ifnull(author->lastname, ' ', author->lastname) as author_lastname
,ifnull(category->name, ' ', category->name) as category_name
,ifnull(publisher->name, ' ', publisher->name) as publishername
from czulibrary.item

/* record query detail */
select begindate,,item->name as item_name
,item->author->firstname || ' ' || item->author->lastname as author_name
,lender->firstname || ' ' || lender->lastname as lender_name
,borrower->firstname || ' ' || borrower->lastname as borrower_name
,item->category->name as category_name, ,item->publisher->name as publisher
from record;

/* owner list*/
select id,
    ownerlevel, item->name, owneracademicstaff->titlebefore,
    owneracademicstaff->firstname, owneracademicstaff->lastname,
    owneracademicstaff->titleafter
```

```
from czulibrary.owner

/* academicstaff list*/
select id, titlebefore,
            firstname,
             lastname,
            titleafter,
            officenumber,
            officephonenumber,
            {fn    convert(dob,sql_varchar)},
            homeaddress_street,
            homeaddress_zip
from czulibrary.academicstaff
```

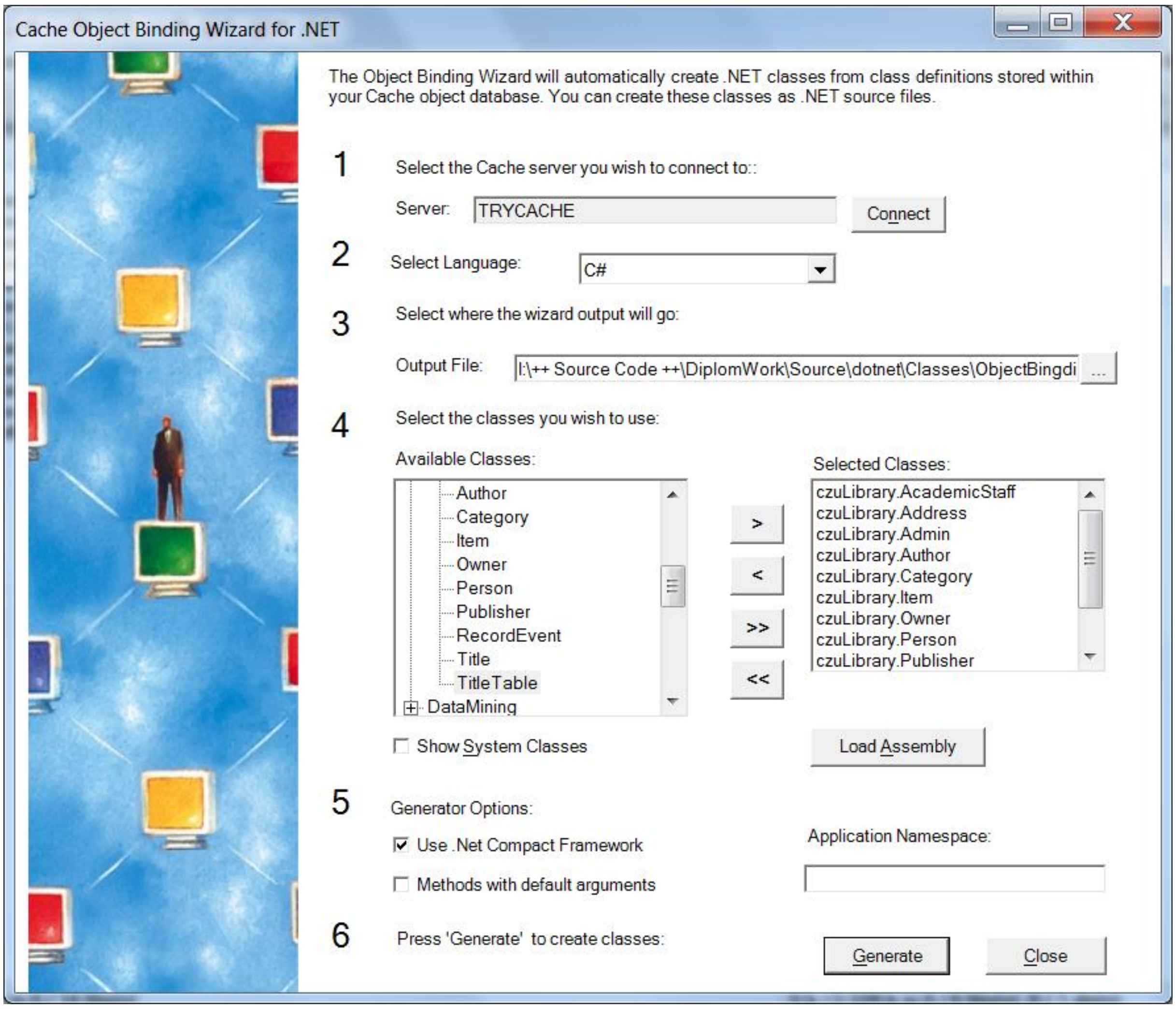
6.7. Object Binding Wizard

Caché object binding wizard is a GUI tool for creating object binding classes for OOP languages such as C++, Java and .NET.

After we have chosen an object from the Caché database server, then it will create automatically binding classes.

We can include those generated object binding classes from the wizard. Then access Caché databases objects as objects using those binding classes.

On the below Figure 10 shows steps for generating binding classes.



Cache Object Binding Wizard for .NET

The Object Binding Wizard will automatically create .NET classes from class definitions stored within your Cache object database. You can create these classes as .NET source files.

1 Select the Cache server you wish to connect to::

Server: TRYCACHE [Connect]

2 Select Language: C#

3 Select where the wizard output will go:

Output File: I:\++ Source Code ++\DiplomWork\Source\dotnet\Classes\ObjectBingdi

4 Select the classes you wish to use:

Available Classes:	Selected Classes:
Author	czuLibrary.AcademicStaff
Category	czuLibrary.Address
Item	czuLibrary.Admin
Owner	czuLibrary.Author
Person	czuLibrary.Category
Publisher	czuLibrary.Item
RecordEvent	czuLibrary.Owner
Title	czuLibrary.Person
TitleTable	czuLibrary.Publisher
DataMining	

☐ Show System Classes [Load Assembly]

5 Generator Options:

☑ Use .Net Compact Framework

☐ Methods with default arguments

Application Namespace:

6 Press 'Generate' to create classes: [Generate] [Close]

Figure 10. Caché object binding tool for .NET. [Source: own]

6.8. Querying using Caché Object script

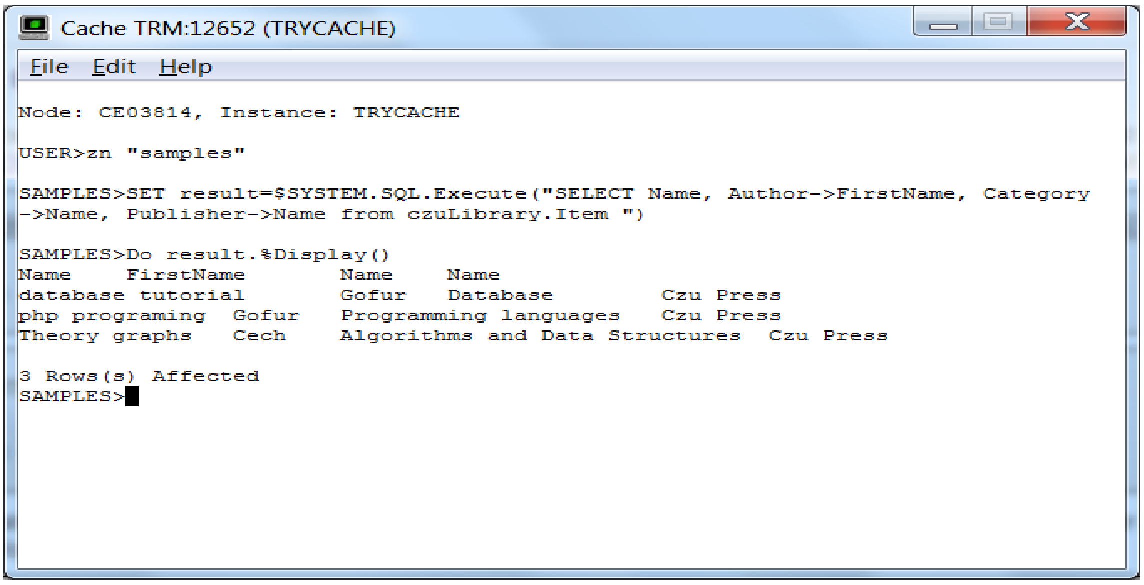
In Cache has its own object- oriented programming language Cache ObjectScript. Cache ObjectScript includes full support for the Cache object modeling and has a wide range of commands, functions, operators, and special variables that can be used for writing techniques.

Cache ObjectScript includes support for object syntax. Object syntax allows you to set and get property values and execute methods, as well as access properties and methods of reference and embedded objects from the object that references them. Object syntax also has features for the creation of new objects, and methods of the class through the structure *##class*.

Cache ObjectScript also supports macros. A set of predefined macros that can be used in the program, as well as a mechanism for creating custom macros.

Some of the key features of Caché Object Script include:

1. Powerful built-in functions for working with strings.
2. Support for multidimensional, sparse arrays: both local and global (persistent).
3. Support for indirection as well as runtime evaluation and execution of commands.
4. A wide variety of commands for directing control flow within an application.
5. Native support for objects including methods, properties, and polymorphism.
6. A set of commands for dealing with I/O devices.



```
Node: CE03814, Instance: TRYCACHE

USER>zn "samples"

SAMPLES>SET result=$SYSTEM.SQL.Execute("SELECT Name, Author->FirstName, Category
->Name, Publisher->Name from czuLibrary.Item ")

SAMPLES>Do result.%Display()
Name     FirstName        Name     Name
database tutorial         Gofur    Database         Czu Press
php programing   Gofur    Programming languages     Czu Press
Theory graphs    Cech     Algorithms and Data Structures   Czu Press

3 Rows(s) Affected
SAMPLES>
```

Figure 11. Caché Object Script example using Caché Terminal. [Source: own]

The followings are some extra features.

1. High performance and scalability of applications.
2. ZEN technology allows literally collect web-page of the built-in object components.
3. Functionally rich library of built-in components.
4. Ability to create your own components ZEN

II. IMPLEMENTATION OF THE czuLibrary.NET APPLICATION

7. General information about czuLibrary.NET

czuLibrary.NET is the name chosen for an example of object database implementation in environment of Caché and evaluate it from the perspective of differences between pure object and relational. As the name saying, it is a library application which is written in .NET using Caché object database. But it is only intended for a group of academic staffs. For instance, it is for one department of a Faculty.

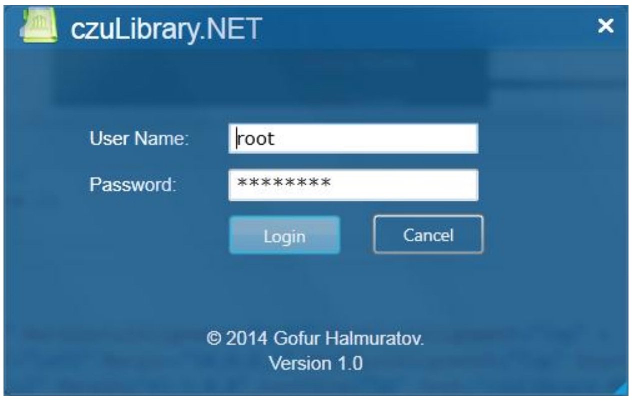
7.1. Features

The application records information about borrowing items from the owner of the items to lender between academic staffs. An item can have more than one owner which are leveled form *level 1* to *level 5*. *Level 1* is the highest owner. For example, an academic staff can borrow an item form any of those owners.

czuLibrary.NET has also managing features of Academic Titles, Categories, Publishers, Authors, Academic Staffs, and Items.

Main goal of this application is to implement an example object database in the environment of the Caché object-oriented database and evaluate it from the perspective of differences between pure object and relational.

7.2. Screenshots

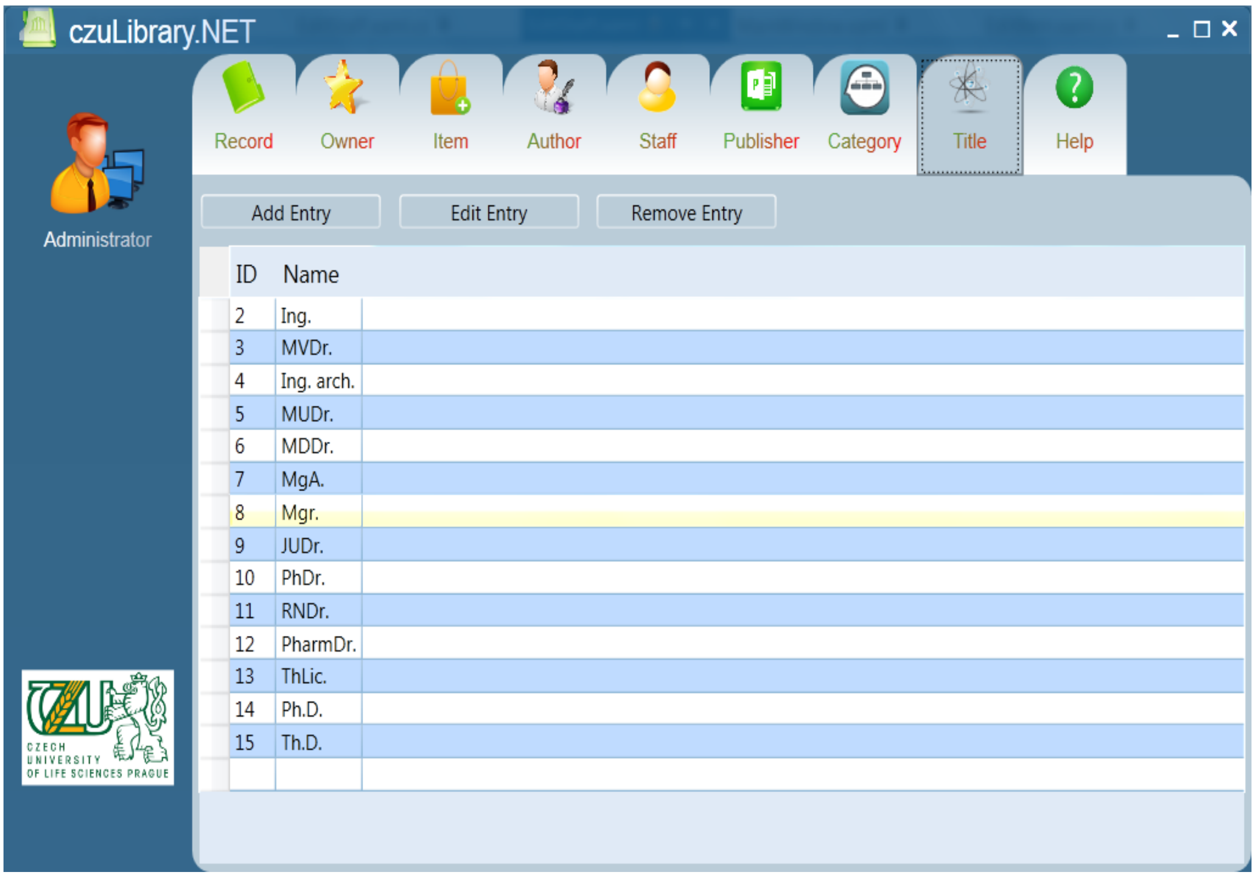


User Name: root

Password: ********

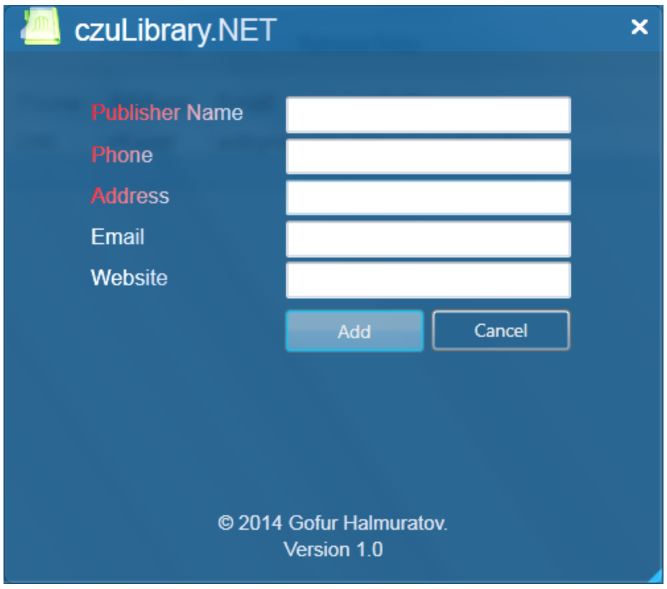
© 2014 Gofur Halmuratov.
Version 1.0

Figure 12. czuLibrary login form. [Source: own]



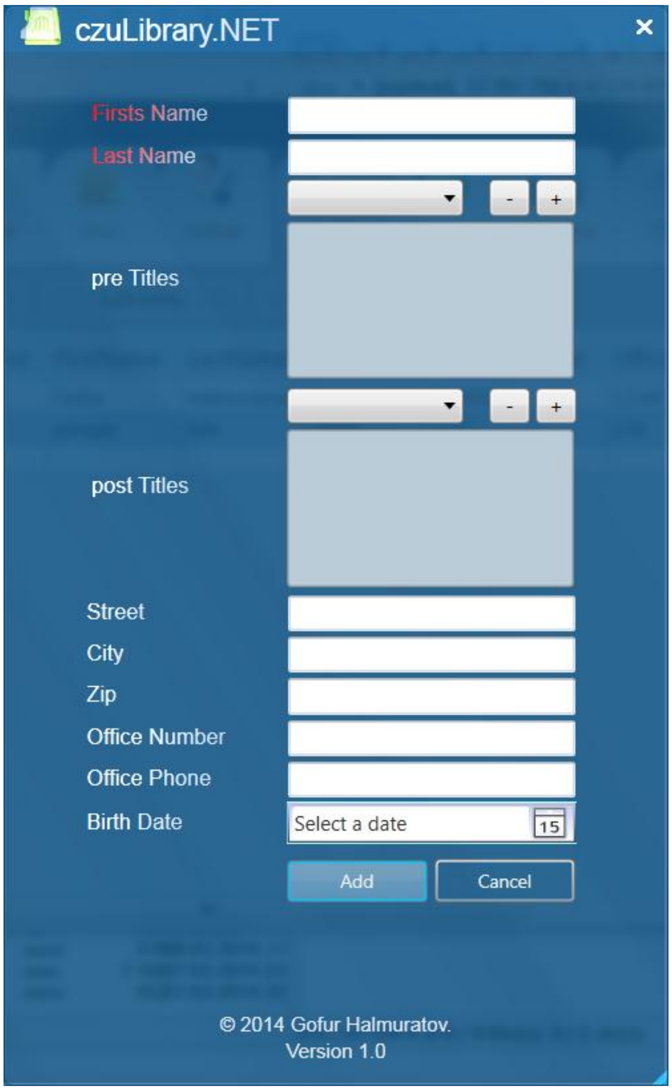
ID	Name
2	Ing.
3	MVDr.
4	Ing. arch.
5	MUDr.
6	MDDr.
7	MgA.
8	Mgr.
9	JUDr.
10	PhDr.
11	RNDr.
12	PharmDr.
13	ThLic.
14	Ph.D.
15	Th.D.

Figure 13. *czuLibrary.NET* application main administrative form. [Source: own]


czuLibrary.NET

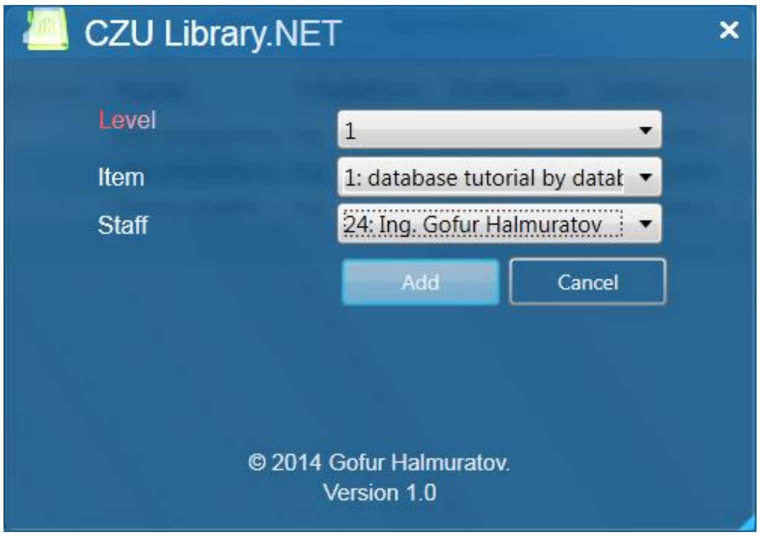
Publisher Name	
Phone	
Address	
Email	
Website	

Figure 14. *czuLibrary.NET Publisher* inserting form. [Source: own]


czuLibrary.NET

Firsts Name	
Last Name	
pre Titles	
post Titles	
Street	
City	
Zip	
Office Number	
Office Phone	
Birth Date	Select a date

Figure 15. *czuLibrary.NET* application *AcademicStaff* Inserting form. [Source: own]


CZU Library.NET

Level: 1

Item: 1: database tutorial by datal

Staff: 24: Ing. Gofur Halmuratov

Figure 16. *czuLibrary.NET* application *Owner* Inserting form. [Source: own]

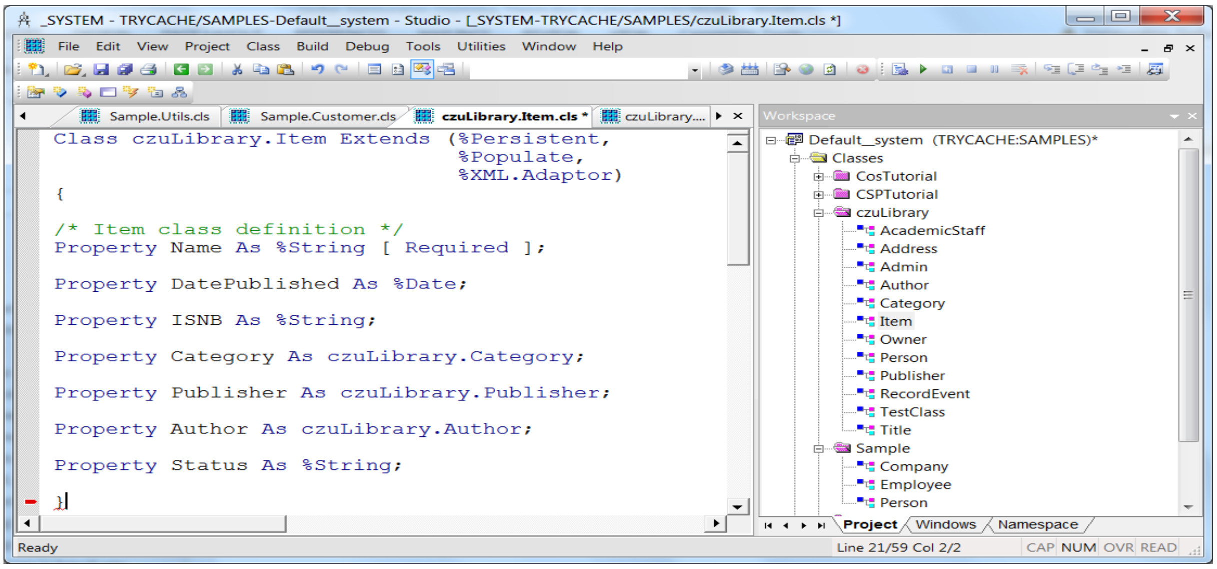
8. *Database*

8.1. *czuLibrary.NET* database model

The application have quite complex object database model. It has following persistent and serial classes:

1. *Title* is a persistent class to store academic titles,
2. *Category* is a persistent class to store an item's categories list,
3. *Publisher* is a persistent class to store an item's publisher,
4. *Person* is a persistent class to store common information about persons.
5. *Author* is a persistent class to store item's author. It is a derived class from *Person*.
6. *Admin* is a persistent class to store administrators list. It is a derived class from *Person*.
7. *AcademicStaff* is a persistent class to store Academic staff list. It is a derived class from *Person.*
8. *Item* is a persistent class to store items, it is not derived class.
9. *Owner* is a persistent class to store item's owners with levels.
10. *Record* is a persistent class to store information about borrowers and lenders.
11. *Address* is a serial object which can be embedded in persistent class.

Item class definition is shown in Figure 17.



```
Class czuLibrary.Item Extends (%Persistent,
                               %Populate,
                               %XML.Adaptor)
{

/* Item class definition */
Property Name As %String [ Required ];

Property DatePublished As %Date;

Property ISNB As %String;

Property Category As czuLibrary.Category;

Property Publisher As czuLibrary.Publisher;

Property Author As czuLibrary.Author;

Property Status As %String;

}
```

Figure 17. Item class definition in Caché Studio. [Source: own]

In figure 18 is shown UML class diagram of *czuLibrary.NET*. The Class diagram describes the structure of *czuLibrary.NET* application by showing the class definition, their attributes (Property), methods, and the relationships among objects. This helps us to understand the application better. For example, Class *AcademicStaff* is derived from Person and so on.

In figure 19 is shown possible relational model of the same application, if we have not chosen the Object-oriented database, then the model would be look like as in Figure 19. I have drawn the relational model to see what would be difference between those designs. By comparing the models, we can distinguish Relational and object models in a case of particular *czuLibrary.NET* application.

As it is seen on rational database model that we need to define many foreign keys, primary keys. It will slow down the database performance. In a case of object database, we do not need foreign key.

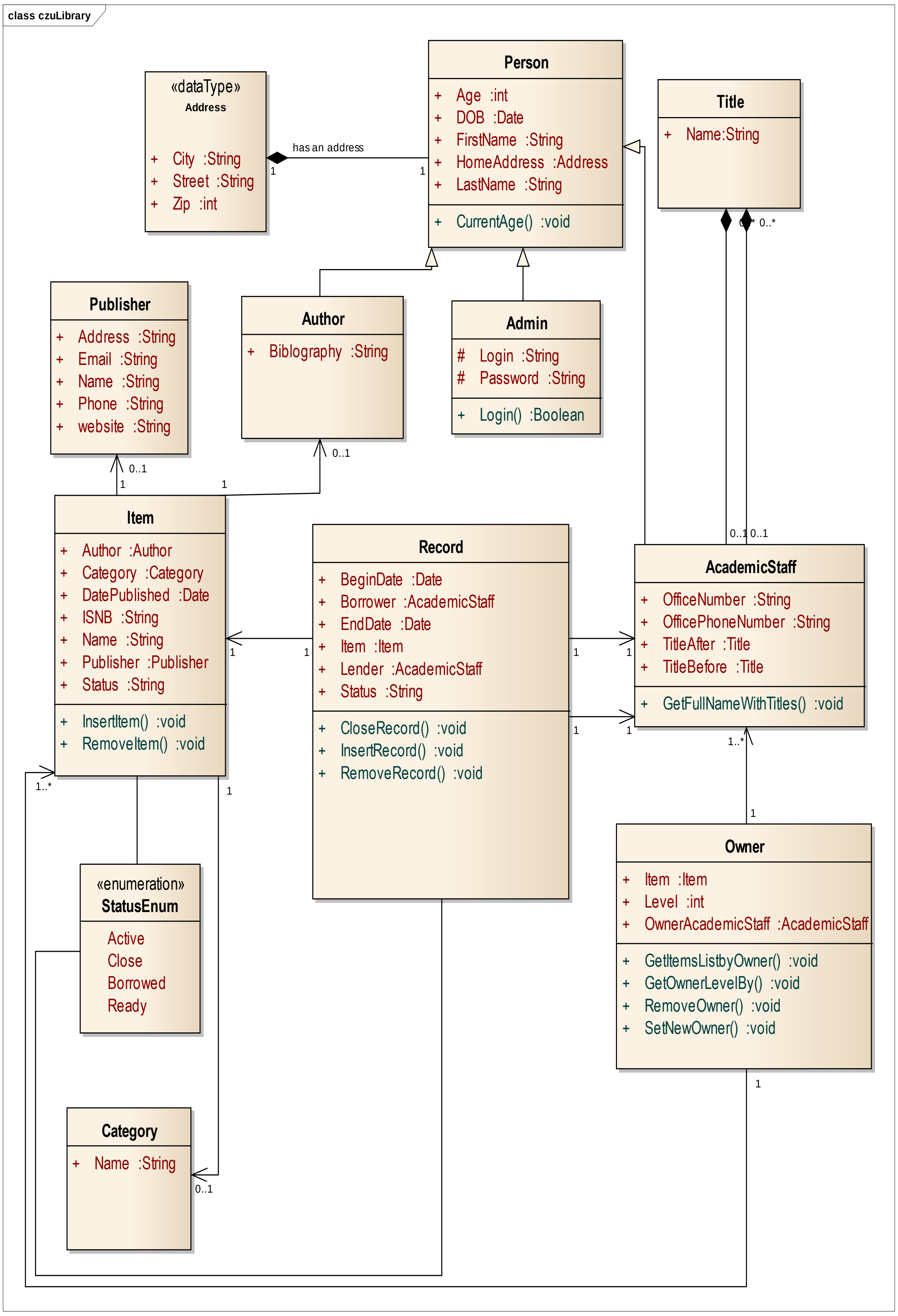


Figure 18. UML Class diagram for *czuLibrary.NET* application. [Source: own]

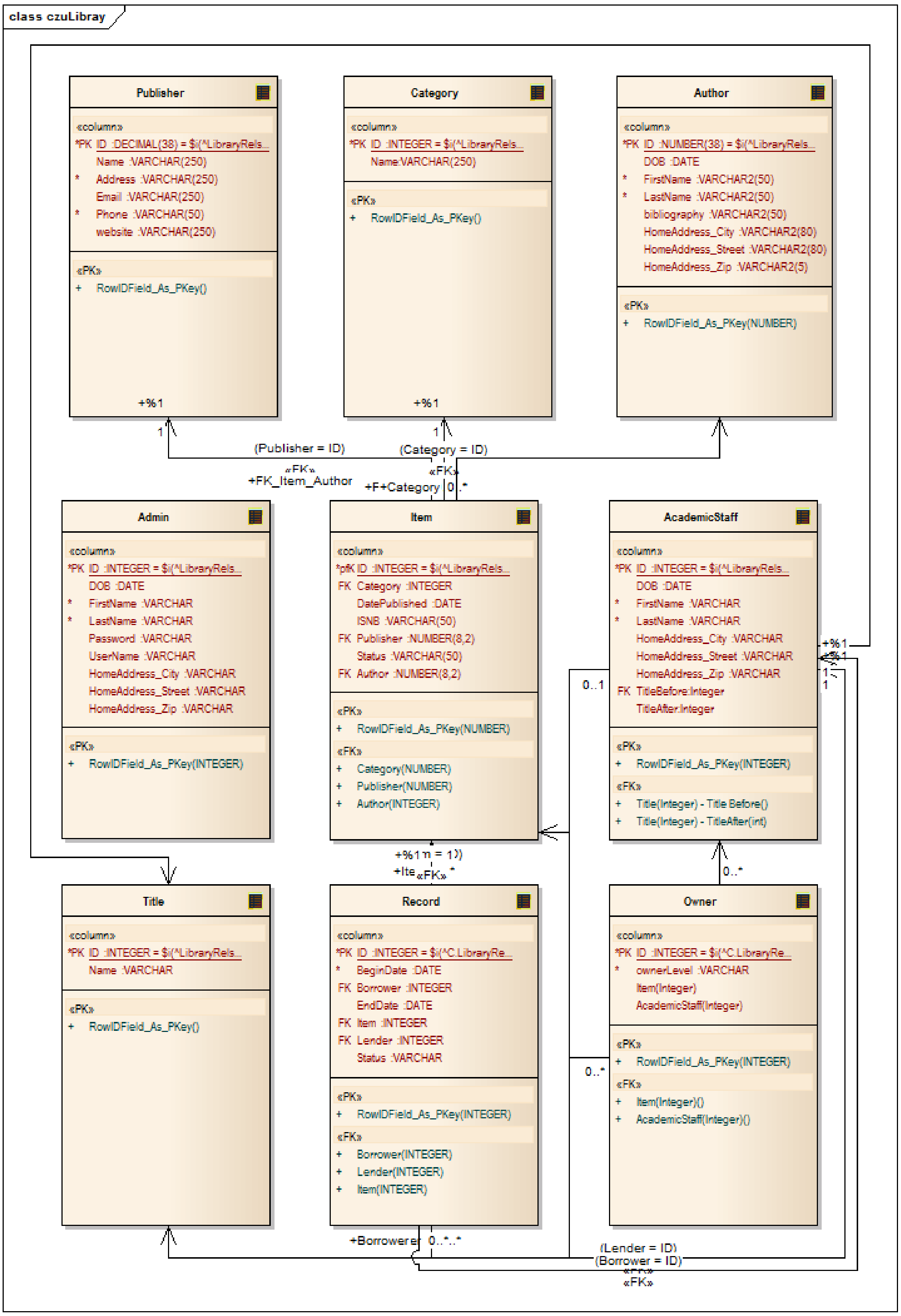


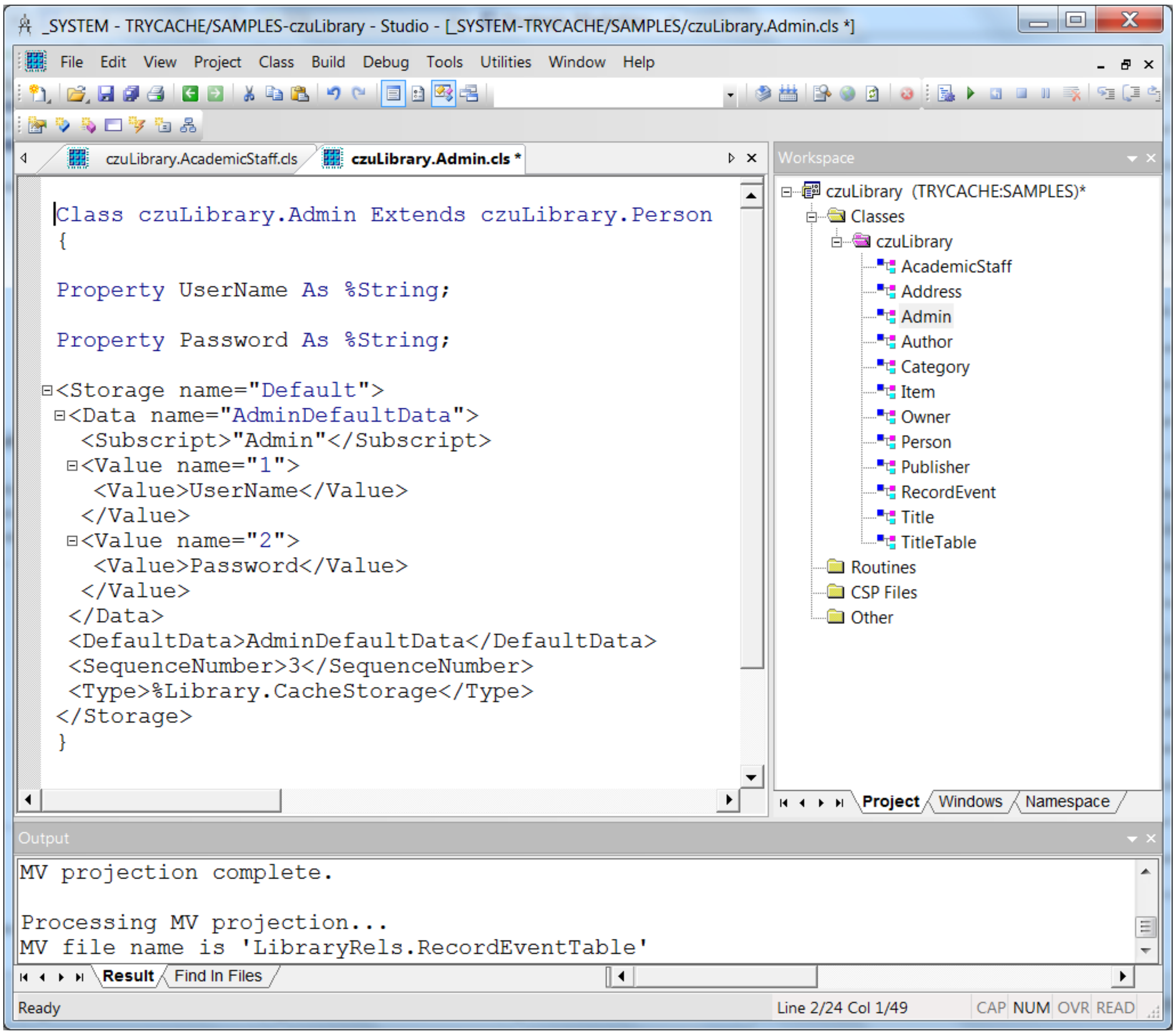
Figure 19. Relational database model for *czuLibrary.NET*. [Source: own]

The Relational model(Figure 19) has many foreign key, it would make problems or slow down querying because we need to join tables as much as foreign keys defined in the model. The Figure 32 shows a difference which I noticed during implementation of the application.

8.2. *Creating* databases using Caché studio application

As the structure is drawn in UML Class diagram, it help us to understand the structure of *czuLibrary.NET*. with using the UML Class diagram, we can easly create class definition for shown classes in the diagram.

In Figure 20 is shown whole czuLibrary database which was created using Caché studio.



```
Class czuLibrary.Admin Extends czuLibrary.Person
{

Property UserName As %String;

Property Password As %String;

<Storage name="Default">
<Data name="AdminDefaultData">
  <Subscript>"Admin"</Subscript>
 <Value name="1">
   <Value>UserName</Value>
  </Value>
 <Value name="2">
   <Value>Password</Value>
  </Value>
 </Data>
 <DefaultData>AdminDefaultData</DefaultData>
 <SequenceNumber>3</SequenceNumber>
 <Type>%Library.CacheStorage</Type>
</Storage>
}
```

```
MV projection complete.

Processing MV projection...
MV file name is 'LibraryRels.RecordEventTable'
```

Figure 20. *czuLibrary.NET* overview in Caché Studio. [Source: own]

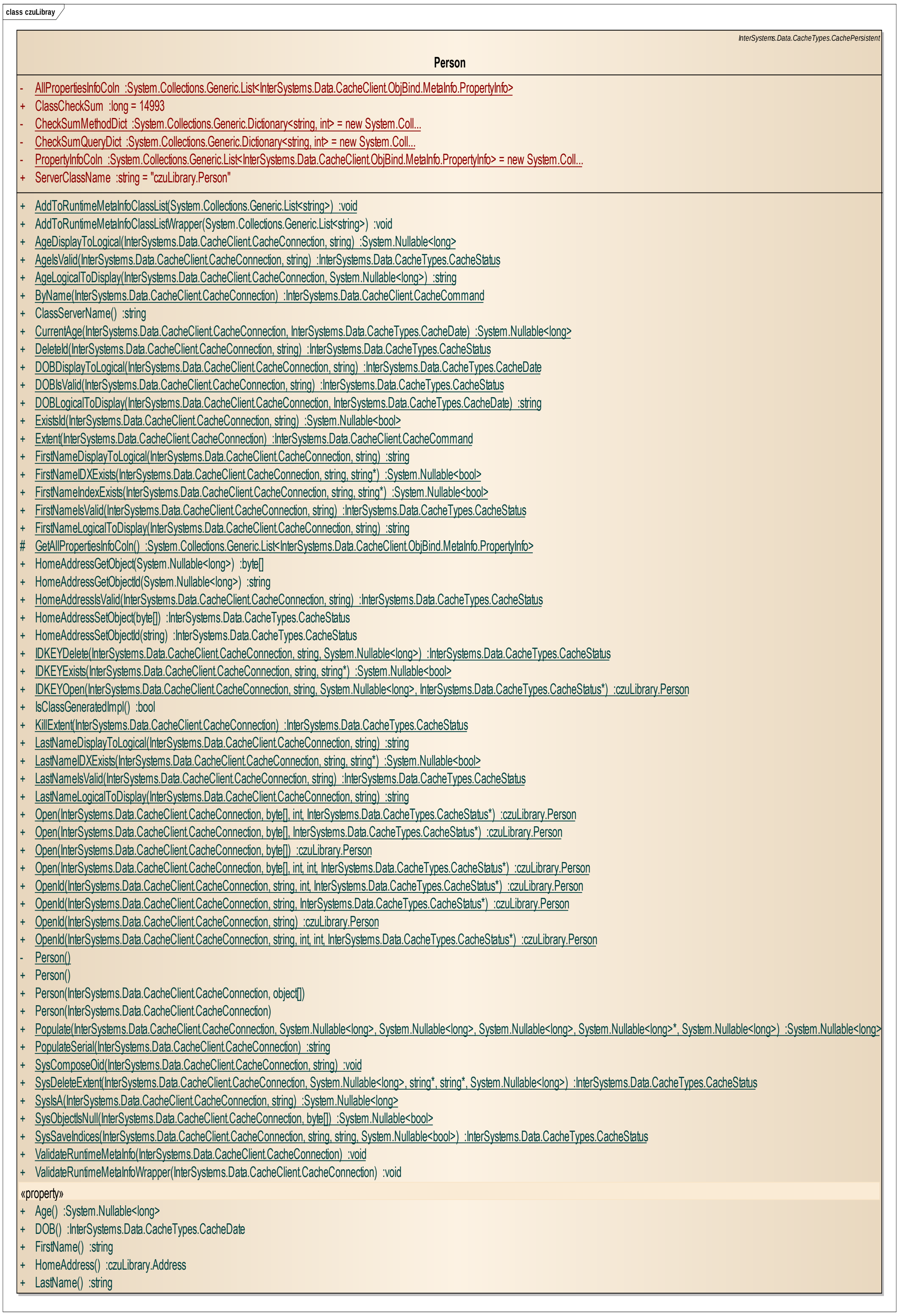
9. Object binding class for .NET

Before accessing Caché objects in .NET, we need to create object binding classes for *czuLibrary.NET* application.

After using Caché Wizard for each of Caché classes of *czuLibrary.NET* application, we will have an object binding class for each object from Caché database.

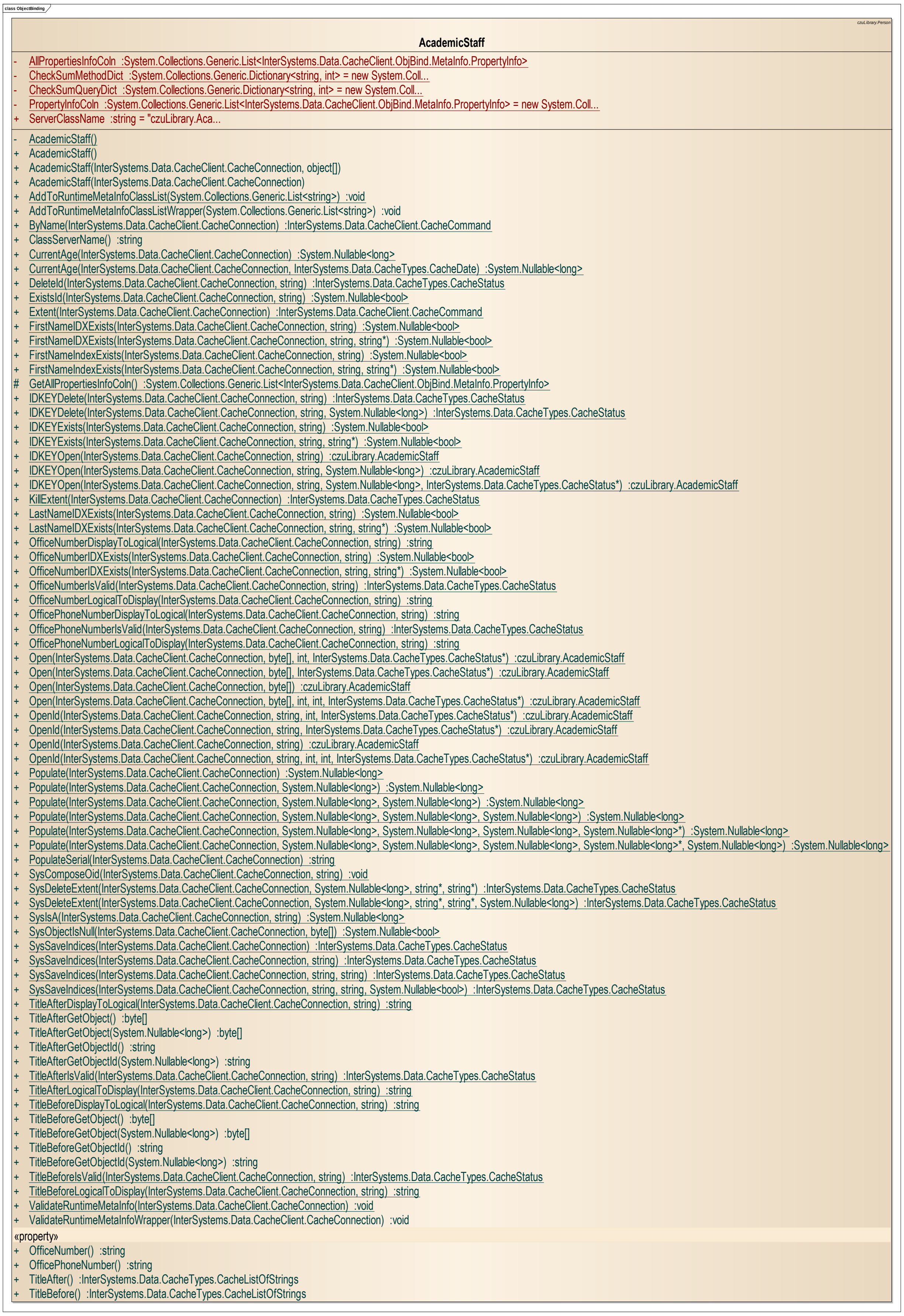
For example, as you see in the Figure 22, Caché object binding Wizard also generated extra methods for *AcademicStaff* objects to help us to access and manipulate object's data easily.

Object binding classes of *czuLibrary.NET* are shown on the pictures between Figure 21 and Figure 31.



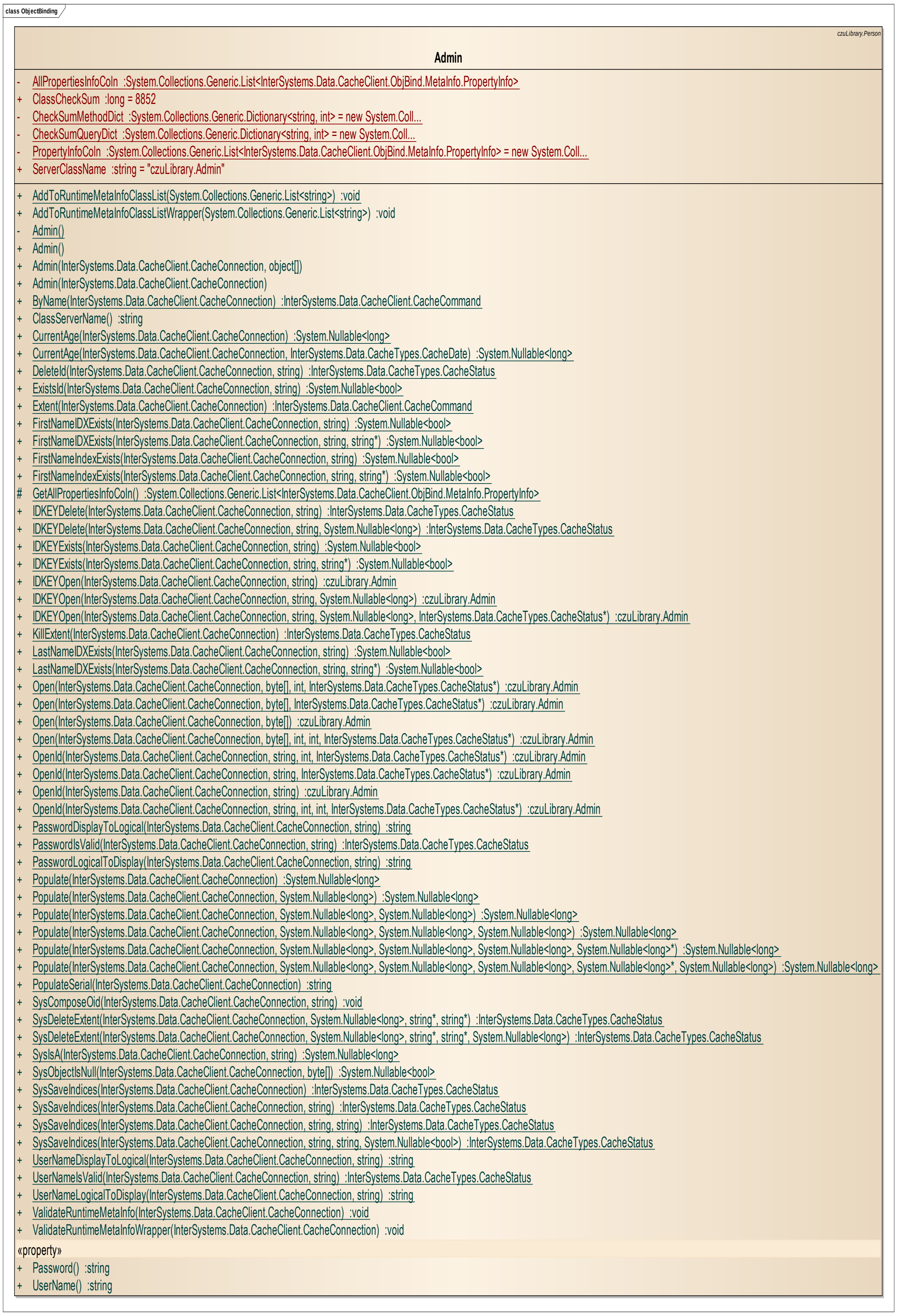
```
class czuLibray

                                              InterSystems.Data.CacheTypes.CachePersistent
Person

-  AllPropertiesInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  ClassCheckSum  :long = 14993
-  CheckSumMethodDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  CheckSumQueryDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  PropertyInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo> = new System.Coll...
+  ServerClassName  :string = "czuLibrary.Person"

+  AddToRuntimeMetaInfoClassList(System.Collections.Generic.List<string>)  :void
+  AddToRuntimeMetaInfoClassListWrapper(System.Collections.Generic.List<string>)  :void
+  AgeDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<long>
+  AgeIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  AgeLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>)  :string
+  ByName(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheClient.CacheCommand
+  ClassServerName()  :string
+  CurrentAge(InterSystems.Data.CacheClient.CacheConnection, InterSystems.Data.CacheTypes.CacheDate)  :System.Nullable<long>
+  DeleteId(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  DOBDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheDate
+  DOBIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  DOBLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, InterSystems.Data.CacheTypes.CacheDate)  :string
+  ExistsId(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  Extent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheClient.CacheCommand
+  FirstNameDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  FirstNameIDXExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  FirstNameIndexExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  FirstNameIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  FirstNameLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
#  GetAllPropertiesInfoColn()  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  HomeAddressGetObject(System.Nullable<long>)  :byte[]
+  HomeAddressGetObjectId(System.Nullable<long>)  :string
+  HomeAddressIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  HomeAddressSetObject(byte[])  :InterSystems.Data.CacheTypes.CacheStatus
+  HomeAddressSetObjectId(string)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Person
+  IsClassGeneratedImpl()  :bool
+  KillExtent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  LastNameDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  LastNameIDXExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  LastNameIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  LastNameLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Person
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Person
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[])  :czuLibrary.Person
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Person
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Person
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Person
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Person
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Person
-  Person()
+  Person()
+  Person(InterSystems.Data.CacheClient.CacheConnection, object[])
+  Person(InterSystems.Data.CacheClient.CacheConnection)
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*, System.Nullable<long>)  :System.Nullable<long>
+  PopulateSerial(InterSystems.Data.CacheClient.CacheConnection)  :string
+  SysComposeOid(InterSystems.Data.CacheClient.CacheConnection, string)  :void
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysIsA(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<long>
+  SysObjectIsNull(InterSystems.Data.CacheClient.CacheConnection, byte[])  :System.Nullable<bool>
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string, System.Nullable<bool>)  :InterSystems.Data.CacheTypes.CacheStatus
+  ValidateRuntimeMetaInfo(InterSystems.Data.CacheClient.CacheConnection)  :void
+  ValidateRuntimeMetaInfoWrapper(InterSystems.Data.CacheClient.CacheConnection)  :void
«property»
+  Age()  :System.Nullable<long>
+  DOB()  :InterSystems.Data.CacheTypes.CacheDate
+  FirstName()  :string
+  HomeAddress()  :czuLibrary.Address
+  LastName()  :string
```

Figure 21. *Person* binding class for .NET generated using Caché Wizard. [Source: own]



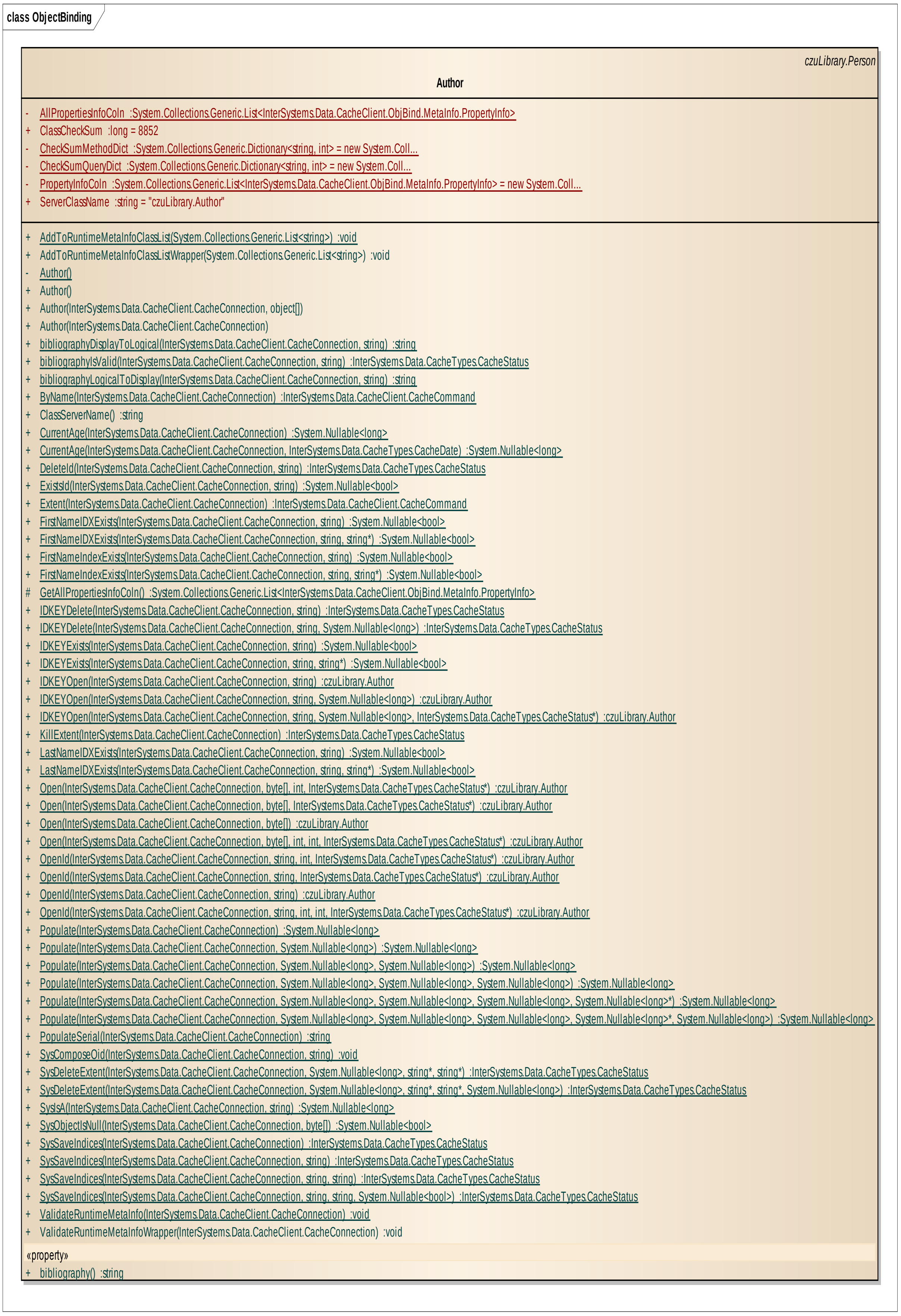
```
-  AllPropertiesInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
-  CheckSumMethodDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  CheckSumQueryDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  PropertyInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo> = new System.Coll...
+  ServerClassName  :string = "czuLibrary.Aca...

-  AcademicStaff()
+  AcademicStaff()
+  AcademicStaff(InterSystems.Data.CacheClient.CacheConnection, object[])
+  AcademicStaff(InterSystems.Data.CacheClient.CacheConnection)
+  AddToRuntimeMetaInfoClassList(System.Collections.Generic.List<string>)  :void
+  AddToRuntimeMetaInfoClassListWrapper(System.Collections.Generic.List<string>)  :void
+  ByName(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheClient.CacheCommand
+  ClassServerName()  :string
+  CurrentAge(InterSystems.Data.CacheClient.CacheConnection)  :System.Nullable<long>
+  CurrentAge(InterSystems.Data.CacheClient.CacheConnection, InterSystems.Data.CacheTypes.CacheDate)  :System.Nullable<long>
+  DeleteId(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  ExistsId(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  Extent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheClient.CacheCommand
+  FirstNameIDXExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  FirstNameIDXExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  FirstNameIndexExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  FirstNameIndexExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
#  GetAllPropertiesInfoColn()  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.AcademicStaff
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :czuLibrary.AcademicStaff
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.AcademicStaff
+  KillExtent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  LastNameIDXExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  LastNameIDXExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  OfficeNumberDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  OfficeNumberIDXExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  OfficeNumberIDXExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  OfficeNumberIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  OfficeNumberLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  OfficePhoneNumberDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  OfficePhoneNumberIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  OfficePhoneNumberLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.AcademicStaff
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.AcademicStaff
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[])  :czuLibrary.AcademicStaff
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.AcademicStaff
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.AcademicStaff
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.AcademicStaff
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.AcademicStaff
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.AcademicStaff
+  Populate(InterSystems.Data.CacheClient.CacheConnection)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*, System.Nullable<long>)  :System.Nullable<long>
+  PopulateSerial(InterSystems.Data.CacheClient.CacheConnection)  :string
+  SysComposeOid(InterSystems.Data.CacheClient.CacheConnection, string)  :void
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysIsA(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<long>
+  SysObjectIsNull(InterSystems.Data.CacheClient.CacheConnection, byte[])  :System.Nullable<bool>
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string, System.Nullable<bool>)  :InterSystems.Data.CacheTypes.CacheStatus
+  TitleAfterDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  TitleAfterGetObject()  :byte[]
+  TitleAfterGetObject(System.Nullable<long>)  :byte[]
+  TitleAfterGetObjectId()  :string
+  TitleAfterGetObjectId(System.Nullable<long>)  :string
+  TitleAfterIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  TitleAfterLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  TitleBeforeDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  TitleBeforeGetObject()  :byte[]
+  TitleBeforeGetObject(System.Nullable<long>)  :byte[]
+  TitleBeforeGetObjectId()  :string
+  TitleBeforeGetObjectId(System.Nullable<long>)  :string
+  TitleBeforeIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  TitleBeforeLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  ValidateRuntimeMetaInfo(InterSystems.Data.CacheClient.CacheConnection)  :void
+  ValidateRuntimeMetaInfoWrapper(InterSystems.Data.CacheClient.CacheConnection)  :void
«property»
+  OfficeNumber()  :string
+  OfficePhoneNumber()  :string
+  TitleAfter()  :InterSystems.Data.CacheTypes.CacheListOfStrings
+  TitleBefore()  :InterSystems.Data.CacheTypes.CacheListOfStrings
```

Figure 22. Object binding *AcademicStaff* class for .NET. [Souce: own]



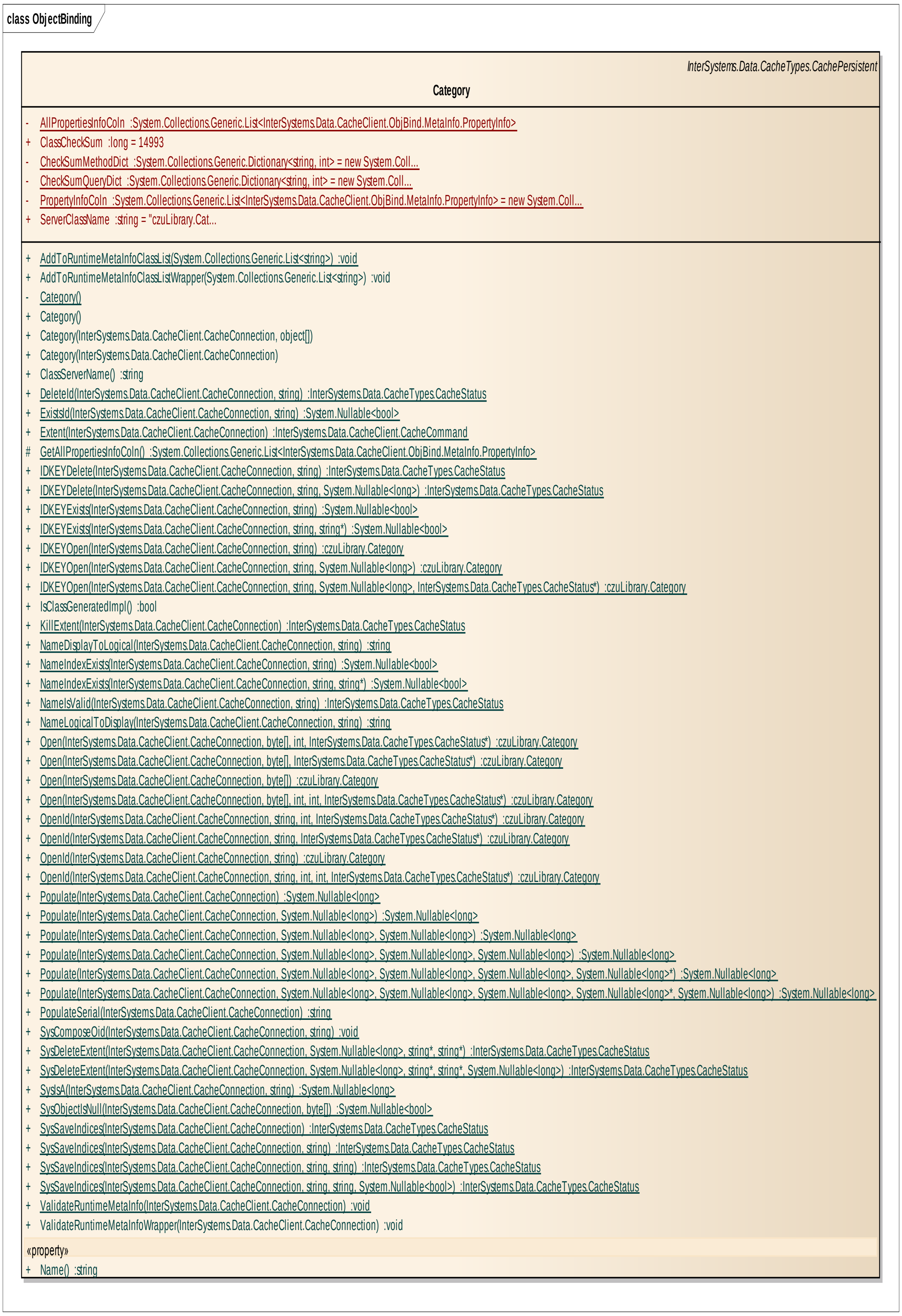
```
class ObjectBinding

                                                                 czuLibrary.Person
                                   Admin

-  AllPropertiesInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  ClassCheckSum  :long = 8852
-  CheckSumMethodDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  CheckSumQueryDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  PropertyInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo> = new System.Coll...
+  ServerClassName  :string = "czuLibrary.Admin"

+  AddToRuntimeMetaInfoClassList(System.Collections.Generic.List<string>)  :void
+  AddToRuntimeMetaInfoClassListWrapper(System.Collections.Generic.List<string>)  :void
-  Admin()
+  Admin()
+  Admin(InterSystems.Data.CacheClient.CacheConnection, object[])
+  Admin(InterSystems.Data.CacheClient.CacheConnection)
+  ByName(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheClient.CacheCommand
+  ClassServerName()  :string
+  CurrentAge(InterSystems.Data.CacheClient.CacheConnection)  :System.Nullable<long>
+  CurrentAge(InterSystems.Data.CacheClient.CacheConnection, InterSystems.Data.CacheTypes.CacheDate)  :System.Nullable<long>
+  DeleteId(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  ExistsId(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  Extent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheClient.CacheCommand
+  FirstNameIDXExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  FirstNameIDXExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  FirstNameIndexExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  FirstNameIndexExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
#  GetAllPropertiesInfoColn()  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Admin
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :czuLibrary.Admin
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Admin
+  KillExtent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  LastNameIDXExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  LastNameIDXExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Admin
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Admin
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[])  :czuLibrary.Admin
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Admin
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Admin
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Admin
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Admin
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Admin
+  PasswordDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  PasswordIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  PasswordLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  Populate(InterSystems.Data.CacheClient.CacheConnection)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*, System.Nullable<long>)  :System.Nullable<long>
+  PopulateSerial(InterSystems.Data.CacheClient.CacheConnection)  :string
+  SysComposeOid(InterSystems.Data.CacheClient.CacheConnection, string)  :void
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysIsA(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<long>
+  SysObjectIsNull(InterSystems.Data.CacheClient.CacheConnection, byte[])  :System.Nullable<bool>
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string, System.Nullable<bool>)  :InterSystems.Data.CacheTypes.CacheStatus
+  UserNameDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  UserNameIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  UserNameLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  ValidateRuntimeMetaInfo(InterSystems.Data.CacheClient.CacheConnection)  :void
+  ValidateRuntimeMetaInfoWrapper(InterSystems.Data.CacheClient.CacheConnection)  :void
«property»
+  Password()  :string
+  UserName()  :string
```

Figure 23. Object binding *Admin* class for .NET. [Souce: own]



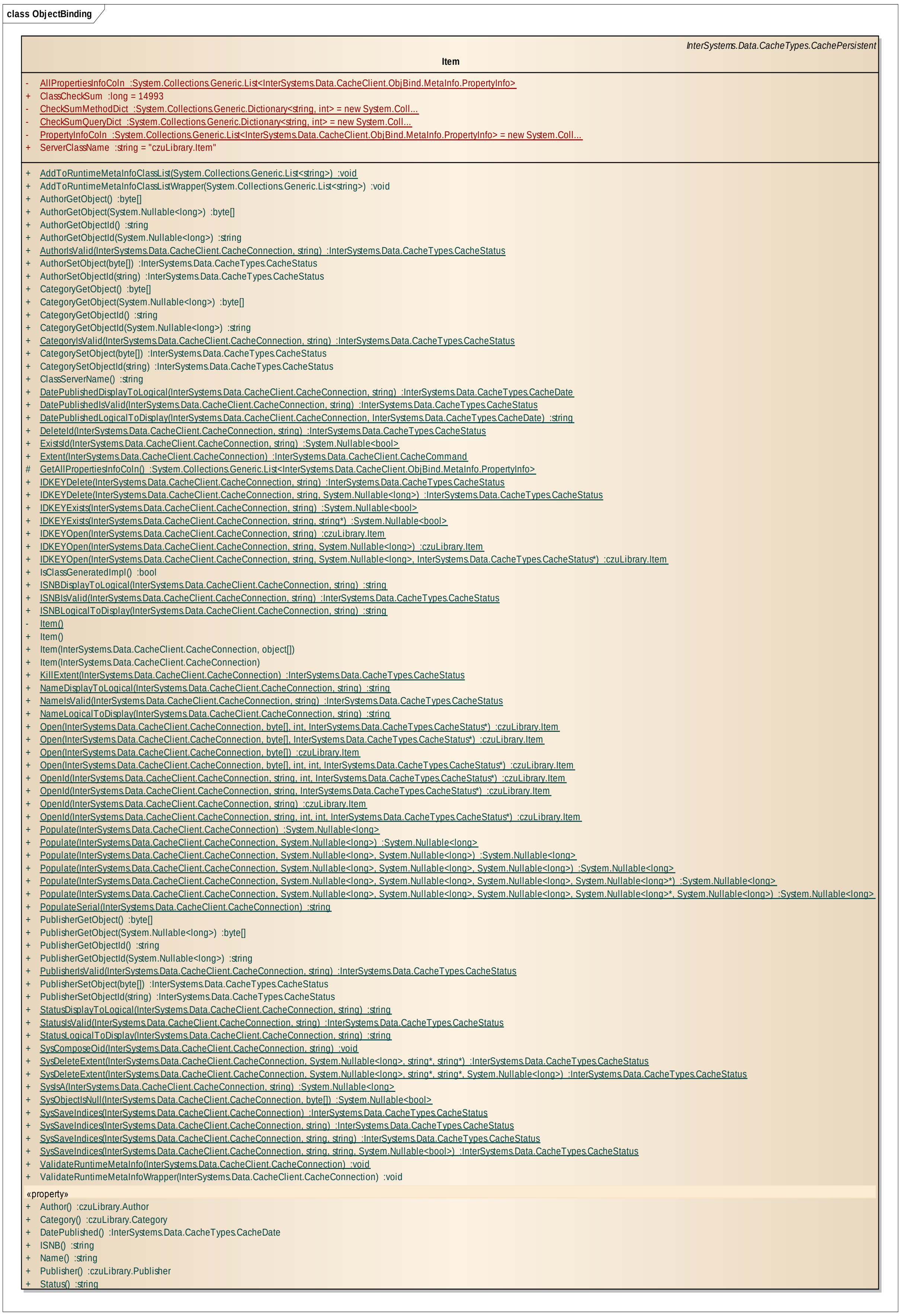
```
class ObjectBinding

czuLibrary.Person
Author

-  AllPropertiesInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  ClassCheckSum  :long = 8852
-  CheckSumMethodDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  CheckSumQueryDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  PropertyInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo> = new System.Coll...
+  ServerClassName  :string = "czuLibrary.Author"

+  AddToRuntimeMetaInfoClassList(System.Collections.Generic.List<string>)  :void
+  AddToRuntimeMetaInfoClassListWrapper(System.Collections.Generic.List<string>)  :void
-  Author()
+  Author()
+  Author(InterSystems.Data.CacheClient.CacheConnection, object[])
+  Author(InterSystems.Data.CacheClient.CacheConnection)
+  bibliographyDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  bibliographyIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  bibliographyLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  ByName(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheClient.CacheCommand
+  ClassServerName()  :string
+  CurrentAge(InterSystems.Data.CacheClient.CacheConnection)  :System.Nullable<long>
+  CurrentAge(InterSystems.Data.CacheClient.CacheConnection, InterSystems.Data.CacheTypes.CacheDate)  :System.Nullable<long>
+  DeleteId(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  ExistsId(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  Extent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheClient.CacheCommand
+  FirstNameIDXExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  FirstNameIDXExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  FirstNameIndexExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  FirstNameIndexExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
#  GetAllPropertiesInfoColn()  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Author
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :czuLibrary.Author
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Author
+  KillExtent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  LastNameIDXExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  LastNameIDXExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Author
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Author
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[])  :czuLibrary.Author
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Author
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Author
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Author
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Author
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Author
+  Populate(InterSystems.Data.CacheClient.CacheConnection)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*, System.Nullable<long>)  :System.Nullable<long>
+  PopulateSerial(InterSystems.Data.CacheClient.CacheConnection)  :string
+  SysComposeOid(InterSystems.Data.CacheClient.CacheConnection, string)  :void
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysIsA(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<long>
+  SysObjectIsNull(InterSystems.Data.CacheClient.CacheConnection, byte[])  :System.Nullable<bool>
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string, System.Nullable<bool>)  :InterSystems.Data.CacheTypes.CacheStatus
+  ValidateRuntimeMetaInfo(InterSystems.Data.CacheClient.CacheConnection)  :void
+  ValidateRuntimeMetaInfoWrapper(InterSystems.Data.CacheClient.CacheConnection)  :void

«property»
+  bibliography()  :string
```

Figure 24. Object binding *Author* class for .NET. [Souce: own]



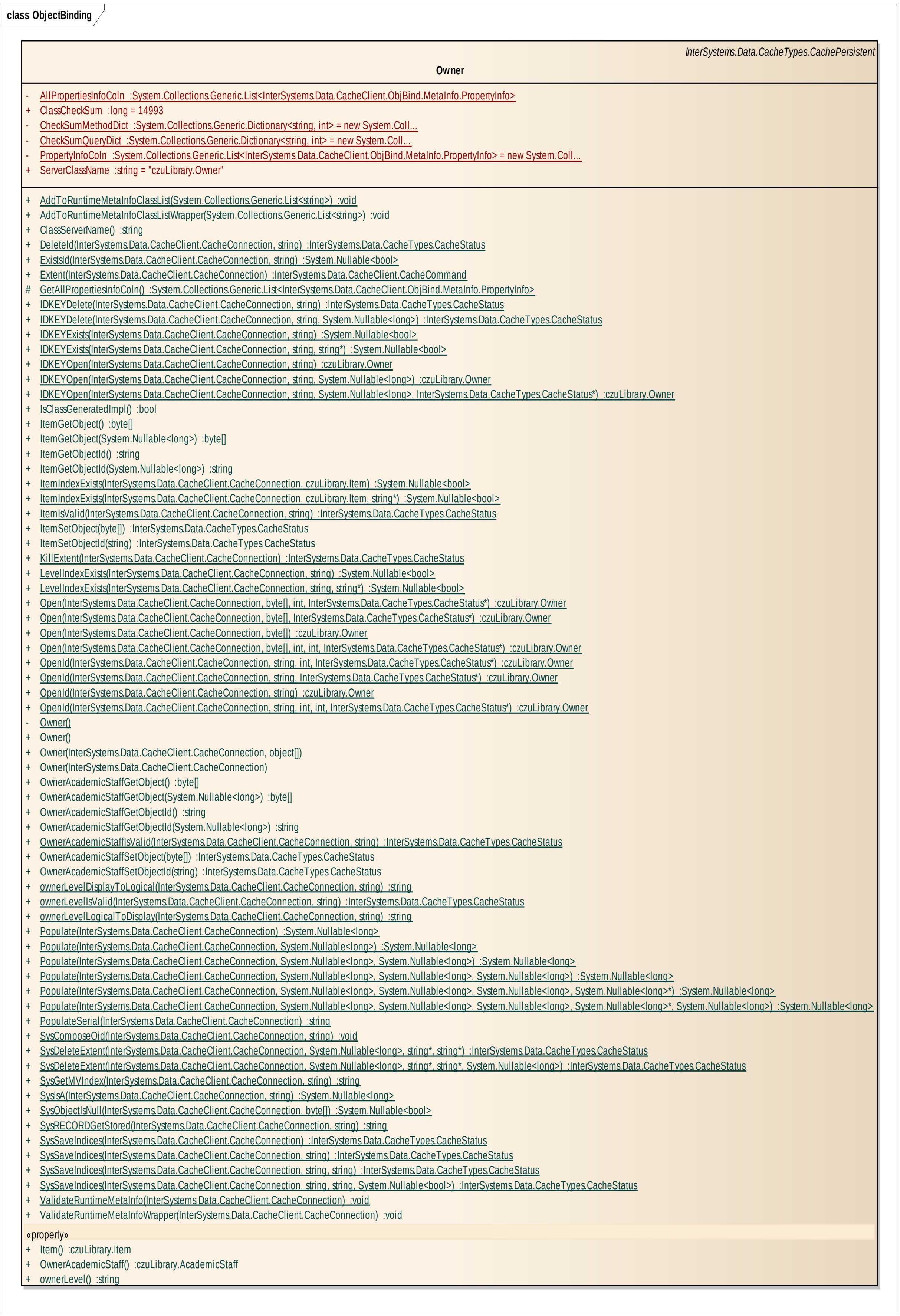
```
class ObjectBinding

                                              InterSystems.Data.CacheTypes.CachePersistent
                                Category

-  AllPropertiesInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  ClassCheckSum  :long = 14993
-  CheckSumMethodDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  CheckSumQueryDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  PropertyInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo> = new System.Coll...
+  ServerClassName  :string = "czuLibrary.Cat...

+  AddToRuntimeMetaInfoClassList(System.Collections.Generic.List<string>)  :void
+  AddToRuntimeMetaInfoClassListWrapper(System.Collections.Generic.List<string>)  :void
-  Category()
+  Category()
+  Category(InterSystems.Data.CacheClient.CacheConnection, object[])
+  Category(InterSystems.Data.CacheClient.CacheConnection)
+  ClassServerName()  :string
+  DeleteId(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  ExistsId(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  Extent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheClient.CacheCommand
#  GetAllPropertiesInfoColn()  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Category
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :czuLibrary.Category
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Category
+  IsClassGeneratedImpl()  :bool
+  KillExtent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  NameDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  NameIndexExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  NameIndexExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  NameIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  NameLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Category
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Category
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[])  :czuLibrary.Category
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Category
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Category
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Category
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Category
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Category
+  Populate(InterSystems.Data.CacheClient.CacheConnection)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*, System.Nullable<long>)  :System.Nullable<long>
+  PopulateSerial(InterSystems.Data.CacheClient.CacheConnection)  :string
+  SysComposeOid(InterSystems.Data.CacheClient.CacheConnection, string)  :void
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysIsA(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<long>
+  SysObjectIsNull(InterSystems.Data.CacheClient.CacheConnection, byte[])  :System.Nullable<bool>
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string, System.Nullable<bool>)  :InterSystems.Data.CacheTypes.CacheStatus
+  ValidateRuntimeMetaInfo(InterSystems.Data.CacheClient.CacheConnection)  :void
+  ValidateRuntimeMetaInfoWrapper(InterSystems.Data.CacheClient.CacheConnection)  :void

«property»
+  Name()  :string
```

Figure 25. Object binding *Category* class for .NET. [Souce: own]



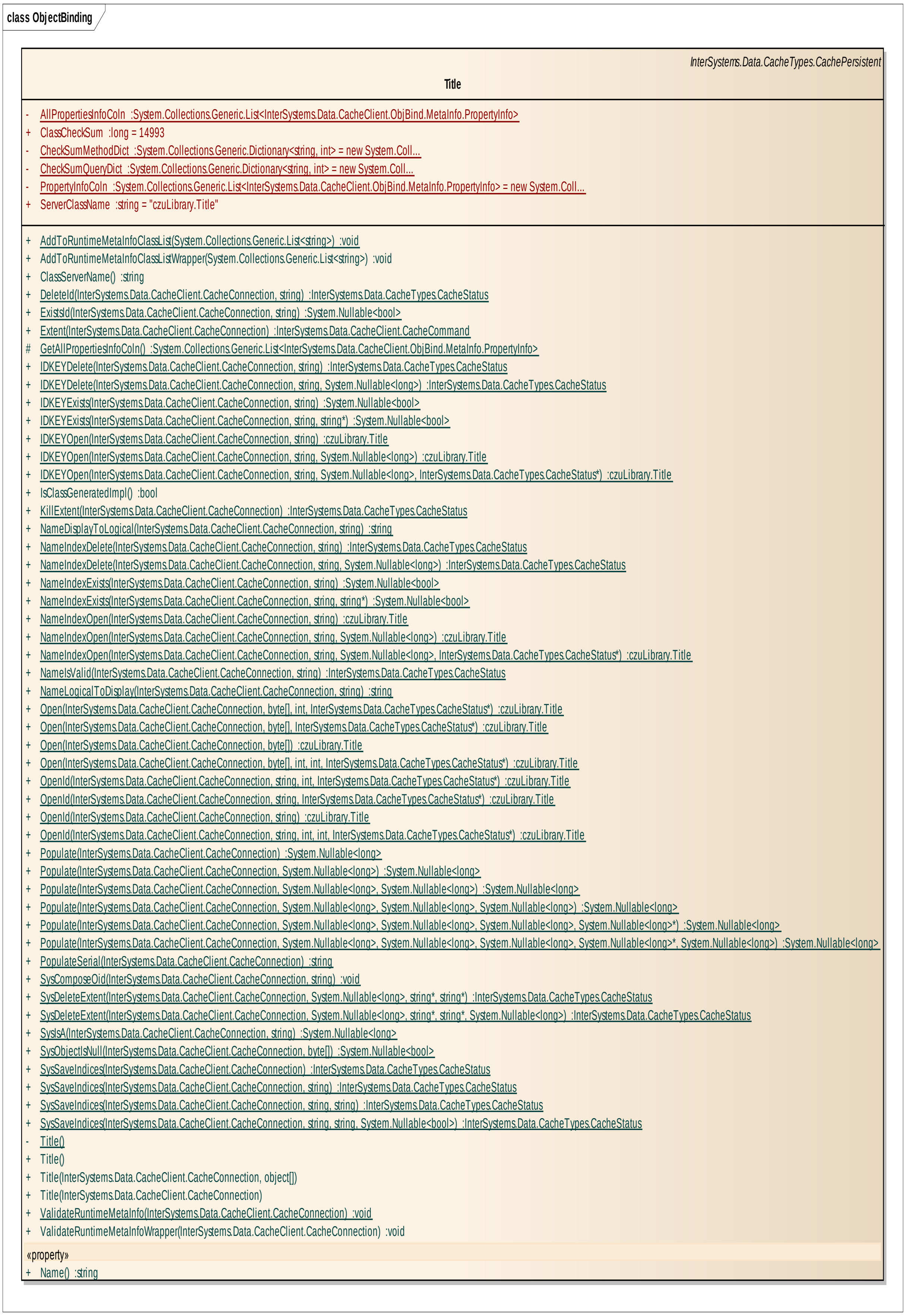
```
class ObjectBinding

                                                    InterSystems.Data.CacheTypes.CachePersistent
Item

-  AllPropertiesInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  ClassCheckSum  :long = 14993
-  CheckSumMethodDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  CheckSumQueryDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  PropertyInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo> = new System.Coll...
+  ServerClassName  :string = "czuLibrary.Item"

+  AddToRuntimeMetaInfoClassList(System.Collections.Generic.List<string>)  :void
+  AddToRuntimeMetaInfoClassListWrapper(System.Collections.Generic.List<string>)  :void
+  AuthorGetObject()  :byte[]
+  AuthorGetObject(System.Nullable<long>)  :byte[]
+  AuthorGetObjectId()  :string
+  AuthorGetObjectId(System.Nullable<long>)  :string
+  AuthorIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  AuthorSetObject(byte[])  :InterSystems.Data.CacheTypes.CacheStatus
+  AuthorSetObjectId(string)  :InterSystems.Data.CacheTypes.CacheStatus
+  CategoryGetObject()  :byte[]
+  CategoryGetObject(System.Nullable<long>)  :byte[]
+  CategoryGetObjectId()  :string
+  CategoryGetObjectId(System.Nullable<long>)  :string
+  CategoryIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  CategorySetObject(byte[])  :InterSystems.Data.CacheTypes.CacheStatus
+  CategorySetObjectId(string)  :InterSystems.Data.CacheTypes.CacheStatus
+  ClassServerName()  :string
+  DatePublishedDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheDate
+  DatePublishedIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  DatePublishedLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, InterSystems.Data.CacheTypes.CacheDate)  :string
+  DeleteId(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  ExistsId(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  Extent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheClient.CacheCommand
#  GetAllPropertiesInfoColn()  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Item
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :czuLibrary.Item
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Item
+  IsClassGeneratedImpl()  :bool
+  ISNBDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  ISNBIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  ISNBLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
-  Item()
+  Item()
+  Item(InterSystems.Data.CacheClient.CacheConnection, object[])
+  Item(InterSystems.Data.CacheClient.CacheConnection)
+  KillExtent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  NameDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  NameIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  NameLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Item
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Item
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[])  :czuLibrary.Item
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Item
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Item
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Item
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Item
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Item
+  Populate(InterSystems.Data.CacheClient.CacheConnection)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*, System.Nullable<long>)  :System.Nullable<long>
+  PopulateSerial(InterSystems.Data.CacheClient.CacheConnection)  :string
+  PublisherGetObject()  :byte[]
+  PublisherGetObject(System.Nullable<long>)  :byte[]
+  PublisherGetObjectId()  :string
+  PublisherGetObjectId(System.Nullable<long>)  :string
+  PublisherIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  PublisherSetObject(byte[])  :InterSystems.Data.CacheTypes.CacheStatus
+  PublisherSetObjectId(string)  :InterSystems.Data.CacheTypes.CacheStatus
+  StatusDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  StatusIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  StatusLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  SysComposeOid(InterSystems.Data.CacheClient.CacheConnection, string)  :void
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysIsA(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<long>
+  SysObjectIsNull(InterSystems.Data.CacheClient.CacheConnection, byte[])  :System.Nullable<bool>
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string, System.Nullable<bool>)  :InterSystems.Data.CacheTypes.CacheStatus
+  ValidateRuntimeMetaInfo(InterSystems.Data.CacheClient.CacheConnection)  :void
+  ValidateRuntimeMetaInfoWrapper(InterSystems.Data.CacheClient.CacheConnection)  :void

«property»
+  Author()  :czuLibrary.Author
+  Category()  :czuLibrary.Category
+  DatePublished()  :InterSystems.Data.CacheTypes.CacheDate
+  ISNB()  :string
+  Name()  :string
+  Publisher()  :czuLibrary.Publisher
+  Status()  :string
```

Figure 26. Object binding *Item* class for .NET. [Souce: own]



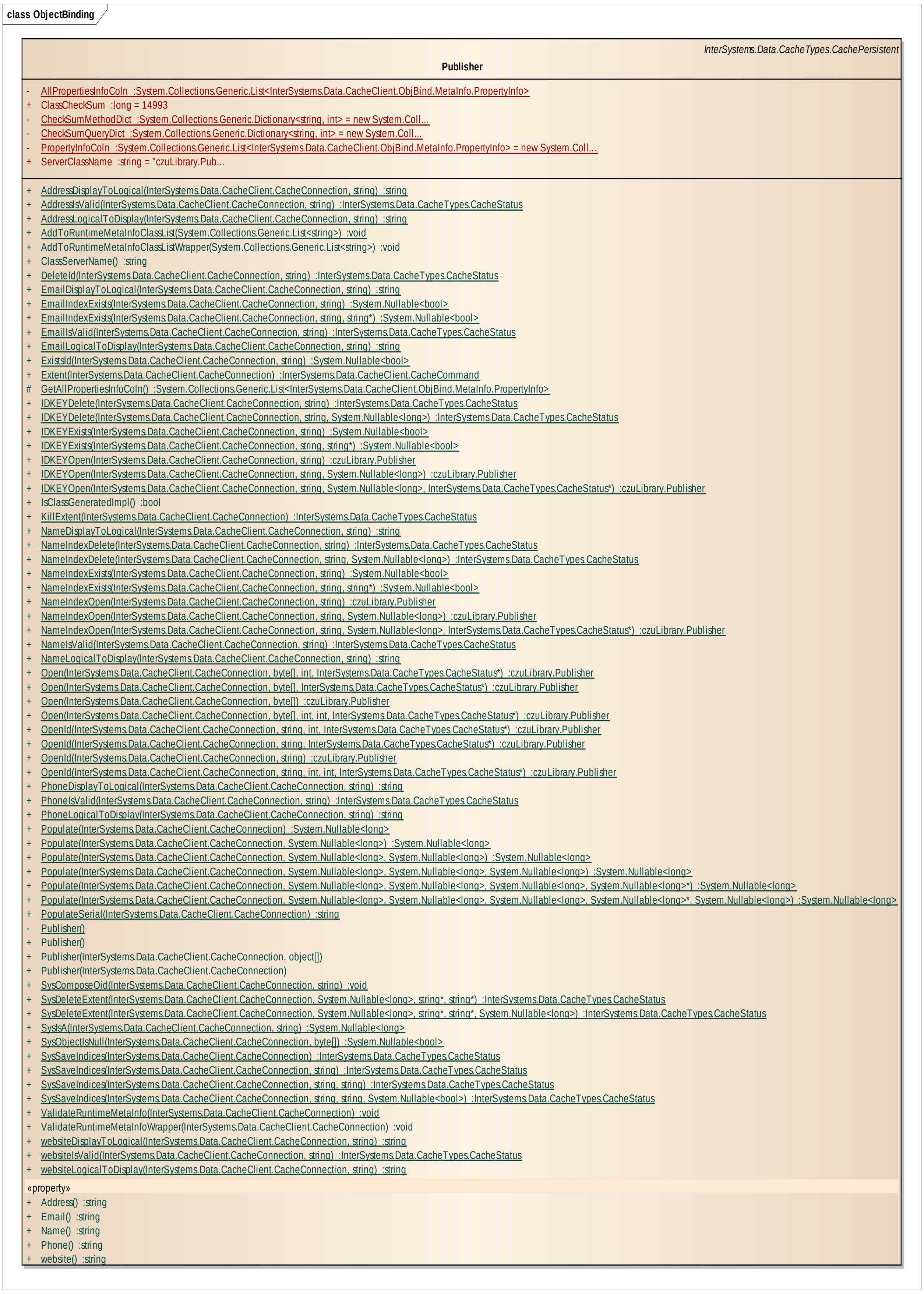
```
class ObjectBinding

                                   InterSystems.Data.CacheTypes.CachePersistent
Owner

-  AllPropertiesInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  ClassCheckSum  :long = 14993
-  CheckSumMethodDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  CheckSumQueryDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  PropertyInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo> = new System.Coll...
+  ServerClassName  :string = "czuLibrary.Owner"

+  AddToRuntimeMetaInfoClassList(System.Collections.Generic.List<string>)  :void
+  AddToRuntimeMetaInfoClassListWrapper(System.Collections.Generic.List<string>)  :void
+  ClassServerName()  :string
+  DeleteId(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  ExistsId(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  Extent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheClient.CacheCommand
#  GetAllPropertiesInfoColn()  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Owner
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :czuLibrary.Owner
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Owner
+  IsClassGeneratedImpl()  :bool
+  ItemGetObject()  :byte[]
+  ItemGetObject(System.Nullable<long>)  :byte[]
+  ItemGetObjectId()  :string
+  ItemGetObjectId(System.Nullable<long>)  :string
+  ItemIndexExists(InterSystems.Data.CacheClient.CacheConnection, czuLibrary.Item)  :System.Nullable<bool>
+  ItemIndexExists(InterSystems.Data.CacheClient.CacheConnection, czuLibrary.Item, string*)  :System.Nullable<bool>
+  ItemIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  ItemSetObject(byte[])  :InterSystems.Data.CacheTypes.CacheStatus
+  ItemSetObjectId(string)  :InterSystems.Data.CacheTypes.CacheStatus
+  KillExtent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  LevelIndexExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  LevelIndexExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Owner
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Owner
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[])  :czuLibrary.Owner
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Owner
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Owner
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Owner
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Owner
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Owner
-  Owner()
+  Owner()
+  Owner(InterSystems.Data.CacheClient.CacheConnection, object[])
+  Owner(InterSystems.Data.CacheClient.CacheConnection)
+  OwnerAcademicStaffGetObject()  :byte[]
+  OwnerAcademicStaffGetObject(System.Nullable<long>)  :byte[]
+  OwnerAcademicStaffGetObjectId()  :string
+  OwnerAcademicStaffGetObjectId(System.Nullable<long>)  :string
+  OwnerAcademicStaffIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  OwnerAcademicStaffSetObject(byte[])  :InterSystems.Data.CacheTypes.CacheStatus
+  OwnerAcademicStaffSetObjectId(string)  :InterSystems.Data.CacheTypes.CacheStatus
+  ownerLevelDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  ownerLevelIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  ownerLevelLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  Populate(InterSystems.Data.CacheClient.CacheConnection)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*, System.Nullable<long>)  :System.Nullable<long>
+  PopulateSerial(InterSystems.Data.CacheClient.CacheConnection)  :string
+  SysComposeOid(InterSystems.Data.CacheClient.CacheConnection, string)  :void
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysGetMVIndex(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  SysIsA(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<long>
+  SysObjectIsNull(InterSystems.Data.CacheClient.CacheConnection, byte[])  :System.Nullable<bool>
+  SysRECORDGetStored(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string, System.Nullable<bool>)  :InterSystems.Data.CacheTypes.CacheStatus
+  ValidateRuntimeMetaInfo(InterSystems.Data.CacheClient.CacheConnection)  :void
+  ValidateRuntimeMetaInfoWrapper(InterSystems.Data.CacheClient.CacheConnection)  :void

«property»
+  Item()  :czuLibrary.Item
+  OwnerAcademicStaff()  :czuLibrary.AcademicStaff
+  ownerLevel()  :string
```

Figure 27. Object binding Owner class for .NET. [Souce: own]



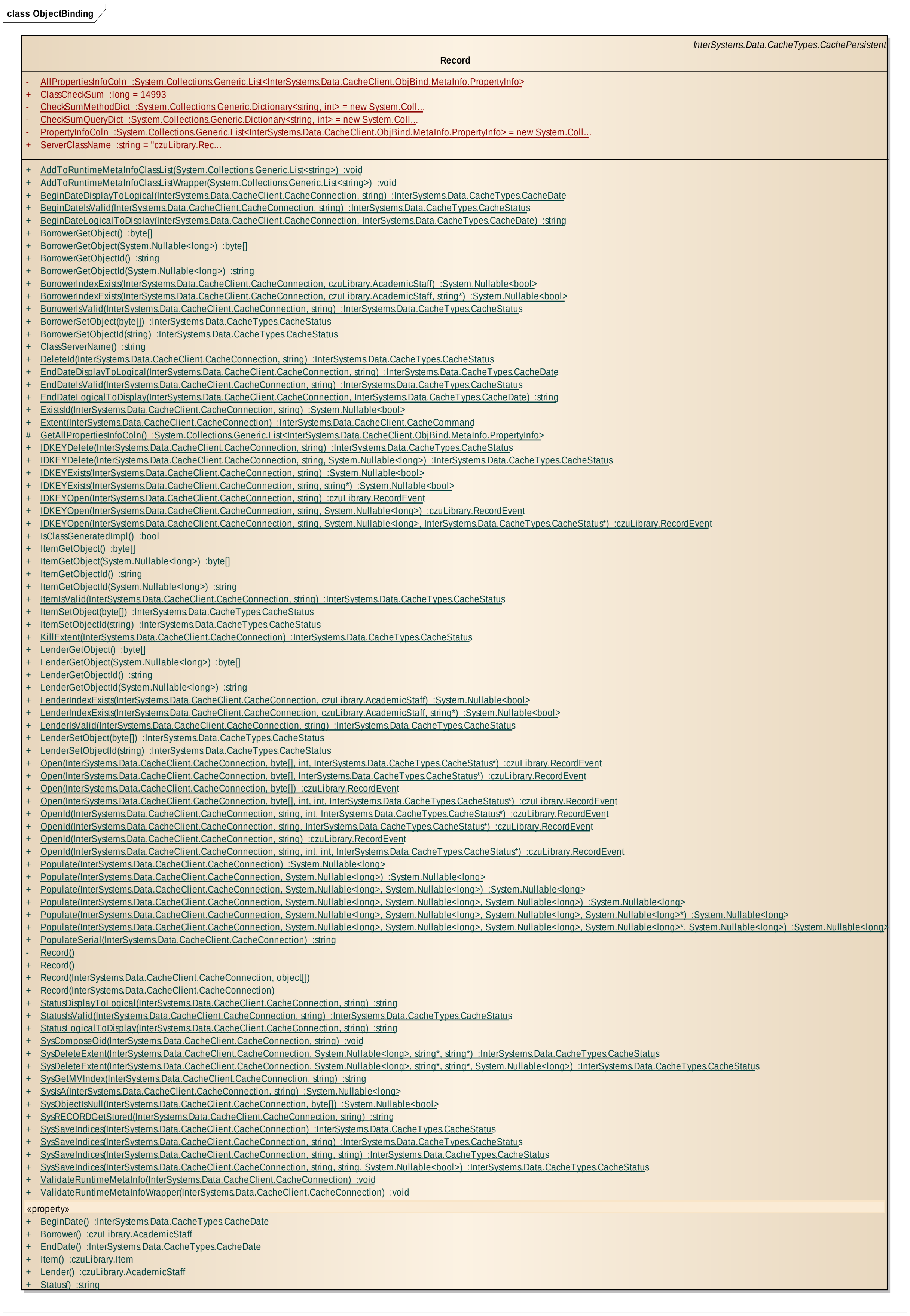
```
class ObjectBinding

                                         InterSystems.Data.CacheTypes.CachePersistent
                                   Title

-  AllPropertiesInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  ClassCheckSum  :long = 14993
-  CheckSumMethodDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  CheckSumQueryDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  PropertyInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo> = new System.Coll...
+  ServerClassName  :string = "czuLibrary.Title"

+  AddToRuntimeMetaInfoClassList(System.Collections.Generic.List<string>)  :void
+  AddToRuntimeMetaInfoClassListWrapper(System.Collections.Generic.List<string>)  :void
+  ClassServerName()  :string
+  DeleteId(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  ExistsId(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  Extent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheClient.CacheCommand
#  GetAllPropertiesInfoColn()  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Title
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :czuLibrary.Title
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Title
+  IsClassGeneratedImpl()  :bool
+  KillExtent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  NameDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  NameIndexDelete(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  NameIndexDelete(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  NameIndexExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  NameIndexExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  NameIndexOpen(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Title
+  NameIndexOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :czuLibrary.Title
+  NameIndexOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Title
+  NameIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  NameLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Title
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Title
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[])  :czuLibrary.Title
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Title
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Title
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Title
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Title
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Title
+  Populate(InterSystems.Data.CacheClient.CacheConnection)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*, System.Nullable<long>)  :System.Nullable<long>
+  PopulateSerial(InterSystems.Data.CacheClient.CacheConnection)  :string
+  SysComposeOid(InterSystems.Data.CacheClient.CacheConnection, string)  :void
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysIsA(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<long>
+  SysObjectIsNull(InterSystems.Data.CacheClient.CacheConnection, byte[])  :System.Nullable<bool>
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string, System.Nullable<bool>)  :InterSystems.Data.CacheTypes.CacheStatus
-  Title()
+  Title()
+  Title(InterSystems.Data.CacheClient.CacheConnection, object[])
+  Title(InterSystems.Data.CacheClient.CacheConnection)
+  ValidateRuntimeMetaInfo(InterSystems.Data.CacheClient.CacheConnection)  :void
+  ValidateRuntimeMetaInfoWrapper(InterSystems.Data.CacheClient.CacheConnection)  :void

«property»
+  Name()  :string
```

Figure 28. Object binding Title class for .NET. [Souce: own]



```
class ObjectBinding

InterSystems.Data.CacheTypes.CachePersistent
Publisher

-  AllPropertiesInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  ClassCheckSum  :long = 14993
-  CheckSumMethodDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  CheckSumQueryDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  PropertyInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo> = new System.Coll...
+  ServerClassName  :string = "czuLibrary.Pub...

+  AddressDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  AddressIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  AddressLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  AddToRuntimeMetaInfoClassList(System.Collections.Generic.List<string>)  :void
+  AddToRuntimeMetaInfoClassListWrapper(System.Collections.Generic.List<string>)  :void
+  ClassServerName()  :string
+  DeleteId(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  EmailDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  EmailIndexExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  EmailIndexExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  EmailIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  EmailLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  ExistsId(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  Extent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheClient.CacheCommand
#  GetAllPropertiesInfoColn()  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Publisher
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :czuLibrary.Publisher
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Publisher
+  IsClassGeneratedImpl()  :bool
+  KillExtent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  NameDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  NameIndexDelete(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  NameIndexDelete(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  NameIndexExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  NameIndexExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  NameIndexOpen(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Publisher
+  NameIndexOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :czuLibrary.Publisher
+  NameIndexOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Publisher
+  NameIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  NameLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Publisher
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Publisher
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[])  :czuLibrary.Publisher
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Publisher
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Publisher
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Publisher
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.Publisher
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.Publisher
+  PhoneDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  PhoneIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  PhoneLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  Populate(InterSystems.Data.CacheClient.CacheConnection)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*, System.Nullable<long>)  :System.Nullable<long>
+  PopulateSerial(InterSystems.Data.CacheClient.CacheConnection)  :string
-  Publisher()
+  Publisher()
+  Publisher(InterSystems.Data.CacheClient.CacheConnection, object[])
+  Publisher(InterSystems.Data.CacheClient.CacheConnection)
+  SysComposeOid(InterSystems.Data.CacheClient.CacheConnection, string)  :void
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysIsA(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<long>
+  SysObjectIsNull(InterSystems.Data.CacheClient.CacheConnection, byte[])  :System.Nullable<bool>
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string, System.Nullable<bool>)  :InterSystems.Data.CacheTypes.CacheStatus
+  ValidateRuntimeMetaInfo(InterSystems.Data.CacheClient.CacheConnection)  :void
+  ValidateRuntimeMetaInfoWrapper(InterSystems.Data.CacheClient.CacheConnection)  :void
+  websiteDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  websiteIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  websiteLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string

«property»
+  Address()  :string
+  Email()  :string
+  Name()  :string
+  Phone()  :string
+  website()  :string
```

Figure 29. Object binding Publisher class for .NET. [Souce: own]



```
class ObjectBinding

                                                    InterSystems.Data.CacheTypes.CachePersistent
                                  Record

-  AllPropertiesInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  ClassCheckSum  :long = 14993
-  CheckSumMethodDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  CheckSumQueryDict  :System.Collections.Generic.Dictionary<string, int> = new System.Coll...
-  PropertyInfoColn  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo> = new System.Coll...
+  ServerClassName  :string = "czuLibrary.Rec...

+  AddToRuntimeMetaInfoClassList(System.Collections.Generic.List<string>)  :void
+  AddToRuntimeMetaInfoClassListWrapper(System.Collections.Generic.List<string>)  :void
+  BeginDateDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheDate
+  BeginDateIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  BeginDateLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, InterSystems.Data.CacheTypes.CacheDate)  :string
+  BorrowerGetObject()  :byte[]
+  BorrowerGetObject(System.Nullable<long>)  :byte[]
+  BorrowerGetObjectId()  :string
+  BorrowerGetObjectId(System.Nullable<long>)  :string
+  BorrowerIndexExists(InterSystems.Data.CacheClient.CacheConnection, czuLibrary.AcademicStaff)  :System.Nullable<bool>
+  BorrowerIndexExists(InterSystems.Data.CacheClient.CacheConnection, czuLibrary.AcademicStaff, string*)  :System.Nullable<bool>
+  BorrowerIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  BorrowerSetObject(byte[])  :InterSystems.Data.CacheTypes.CacheStatus
+  BorrowerSetObjectId(string)  :InterSystems.Data.CacheTypes.CacheStatus
+  ClassServerName()  :string
+  DeleteId(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  EndDateDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheDate
+  EndDateIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  EndDateLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, InterSystems.Data.CacheTypes.CacheDate)  :string
+  ExistsId(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  Extent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheClient.CacheCommand
#  GetAllPropertiesInfoColn()  :System.Collections.Generic.List<InterSystems.Data.CacheClient.ObjBind.MetaInfo.PropertyInfo>
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYDelete(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<bool>
+  IDKEYExists(InterSystems.Data.CacheClient.CacheConnection, string, string*)  :System.Nullable<bool>
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.RecordEvent
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>)  :czuLibrary.RecordEvent
+  IDKEYOpen(InterSystems.Data.CacheClient.CacheConnection, string, System.Nullable<long>, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.RecordEvent
+  IsClassGeneratedImpl()  :bool
+  ItemGetObject()  :byte[]
+  ItemGetObject(System.Nullable<long>)  :byte[]
+  ItemGetObjectId()  :string
+  ItemGetObjectId(System.Nullable<long>)  :string
+  ItemIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  ItemSetObject(byte[])  :InterSystems.Data.CacheTypes.CacheStatus
+  ItemSetObjectId(string)  :InterSystems.Data.CacheTypes.CacheStatus
+  KillExtent(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  LenderGetObject()  :byte[]
+  LenderGetObject(System.Nullable<long>)  :byte[]
+  LenderGetObjectId()  :string
+  LenderGetObjectId(System.Nullable<long>)  :string
+  LenderIndexExists(InterSystems.Data.CacheClient.CacheConnection, czuLibrary.AcademicStaff)  :System.Nullable<bool>
+  LenderIndexExists(InterSystems.Data.CacheClient.CacheConnection, czuLibrary.AcademicStaff, string*)  :System.Nullable<bool>
+  LenderIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  LenderSetObject(byte[])  :InterSystems.Data.CacheTypes.CacheStatus
+  LenderSetObjectId(string)  :InterSystems.Data.CacheTypes.CacheStatus
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.RecordEvent
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.RecordEvent
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[])  :czuLibrary.RecordEvent
+  Open(InterSystems.Data.CacheClient.CacheConnection, byte[], int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.RecordEvent
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.RecordEvent
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.RecordEvent
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string)  :czuLibrary.RecordEvent
+  OpenId(InterSystems.Data.CacheClient.CacheConnection, string, int, int, InterSystems.Data.CacheTypes.CacheStatus*)  :czuLibrary.RecordEvent
+  Populate(InterSystems.Data.CacheClient.CacheConnection)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*)  :System.Nullable<long>
+  Populate(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>, System.Nullable<long>*, System.Nullable<long>)  :System.Nullable<long>
+  PopulateSerial(InterSystems.Data.CacheClient.CacheConnection)  :string
-  Record()
+  Record()
+  Record(InterSystems.Data.CacheClient.CacheConnection, object[])
+  Record(InterSystems.Data.CacheClient.CacheConnection)
+  StatusDisplayToLogical(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  StatusIsValid(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  StatusLogicalToDisplay(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  SysComposeOid(InterSystems.Data.CacheClient.CacheConnection, string)  :void
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysDeleteExtent(InterSystems.Data.CacheClient.CacheConnection, System.Nullable<long>, string*, string*, System.Nullable<long>)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysGetMVIndex(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  SysIsA(InterSystems.Data.CacheClient.CacheConnection, string)  :System.Nullable<long>
+  SysObjectIsNull(InterSystems.Data.CacheClient.CacheConnection, byte[])  :System.Nullable<bool>
+  SysRECORDGetStored(InterSystems.Data.CacheClient.CacheConnection, string)  :string
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string)  :InterSystems.Data.CacheTypes.CacheStatus
+  SysSaveIndices(InterSystems.Data.CacheClient.CacheConnection, string, string, System.Nullable<bool>)  :InterSystems.Data.CacheTypes.CacheStatus
+  ValidateRuntimeMetaInfo(InterSystems.Data.CacheClient.CacheConnection)  :void
+  ValidateRuntimeMetaInfoWrapper(InterSystems.Data.CacheClient.CacheConnection)  :void

«property»
+  BeginDate()  :InterSystems.Data.CacheTypes.CacheDate
+  Borrower()  :czuLibrary.AcademicStaff
+  EndDate()  :InterSystems.Data.CacheTypes.CacheDate
+  Item()  :czuLibrary.Item
+  Lender()  :czuLibrary.AcademicStaff
+  Status()  :string
```

Figure 30. Object binding Record class for .NET. [Souce: own]

10. Using Object binding classes in .NET

After Caché Wizard has generated object binding classes for .NET, then we have to reference InterSystems.Data.CacheClient.dll library from Caché into our project. It is located in the folder „\InterSystems\TryCache\dev\dotnet\bin\" under Caché database home path.

After the reference has been added to the application, we can connect to Caché object database server by below shown *CacheConnection()* function.

```
/// <summary>
/// connection to InterSystem Cache database
/// </summary>
/// <returns>bool</returns>
public static bool CacheConnection()
{
   bool okay = true;
   // Define a hardwired default connection string.
   string ConnStr =
     "Server = " + _Server + ";"
     + " Port = " + _Port + ";"
     + " Namespace = " + _NameSpaces + ";"
     + " User ID = " + _UserName + ";"
     + " Password = " + _Password + ";";
   try {  // Create a connection object, then try to open
      // the connection using the default connection string.
      CacheConnect = new CacheConnection();
      CacheConnect.ConnectionString = ConnStr;
      CacheConnect.Open();
   } catch {
      okay = false;
   }
   return okay;
} // end CacheConnection()
```

Here is an example of accessing first item from Item object then change name and save changes.

```
czuLibrary.Item   item = czuLibrary.Item.OpenId("1");
item.Name = "TestName";
GlobalClass.status = item.Save();
if (!GlobalClass.status.IsOK) {
  MessageBox.Show(GlobalClass.status.Message);
}
Else {
```

```
            MessageBox.Show("Item’s name is changed")
          }
```

The following is an example of using extended SQL language. This query selects details of authors.

```
select
  id,
  firstname,
  lastname,
  bibliography,
  homeaddress_city,
  homeaddress_street,
  homeaddress_zip,
  { fn convert(dob, sql_varchar) } as date_of_birth
from czulibrary.author;
```

The following embedded SQL example matches Home_City column values to the elements of the cities (Czech Republic cities) list:

```
set cities=$listbuild("prague","zlin")
&sql(declare citycursor cursor for
        select name,
                     home_city
        into :name,:
                   city
      from czulibrary.academicstaff
        where home_city %inlist :cities)
  &sql(open citycursor)
   new sqlcode,%rowcount,%rowid
   for { &sql(fetch citycursor)
        quit:sqlcode
        write !,"#",%rowcount," name=",name," city=",city,!
 }
   write !,"final fetch sqlcode: ",sqlcode
   &sql(close citycursor)
```

The $LISTGET functions in the following embedded SQL example both return „*Brown*”, the first element in the list:

```
set c=$listbuild("brown","red","green","blue")
&sql(select $listget(:c),$listget(:c,1)
into :d,:e
from czulibrary.person)
if sqlcode = 0 {
    write !,"error code",sqlcode }
else {
    write !,"the one-arg element returned is ",d
    write !,"the two-arg element returned is ",e }
```

11. Using framework WPF (Windows presentation framework) as GUI

*„Windows Presentation Foundation (WPF) is a next-generation presentation system for building Windows client applications with visually stunning user experiences. With WPF, you can create a wide range of both standalone and browser-hosted applications. " (*MSDN*, 2014)*

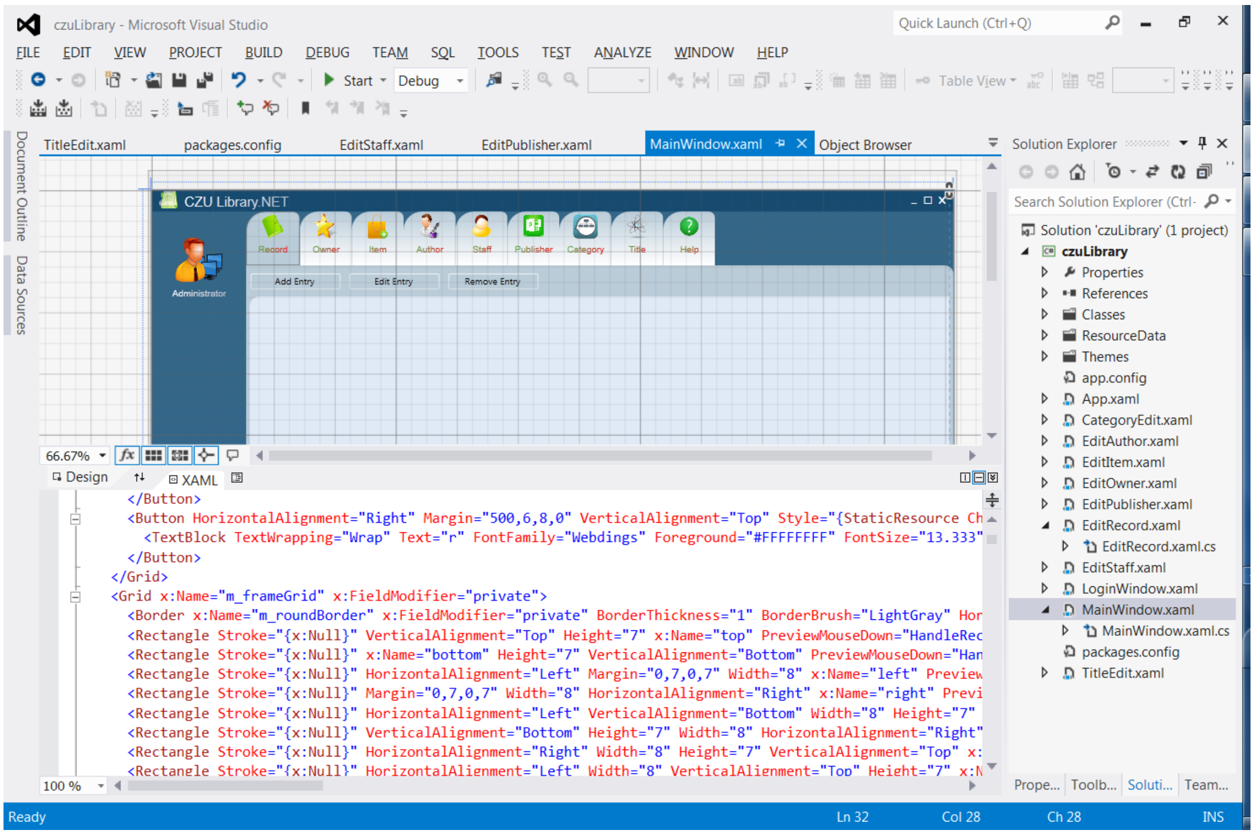
Windows Presentation Foundation framework is used to creating very beautiful and user friendly graphical interface for *czuLibrary.NET*.

Windows Presentation Foundation (WPF) is a comprehensive API- interface for creating desktop graphics programs with rich design and interactivity. In contrast to the older Windows Forms, WPF includes a new model for building custom applications (based on the powerful WPF infrastructure, based on DirectX).

This means you can use advanced graphic effects without paying for this performance, as it was in Windows Forms. In fact, even become available advanced tools such as support for video and three-dimensional content. Using these facilities (with a good graphic design tool), you can create eye-catching user interfaces and visual effects that were simply not possible in Windows Forms.

The main purpose of Windows Presentation Foundation (WPF), to help developers create compelling and effective user interfaces. Learn how WPF unified platform helps designers make active participants in creating user interfaces, and provides a common programming model for standalone and browser applications.

To build an application developed in this tutorial, you must install Microsoft. NET Framework Software Development Kit and Software (SDK) for Windows.



```
      </Button>
      <Button HorizontalAlignment="Right" Margin="500,6,8,0" VerticalAlignment="Top" Style="{StaticResource Ch
        <TextBlock TextWrapping="Wrap" Text="r" FontFamily="Webdings" Foreground="#FFFFFFFF" FontSize="13.333"
      </Button>
    </Grid>
    <Grid x:Name="m_frameGrid" x:FieldModifier="private">
      <Border x:Name="m_roundBorder" x:FieldModifier="private" BorderThickness="1" BorderBrush="LightGray" Hor
      <Rectangle Stroke="{x:Null}" VerticalAlignment="Top" Height="7" x:Name="top" PreviewMouseDown="HandleRec
      <Rectangle Stroke="{x:Null}" x:Name="bottom" Height="7" VerticalAlignment="Bottom" PreviewMouseDown="Han
      <Rectangle Stroke="{x:Null}" HorizontalAlignment="Left" Margin="0,7,0,7" Width="8" x:Name="left" Preview
      <Rectangle Stroke="{x:Null}" Margin="0,7,0,7" Width="8" HorizontalAlignment="Right" x:Name="right" Previ
      <Rectangle Stroke="{x:Null}" HorizontalAlignment="Left" VerticalAlignment="Bottom" Width="8" Height="7"
      <Rectangle Stroke="{x:Null}" VerticalAlignment="Bottom" Height="7" Width="8" HorizontalAlignment="Right"
      <Rectangle Stroke="{x:Null}" HorizontalAlignment="Right" Width="8" Height="7" VerticalAlignment="Top" x:
      <Rectangle Stroke="{x:Null}" HorizontalAlignment="Left" Width="8" VerticalAlignment="Top" Height="7" x:N
```

Figure 31. *czuLibrary.NET* application GUI using WPF. [Souce: own]

12. Discussions

12.1. SQL Query comparison of Relational and Object databases

czuLibrary,NET uses object oriented database as its database. But I have drawn also relational model for the application (see Figure 19).

In the Figure 32, Difference of object-oriented database model and relational database model of *czuLibrary.NET* are shown side by side, it is relational model on the left side, and on the other side is object model of the same application.

In this particular case, we can see that there are three foreign keys on each table of Record and Item, and two foreign keys on table *AcademicStaff*. On the contrary side has no foreign key, it means we do not join objects to query over the three objects or tables.

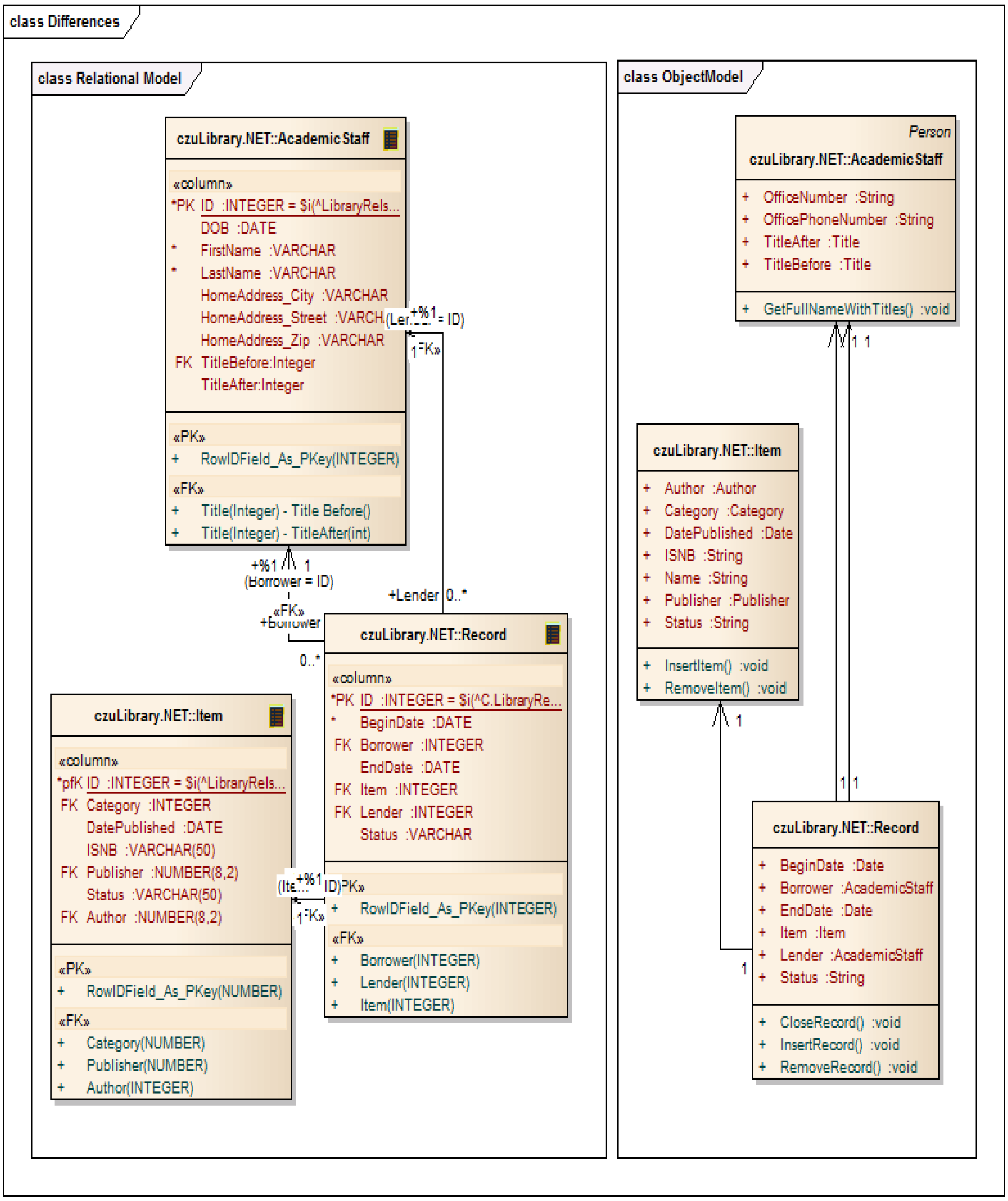
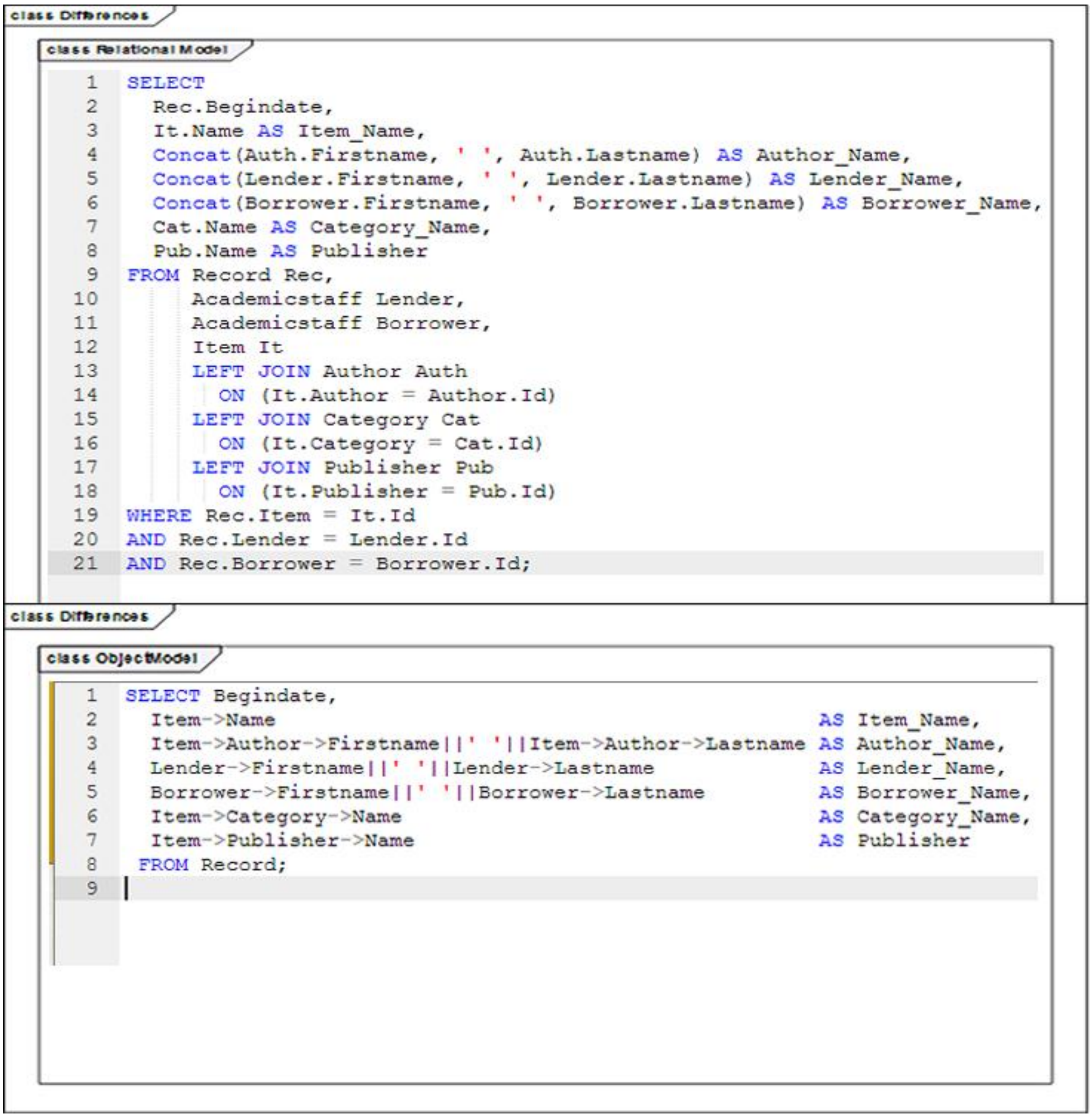


Figure 32. Differences of Relational and Object databases for *czuLibrary.NET*.

Figure 33 shows two different SQL query, on the top is for relational model, and on the bottom is for object model.

As you see there in Figure 33, to select item's name, borrower's name and lender's name, we have needed six different joins, but on contrary side, we do not need join objects. Joins are not required in Object-oriented database. It gives us to reach high-performance effort and we can easily select borrowing information with easy and high performance.



```
SELECT
  Rec.Begindate,
  It.Name AS Item_Name,
  Concat(Auth.Firstname, ' ', Auth.Lastname) AS Author_Name,
  Concat(Lender.Firstname, ' ', Lender.Lastname) AS Lender_Name,
  Concat(Borrower.Firstname, ' ', Borrower.Lastname) AS Borrower_Name,
  Cat.Name AS Category_Name,
  Pub.Name AS Publisher
FROM Record Rec,
     Academicstaff Lender,
     Academicstaff Borrower,
     Item It
     LEFT JOIN Author Auth
       ON (It.Author = Author.Id)
     LEFT JOIN Category Cat
       ON (It.Category = Cat.Id)
     LEFT JOIN Publisher Pub
       ON (It.Publisher = Pub.Id)
WHERE Rec.Item = It.Id
AND Rec.Lender = Lender.Id
AND Rec.Borrower = Borrower.Id;
```

```
SELECT Begindate,
  Item->Name                                         AS Item_Name,
  Item->Author->Firstname||' '||Item->Author->Lastname AS Author_Name,
  Lender->Firstname||' '||Lender->Lastname           AS Lender_Name,
  Borrower->Firstname||' '||Borrower->Lastname       AS Borrower_Name,
  Item->Category->Name                               AS Category_Name,
  Item->Publisher->Name                              AS Publisher
 FROM Record;
```

Figure 33. Query difference between Relational and Object databases.

Dispute this, I think the active use of object -oriented database system requires further research and practical study.

13. Conclusions

Object-oriented approach models the real world more completely than a relational approach. Complex objects are inefficient to model in a RDBMS. When application behavior is separated from database behavior, mapping between the relational and the object world has to be made.

For simple data models there are no need to use an ODBMS, there is no performance advantage, searching for requested data and optimizations of queries are supported by the products.

RDBMS has a dominating position on the market, their products have matured and contains all necessary tools. All major vendors use the SQL standard, which increases portability between different products. As long as no standard has been chosen for object-oriented database systems, they have a great disadvantage against the relational databases.

ODBMS and RDBMS will coexist and hold different niches of the market, since Commercial relational database systems have collected more than a decade of Business experience, are well established and do much better cope with the simple structured data they are designed for.

Object-oriented database management systems in general and today's implementation of InterSystems Caché in particular (Build: Caché v2013.1.2 (601)) can be very efficient and easy-to-use alternatives to the relational database manage systems. With InterSystems Caché, we can handle very big data easily and achieve high-performance database.

In the case of object-oriented features such as inheritance and polymorphism or a complex object graph, a relational mapping to the object model becomes very hard to define, due to the pronounced impedance mismatch between the OOP and SQL worlds. Relational DBMS do not alleviate this issue to a significant extent either.

There are also performance implications, which favor InterSystems Caché. Storing an object graph through an object-relational mapping is several orders of magnitude slower than using SQL, which, in turn, can actually be slower than directly persisting a complex object graph.

In the case of an evolving object model (the attributes and methods of types or type templates might change), a relational database needs a schema alteration, in order to accommodate the changes. This is not the case with Caché, whose database structure mirrors that of the application and modifications are performed seamlessly, on-the-fly.

The application developed (*czuLibrary.NET*) is a well-chosen example of using the strengths of OODBMS' in real-world scenarios. It manages the storage, visualization and querying of the Items of the user and allows tracking Item already stored, deleting existing Item, searching for. It benefits from object-oriented persistence features in terms of natural, straightforward development, as well as performance opportunities.

References

[1] KROHA, Petr. *Objects and databases*. New York: McGraw-Hill Book Co., c1993, xii, 244 p. ISBN 00-770-7790-3.

[2] MERUNKA, Vojtěch. *Database Systems Course*. CZU Press, 2004.

[3] BERTINO, Elisaistos. *0bject-oriented database systems - concepts and architectures*. Vyd. 1. London: Addison-Wesley, 1993, 260 s. ISBN 02-016-2439-7.

[4] DELOBEL, C. *Databases: from relational to object-oriented systems*. Vyd. 1. London: Thomson Publishing, 1995, 382 s. ISBN 18-503-2124-8.

[5] LING, W. *Deductive and object-oriented databases*. Vyd. 1. Berlin: Springer-Verlag, 1995, 558 s. ISBN 35-406-0608-4.

[6] MARK LINES, Scott Ambler. *Disciplined Agile Delivery a practitioner's guide to agile software delivery in the enterprise*. S.l.: IBM Press, 2012. ISBN 01-328-1013-1.

[7] JEFFERY, Keith. *Expert database systems*. San Diego: Academic Press, c1992, xiv, 254 p. A.P.I.C. studies in data processing, no. 39. ISBN 01-238-2255-6.

[8] MCLAUGHLIN, Brett. *Head first object-oriented analysis and design*. 1st ed. Beijing: O'Reilly, c2007, xxxiv, 600 s. ISBN 05-960-0867-8.

[9] CATTELL, R. *Object data management: object-oriented and extended relational database systems*. Reading, Mass.: Addison-Wesley, c1991, xvii, 318 p. ISBN 02-015-3092-9.

[10] EMBLEY, David W. *Object database development: concepts and principles*. Reading, Mass.: Addison-Wesley, c1998, xviii, 877 p. ISBN 02-012-5829-3.

[11] NORRIE, Moira C a Michael GROSSNIKLAUS. *Object databases: second international conference, ICOODB 2009, Zurich, Switzerland, July 1-3, 2009: revised papers*. New York: Springer, c2010, viii, 166 p. ISBN 978-364-2146-800.

[12] KIRSTEN, W. *Object-oriented application development using the Caché post relational database*. 2nd, rev. and updated Ed. New York: Springer, 2003, xiv, 390 p. ISBN 35-400-0960-4.

[13] PRABHU, C.S.R. *Object-oriented database systems: approaches and architectures*. 2nd edition (7th printing). New Delhi: Prentice-Hall of India, 2005. ISBN 81-203-2789-6.

[14] KHOSHAFIAN, Setrag. *Object-oriented databases*. New York: John Wiley, c1993, xix, 362 p. ISBN 04-715-7058-3.

[15] BROWN, Alan W. *Object-oriented databases and their applications to software engineering*. Vyd. 1. London: McGraw-Hill, 1991, 260 s. ISBN 0201530929.

[16] MARTIN, James a James J ODELL. *Object-oriented methods: a foundation*. UML ed., 2nd ed. Upper Saddle River, N.J.: Prentice Hall PTR, c1998, xix, 408 p. ISBN 01-390-5597-5.

[17] MACDONALD, Matthew. *Pro WPF 4.5 in VB*. New York: Distributed to the book trade worldwide by Springer, c2012, xl, 1063 p. ISBN 978-0-596-00699-0.

[18] LIBERTY, Jesse. *Programming C#*. 4th ed.; Sebastopol, CA: O'Reilly, c2005, xx, 644 p. ISBN 05-960-0699-3.

[19] AMBLER, Scott W. *The object primer: agile modeling-driven development with UML 2.0*. 3rd Ed. New York: Cambridge University Press, 2004, xxvi, 545 p. ISBN 05-215-4018-6.

[20] FOWLER, Martin. *UML distilled: a brief guide to the standard object modeling language*. 3rd ed. Boston: Addison-Wesley, c2004, xxx, 175 p. ISBN 03-211-9368-7.

[21] YOSIFOVICH, Pavel. *Windows Presentation Foundation 4.5 cookbook*. Birmingham: Packt Publishing, 2012, iv, 448 p. ISBN 978-1849686228.

[22] InterSystems Caché. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-03-20]. Dostupné z: http://en.wikipedia.org/wiki/InterSystems_Cach%C3%A9

[23] Object Databases. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-03-20]. Dostupné z: http://en.wikipedia.org/wiki/Object_database

[24] *InterSystems* [online]. [cit. 2014-03-20]. Dostupné z: http://docs.intersystems.com

[25] UML. *Uml.org* [online]. 2014 [cit. 2014-03-27]. Dostupné z: http://www.uml.org/

[26] MSDN WPF. *Windows presentation framework* [online]. [cit. 2014-03-23]. Dostupné z: http://msdn.microsoft.com/en-us/library/aa970268(v=vs.110).aspx

[27] Techopedia. [online]. [cit. 2014-03-25]. Dostupné z: http://www.techopedia.com/definition/12027/object-oriented-database-management-system-oodbms

[28] MILES, Russ. *Learning UML 2.0*. 1st ed. Sebastopol: O´Reilly, 2006, 269 s. ISBN 05-960-0982-8.

[29] FOWLER, Martin. *UML distilled: a brief guide to the standard object modeling language*. 3rd ed. Boston: Addison-Wesley, c2004, xxx, 175 p. ISBN 03-211-9368-7.

[30] RIDYARD, Susan J. *The royal saints of Anglo-Saxon England: a study of West Saxon and East Anglian cults*. Cambridge: Cambridge University Press, 1988. ISSN 11842369.

[31] Codd. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-03-29]. Dostupné z: http://en.wikipedia.org/wiki/Edgar_F._Codd

List of Figures