

C A C H É G O M P O N E N T S

CACHE OBJECT TECHNOLOGY

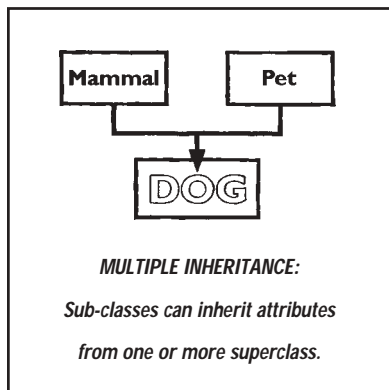


Caché objects are powered by Caché's post-relational database, the ideal match for high-performance applications developed with Web and object-oriented technologies. Using Caché object technology, programmers can create meaningful data structures that represent the real world and streamline the development process.

Caché Objects Enable Rapid Development

In order to speed application development, Caché object technology supports the concepts of inheritance, encapsulation, and polymorphism.

Inheritance is the ability to derive one class of objects from another. The new class (a subclass) will always have an "is a" relationship to the superclass. For example, a dog "is a" mammal, so the "Dog" class can inherit all the properties and methods of the "Mammal" class, as well as containing properties and methods that are unique to dogs. Multiple inheritance means a subclass can be derived from more than one superclass. A dog "is a" mammal, and "is a" pet, so the "Dog" class can inherit the attributes of both the "Mammal" class and the "Pet" class.



Encapsulation means that, as far as the application is concerned, classes can be viewed as a sort of "black box". No matter how complex, a class has a finite number of properties and methods. Once a class is defined, the application doesn't need to know its internal workings. The application deals only with the properties and methods of the class. This black box approach yields two important benefits.

A) Classes are modular.

Programmers can improve the internal workings of classes without affecting the rest of the application at all.

B) Classes are interoperable.

Classes can be shared between applications, because the interface (the properties and methods) remains constant.

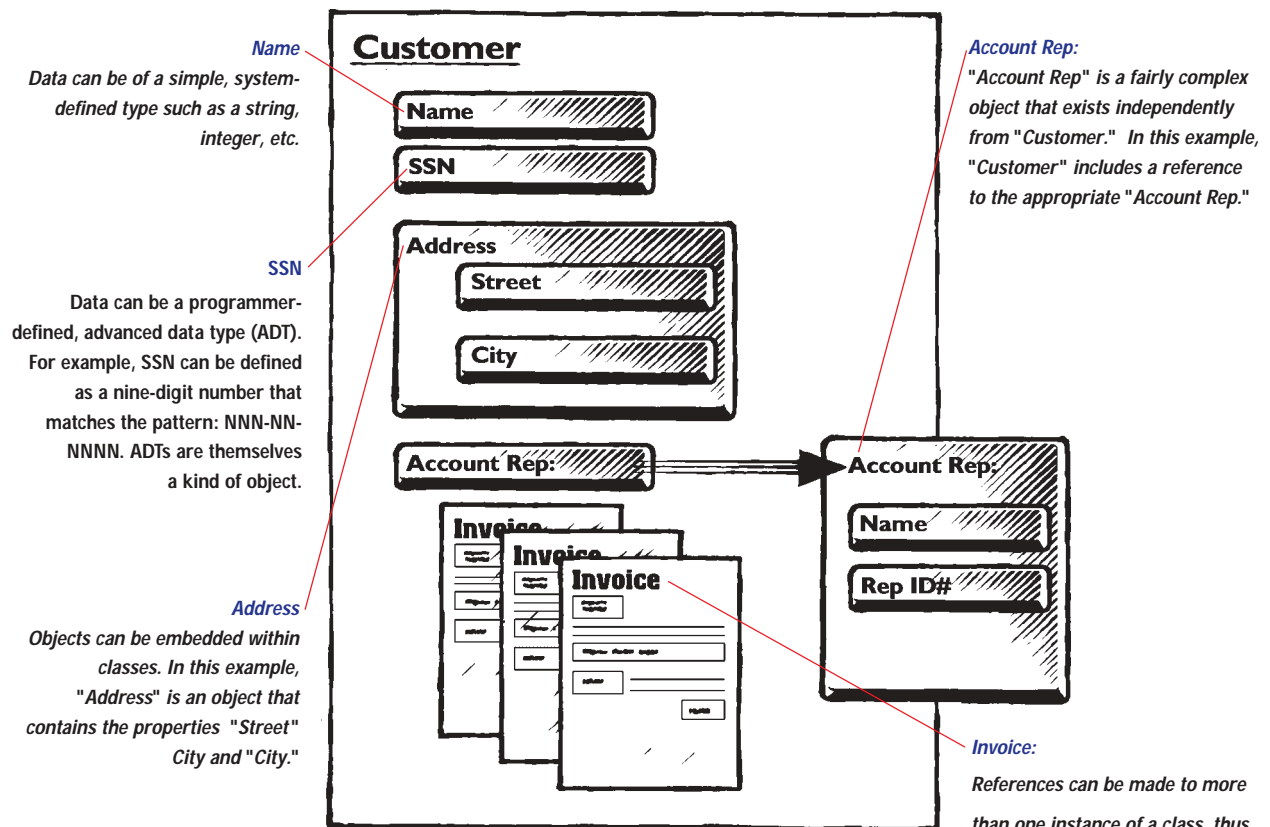
Polymorphism refers to the fact that methods used in multiple classes can share a common interface, even if the underlying implementation is different. For example, say an application uses several different classes: Letter, Mailing Label, and ID Badge, all of which contain a method called PrintAddress. The application doesn't need to contain special instructions about formatting an address for each kind of object. It merely includes a command that says something like "DO PrintAddress(objectID)". Polymorphism ensures that each object carries out the instruction in a manner appropriate for the class to which that object belongs.

CACHÉ OBJECT TECHNOLOGY

Caché Objects Model The Real World

Caché object technology attempts to describe the way that humans actually think about and use data. It does this by bundling together data and the code that controls how the data is used. In object parlance, the various pieces of data contained in a class** are called "properties" and the sections of code that describe how the data behaves are called "methods."

Caché object technology also promotes a naturalistic view of data by not restricting properties to simple, computer-centric data types. Classes may contain other classes, or references to other classes, which makes it easy to build useful and meaningful data models. Here's a simple example:



Even though "Customer" contains a large amount of information, an application can treat it as a single entity - an object.

** A "class" is a template - a generic description of the data. Each particular "instance" of a class is an object, and contains actual data.

Creating Caché Objects

Caché classes are rapidly created and edited with the Caché Studio. The Studio is an integrated development environment (IDE) where developers can perform all of their application development tasks. For data modeling, this includes specifying properties, coding and debugging object methods, and defining specialized data types. Caché's support for advanced object concepts – simple and multiple inheritance, embedded objects, references to objects, collections, relationships, and polymorphism – make the Caché Studio a powerful and productive environment for modeling data and business processes.



Importing/Exporting Data Models

The Caché Studio includes a wizard for the easy creation of Caché classes, but there are several other ways to input and export class definitions to and from the Studio.

The Caché RoseLink allows classes defined using Rational Software's popular Rose object modeling tool to be imported into Caché. Similarly, Caché class definitions can be exported to Rose for use within Rose's modeling environment.

Caché can also create objects from relational DDL files. The resulting object classes will be very simple: their properties will be single-valued system-defined data types that correspond to the relational table's fields, and their only methods will be those persistence methods required to move the data to and from the disk. However, thanks to Caché's Unified Data Architecture, even these simple classes are immediately available for use with object programming languages, and they may be used as building blocks to create more complex data models.

XML provides another way to transport class definitions from one application to another. Class definitions can be exported/imported as XML documents.

Caché Scripting Languages

Methods in Caché objects are coded using either (or both) Caché ObjectScript or Caché Basic. Both languages allow developers to use all of Caché's data access modes – Objects, SQL, and Multidimensional – within the same routine.

Integrating Caché Objects With Other Technologies

By virtue of Caché's Unified Data Architecture, all Caché classes are automatically accessible as relational tables via ODBC and JDBC. And by taking advantage of inheritance, Caché classes can easily be adapted for use with XML and object-oriented technologies.



Caché Server Pages

A class designated as a Caché Server Page automatically inherits all the necessary Web session management methods, plus the "OnPage()" method where developers can code the page content.

XML

Inheriting properties and methods from the %XML.Adaptor class (provided by InterSystems) enables a class to import/export XML data. Caché will automatically determine the mapping between Caché objects and XML documents, or developers may create custom maps.

COM

A single command within the Caché Studio projects Caché classes as COM classes for use with tools such as Visual Basic, Delphi, and any software compatible with the COM interface. Caché also includes a COM Gateway, which allows COM objects to be used by Caché applications.

C++

Similarly, a single command can create C++ projections of Caché classes.

Java

One command can project Caché classes as Java classes. Caché also provides a class library that allows Java programmers to access Caché objects in the Caché database.

EJBs

EJB projections can also be created with one click from within Caché Studio. Caché allows developers to take advantage of the speed of Bean-Managed Persistence, without having to do lots of tedious coding to map between Java classes and relational tables. Caché supports BEA's WebLogic application server.

InterSystems Corporation

World Headquarters
One Memorial Drive
Cambridge, MA 02142
Tel: 1.617.621.0600
Fax: 1.617.494.1631

www.InterSystems.com

