Browser Development Tool @net

by Ditmar Tybussek

Introducing @net—This in-house application creation tool won second prize in the InterSystems Caché Innovator Awards. Bewidata does not plan to turn it into a commercial product, but will allow others to use the tool. It particularly needs help text in English. That said, this is an exciting tool that facilitates development while using a browser interface to build applications with a browser presentation layer. Much of the text of this article was adapted from the submission for the competition. Having seen the application in action, I agree wholeheartedly with the judges at IDC!—Editor.

What is @net?

Bewidata developed @net for use as an internal development tool. It uses Caché and Weblink capabilities to connect an M database to a browser front end. The browser is used as the front end for both programming and user functions. This approach makes it possible for programmers to create applications while online anywhere in the world. The application can be executed instantly, without need of compilation—also worldwide.

The browser front end of @net creates M applications. It is designed to be multicultural and multilingual and to produce either simple or complex network applications which are data-intensive, object oriented, and dynamic. The programming approach facilitated by @net works well both for providing a new front end for traditional applications and for newer applications such as e-commerce.

Why Use Browser Technology?

The Internet and local intranets are being used today by an ever increasing number of companies and private users. It has been recognized that internet technology is less expensive than most earlier types of online connections. Since the internet involves a world-wide network, the importance of international applications is constantly increasing. Large, international software suppliers can benefit, for example, when their applications can be implemented in various languages by numerous users, all of whom have widely diversified tasks in mind. Just from observing the current development in the European marketplace, the advantages a multilingual and multicultural application can provide become readily apparent. Users in different countries do not need to settle on a common language. Each user works in the language of his or her choice. Users can feel at home with programs much more quickly if they can use the language, methods of writing currency amounts, the date and the time formats with which they are already familiar.

@net enables flexible distribution of resources. It might be reasonable to have your server in Canada, your product development department in the USA, data entry in India and the statistical analysis done by a company in Germany—all online, in multiple languages.

It may be that the statistics department needs a new data field. An email to the developer in the United States is all it takes. This change request can be carried out there, with the help of @net. When the data entry people in India call up that HTML page, this new data field is automatically and immediately activated. Automatically generated help functions within @net make it easy for all users to quickly benefit from new features.

Intranet programming by systems suppliers for third parties is also simplified with @net. In addition to the support functions offered by inheritance during the development process, objects developed using @net provide the ability to distribute systems or sub-systems easily. This makes it possible to create, distribute, and support complex applications.

Working with @net saves time, and finding your way through the program is quite easy. All of the menu features can be easily understood and detailed online help is also available. That makes @net not only useful for private users without any previous programming experience, but also for small, mid-sized or large software producers with complex requirements.

Those using browser technology for HTML documents are also most likely using other products that understand this language as well, like Microsoft Word, Excel, etc. The contents of a list generated with @net can be easily loaded into other programs (Excel, for example) for continued modifications. @net's menu items can be linked not only to other @net applications, but to other applications (*.exe), other URL's or to a static, local HTML page which was created with FrontPage.

More and more companies want to sell their products via the internet. This is also a very interesting application for @net. When an internet shopping page has been created with FrontPage or a similar program, then @net can take over the administration of the data used with that site. That means that all of the activity of the shop—orders, order confirmations, waybills, invoices, stock administration and stock planning—can be handled by @net.

How Does @net Work?

@net creates dynamic HTML pages using Caché functionality in connection with WebLink. The entire program sequence is controlled by single events on the page; for example, the selection of the buttons save, new, erase, open, next data set, branch to another program / form / link, etc.

An application is created by maneuvering through the individual pages and , as an integral part of that navigation, selecting specific methods, invoking related functions and defining necessary parameters. This process does not result in the compilation of completed HTML pages. When these steps are completed and the program is executed, all of the values are analyzed and, based on user and module information (authorization, language, culture, color and form), new dynamic HTML pages are created.

How are @net and the @net Applications Organized?

Since @net makes use of the object model, @net "thinks" of every HTML page as an object. Any combination of numerous attributes, methods or even other objects can be assigned to this object. These attributes can be very simple, like colors, shapes, frames or buttons, or they can be much more complex, like database objects. If an application draws upon a database or a new database is going to be created, @net automatically hands down these attributes to the application. The same thing happens, for example, with layout templates, which can be inherited from the client by the user, then by the application, then by the form, and then by the data field.

What Does the Object Structure of an @net HTML Application Look Like?

Here is an overview of several objects supplied with @net:

—> User-Object
- —> General Characteristics (Name / Password / Information)
- —> Language Traits
- —> Cultural Traits
- —> Layout Parameters
- —> Accessibility Parameters
- —> Company Affiliation
- —> Module Characteristics
- —> Connection Parameters (email Account, IP-No., Server)

—> Menu-Object
- —> Start Characteristics (Form / List / URL / .exe / ...)
- —> Language Traits
- —> Layout Parameters
- —> Module Characteristics
- —> Accessibility Parameters

—>Form-Object
- —> Layout Parameters
  - —> Frame Characteristics (Target Names)
  - —> Colors (Background Images, Background Music...)
  - —> Page Characteristics (Colors, Text, Forms, Images)

—> General Layout (passed down from the clients' characteristics)

—> Form Type Methods
- —> Menu Methods
- —> Standard Form Methods (Individual Fields)
- —> Grid Form Methods (Tabular Form)
- —> Methods for Creating Lists
- —> Manual Form Methods

—> Characteristics for Embedded Objects

—> Module Characteristics

—> Accessibility Parameters

—> Data Editing Characteristics

—> System Parameters

—> Database Characteristics
- —> Primary Key Characteristics
  - —> Layout Characteristics
  - —> Field Characteristics (Names / Field length / Types /...)
    - —> Relational Characteristics
    - —> Index Characteristics
  - —> Language Characteristics
  - —> Cultural Characteristics
- —> Data Field Characteristics
  - —> Layout Characteristics
  - —> Methods of Data Entry (Date, Text, Document, Password, Currency, num bers, Files, Check Boxes,...)
    - —> Relational Characteristics
    - —> Index Characteristics
  - —> Language Characteristics
  - —> Cultural Characteristics
- —> Methods and Characteristics of Data Storage
- —> Language Characteristics
- —> Lock Characteristics
- —> Accessibility Parameters
- —> Index Characteristics and Methods
- —> Data Field Characteristics
  - —> Methods of Data Entry (Date, Text, Document, Password, Currency, Numbers, Files, Check Boxes,...)
  - —> Relational Characteristics
  - —> Index Characteristics

—> Form Field Characteristics
- —> Layout Characteristics
- —> Field Validation Methods
  - —> Caché Routines (Checking the Data Entry Filed)
  - —> JavaScript (Checking the Data Entry Filed)
  - —> VBScript (Checking the Data Entry Filed)
  - —> Data Entry Rules
    - —> User-based
    - —> Application-based
- —> Methods of Data Entry (Selection, Multi-Selection, Radio button)

—> Standard Button Characteristics (save, open, new, delete, help, next.
- —> Save Characteristics =
  - —> JavaScript
  - —> VBScript
  - —> Sub-form Callup Methods
  - —> FrontPage (URL)
  - —> External Characteristics (.exe)
- —> Read Characteristics
- —> Erase Characteristics
- —> Help Characteristics
  - —> Language-dependent Help
- —> Search-Button Characteristics
  - —> Search and Selection Methods
  - —> Display Characteristics
  - —> Search Results Transfer Characteristics
  - —> Characteristics for Creating Sums

—> Button Characteristics (Manual Button)
- —> Language Characteristics
- —> Distribution Methods
  - —> JavaScript
  - —> VBScript
  - —> Sub-form Callup Methods
  - —> FrontPage (URL)
  - —> External Characteristics (.exe)
- —> Button Layout Characteristics

—> Company Characteristics
- —> Modules

—> List Generator Characteristics
- —> Page Layout Characteristics
- —> List Format Characteristics (<pre>for matted,<Table> or XML)
- —> Link-Options
- —> Database Characteristics
  - —> Primary Key Characteristics
    - —> Field Selection Characteristics (single / from-to / combination)
    - —> Field Layout Characteristics
    - —> Relational Characteristics
  - —> Data Field Characteristics
    - —> Field Selection Characteristics
    - —> Field Layout Characteristics
    - —> Relational Characteristics
- —> Testing Characteristics
  - —> Caché Routines
  - —> JavaScript
  - —> VBScript

All of the applications developed using @net are automatically:

Object Oriented: The required functionality was only achievable with an object model. Layout characteristics could be passed along ideally using this method. Object orientation has other advantages, as well, such as the output of Caché ObjectScripts, which allow data and data definitions in existing Caché applications to be passed along to other applications.

Multi-Lingual: The entire tool is multilingual, as are all of the applications which are created with it. Translations can be carried out either manually or automatically. As the applications are made, a translation tool scans the application for incomplete translations and suggests a suitable text, which can then be changed and saved. The translation includes headers, database names, menus, help text, status text, parameters, page tags and buttons. All language-dependent objects possess a language characteristic which is activated as soon as @net recognizes the language preference of the user.

Multi-Cultural: As with language recognition, some fields also have characteristics relating to culture, for example, fields with date, time and amount data. This feature becomes active as soon as @net recognizes the cultural preference of the user. The number "one-thousand," for example, can appear in various formats: English "1,000.00"; German "1.000,00"; Swiss "1'000,00"; Portuguese "1,000$00". etc. Dates and times can also be portrayed in the user's culture, for example, American "04/02/1999" or German "02.04.1999" and English "5 PM" or German "17:00".

Multi-Dimensional: @net makes it possible to join complex applications and to program distribution methods to other forms which may be multidimensionally connected. This also applies to URL's and *.exe applications (for example; Word, Fax, email).

Dynamic: All parts of the application (menus, forms, lists, even the tool itself) are joined with the form definitions, layout parameters, data definitions, data field definitions, button descriptions, search parameters, user language, user access security level, the JavaScript /VBScript, the object collections (save, open, delete, new, next record, prior record) only after Caché has been activated. The result is a dynamic HTML document with embedded JavaScript/VBScript. Testing of the fields is carried out by both JavaScript/VBScript as well as the Caché routines.

User-friendly: Aside from the advantages of a GUI, all fields for entry have been setup to support help text. Help text can be setup to contain both images and videos. It is possible for the user, if he or she so desires, to print out the entire operating instructions. Individual sections of instructions can be selected and viewed on the monitor. This user friendly approach extends to programmers as well. A multitude of help texts and functions are also available for programmers. @net can automatically create forms and lists, for example, depending on previously defined parameters and relationships, and then suggest the most appropriate type of data entry—check boxes, radio buttons or selection fields. Certain kinds of entry forms are automatically recognized by @net (for example, email addresses) and an appropriate button which will carry out the desired function is created automatically.

Some of the basic elements of @net include a list of complete applications and help programs, for example: automatic email and fax systems, downloading, embedded objects, group appointment calendars, date import and export tools, "Pin Board," etc.

Multi-user: Many users often need to have access to a database simultaneously, but the program job is actually terminated after the page is sent. A new kind of data locking system was needed for this kind of programming. The available locking mechanism (L + and L -) could no longer be used because they are process specific. The new mechanism can be individually adjusted for each file and this adjustment receives a limited time slot. The only person authorized to modify a data set is the owner of that set. That means that as soon as a data set is opened for viewing it is automatically locked. After the locking period is over, the next user can access the data set. The person who locked the set originally needs only to change focus from the field in question to cause the data set to be unlocked and available for the next user.

Capable of handing multiple companies in one system: An unlimited number of independent companies can be registered within an @net application. While utilizing one global, the company identifier becomes the first subscript level of the globals in the system. Each user is registered only for the company for which he or she

works. This makes it possible for an unlimited number of companies and users to access a name space in a company's database, all simultaneously and without disturbing one another.

Secure: After a user successfully logs in with his user name and password, a unique user number is assigned by @net. This number is embedded into the dynamic page being worked on and the connection is validated using that number. When the connection is closed, the number is erased and is therefore no longer valid. A different user is thus prevented from retrieving information by simply hitting the "back" button.

Ditmar Tybussek is Executive Manager, BEWIDATA Unternehmensberatung und EDV-Service GmbH fFCr den Deutschen Einzelhandel, Erthalstrasse 1, D-55003 Mainz, Germany

---

Report from the InterSystems Worldwide Developers Conference

by Kate Schell

The InterSystems Developer's conference was held in Orlando May 2-5, 1999. Others can probably offer hard statistical evidence, but the number of attendees from around the world definitely marked it as a major event. I met attendees from Japan, China, Bulgaria, Spain, Germany, and the Dominican Republic, as well as people from across the U.S. and Canada.

I'll be the first to admit that I attended for the two hands-on sessions, known as "academies": The Advanced Object Academy, given by Joe Gallant, and the Web Academy by Rob Tweed. Both were worthwhile. The instructors had excellent grasp of the material, and the examples were good. The number of supporting staff available to help out if you ran into difficulties was impressive. The logistics of setting up the number of PCs needed for the session were handled well.

One of the logistical coups this year was the arrangement of meals in a separate pavilion, away from the restaurants full of overtired children. This meant that it was possible to converse with the other people at the table. Of course, while discussing meals and programmers, it is important to say that the food was both good and plentiful.

Outside of the academies, InterSystems' staff discussed a number of topics ranging from support to product futures.

On Wednesday morning, InterSystems had John Gantz of IDC in to discuss "Survival Tactics for the new Internet Economy"; you'll have to catch his talk next time to find out why he considers church steeples critical to the growth of new technology. Since IDC did the judging for the Caché Innovator Awards, Mr. Gantz announced the winners. There were 31 applicants. Three prizes were awarded:

First Place:
G. Pierce Wood Hospital
Second Place:
Bewidata (See article on "Browser Tool @net" in this issue.)
Third Place:
Credit Information Center for The Américas

All told, the meeting was a good value for the money. Although we stayed at the conference hotel, there were several less expensive hotels available in the area. I admit that I'm tired of the Orlando location, but if it is to remain there, the arrangements made this year made it much more tolerable.

Hats off to the organizers and the presenters of this meeting for a job well done!