Version icon: 2.x, 3.0
Platform icon: DOS, Mac, Windows

Cool Tool: Automate and Enhance Your Reporting with Foxfire!

Whil Hentzen

It's so annoying: You spends days, months or years building the ultimate application for your users, and they're incredibly pleased with it. But wouldn't you know - just as you think you're done - they come back and expect to get the data back out of the application! You now enter the phase of the project that's of critical importance to the user but dreadfully boring to the programmer: building reports. Let's look at the new version of Foxfire!, an ad-hoc reporting tool that I've found indispensable for producing output in my applications over the past four years.

If you're an experienced FoxPro developer and haven't been living under a rock since the beginning of the decade, you're probably aware that Foxfire! from Micromega Systems in San Francisco is the premier third-party reporting tool for custom and vertical market FoxPro applications. The new version, Foxfire! 3.0, has recently been released and runs under FoxPro 2.6 for DOS, Windows and Mac, and Visual FoxPro 3.0 for Windows. It allows you to easily provide report management and ad-hoc query capabilities generation for the users of your application with a minimum of effort.

Most of the Cool Tools I've talked about in this column include a section on exactly what you can do with the product. Sometimes the product is obscure or the use to which you'd put it to isn't immediately obvious; other times, the product's literature or descriptions look like, well, look like a programmer wrote it. Foxfire! suffers from neither of these shortcomings. As a result, I'd probably be doing more of a disservice if I tried to present a laundry list of 'what you can do' with it in the space available. Accordingly, I'm going to briefly describe why Foxfire! is different from other output-oriented tools, briefly address the major areas of functionality and urge you to call up Micromega for a packet of literature if you've not already seen it, and then spend some time discussing how I use it in various scenarios. By seeing a number of real-life examples, I think you'll gain a better appreciation of how it can make you a hero in the eyes of your company or customers.

Requirements of Report Generation and Management

Foxfire! isn't just an overgrown report writer. It is the result of several passes by Micromega at report writing tools, and has been designed and engineered with the intent of creating a facility for getting at the application's data, not just providing a pretty GUI drag and drop report builder interface. The data that has been accumulated in the application is a hidden asset of the organization, but the means that the user has at their disposal to extract that data has to meet several criteria in order to enable that user to successfully mine the data.

First, the mechanism has to be "user-friendly." Obviously, this term is grossly over-used, but the acid test is whether a casual, moderately PC-literate user would be willing to try something out on a whim, or if they have to spend time learning, planning and strategizing before starting their first report. The second criteria is that the mechanism must be forgiving enough so that the user doesn't experience frustration or become overwhelmed. This would include the ability to back out of or undo choices easily, and create the report in steps, instead of requiring an "all or nothing" effort. Finally, the very nature of extracting data from a system is that of discovery - the user can't know all the questions they want to ask at the beginning of the query cycle. Each answer stimulates additional ideas and provides fodder for new questions that weren't known initially. This process of discovery is hampered if the user can't iterate in a timely fashion. A user will wait for 30 seconds for output - they won't wait for 30 minutes.

The keystone of Foxfire!'s interface is the presentation of the report creation and maintenance process. Foxfire! is billed as the 'one minute report writer' and, while a "minute" might be a bit of advertising copywriter license, I've brought Foxfire! into an organization and after a ten minute tutorial, the users were able to create and modify multi-table reports of their own design. In order for the process to work successfully, the user needs to be able to imagine what the report is going to look like, and needs to be able to drive the criteria without knowing how the underlying application is structured. Have you ever met a user who really understood the concept of parent-child-grandchild-lookup table ERDs?

Foxfire! uses a set of DBF files to store information about the application and then uses a single dialog to present that information to the user in a manner that they can understand it. Your job as developer is to populate those tables (Foxfire! has a setup wizard that does most of the work for you) with the proper architecture and then describe the data elements in terms that the user will understand.

This information includes data items (generally, fields from your tables) and joins (definitions of how the tables are related.) The user then uses the Request Editor to make selections for each of the components of the report. These components are the columns that will show up in the report (the data items), the records to be used to populate the report

(filters), the sort order and grouping requirements of the records, and the destination of the output. The report itself is stored in the third table, and thus is available for recall and further editing and manipulation.

04COOLRE.BMP: The Request Editor allows the user to see all of the components of the report at a glance.

At this point, I should make a point that Foxfire! refers to the entity being created (the 'report') as a "request." Foxfire! isn't just a 'report writer' but a tool to dig data out of the application - to request data. What you do with this request is up to you - you may choose to print it out, in which case the output type would be a report, but you may also want to produce alternate types of printed output, such as labels or forms, simply view the output on the screen, or route the result to another destination like a spreadsheet file, a mail merge document, or a graphing package.

Functional Capabilities

For those of you who haven't seen Foxfire! in action before, I'm going to walk through the creation of a request, describing some of the various options available at each stage. I presented this same process to a customer who was looking for an alternative to their current Focus reporting application one morning; they had the request to convert the system to Foxfire! signed before the end of the day.

The primary launching pad of a request is the Request Manager. From it, you can run, preview, edit or delete existing requests and create new ones. You can also access the rest of the Foxfire! infrastructure from the Foxfire! menu

04COOLRM.BMP The Request Manager is the home base for creating and maintaining requests.

After you've selected the type of request (detail, summary, cross tab or labels), you're prompted for a short name and a longer description. Then you start building the report with the Request Editor dialog. First on the list is to select the data items that you want on the request. These come from the Foxfire! metadata files that you populated earlier; while they generally map to the fields in your applications tables, they can also be expressions, functions, or UDFs. Creating a data item is as simple as selecting the field and entering a "user-friendly" English name that the user will feel comfortable with, but you can customize and enhance each data item with over 50 additional attributes.

04COOLDI.BMP The user doesn't have to know cryptic field names or understand how tables are related - using the Data Item mover dialog, they can just pick the elements they want to see on their request.

Again, remember that I'm just scratching the surface - basic requests are simple, but you've got a wealth of depth should you need or desire it.

You could conceivably run the request at this point, and if all you wanted was a simple tabular report, you'd be done. But it's likely that you want to control the order of presentation and which records will appear. The Sort/Group dialog is used to direct how the records in the request will be ordered and grouped. By default, you're presented with a list of the data items that have been selected in the request, but you can choose to display all data items and thus sort on elements that don't appear. Grouping capabilties are part of this dialog, since it's logical to the user to want to segment the report within sort orders.

Next stop on our "one minute" journey is the filter builder. You've seen filter builders in action before - and this one has all the functionality you'd expect, such as selection of data items, comparison operators, and expression builder capability - but as with the rest of the product, there are a number of features that make this one stand out. Oftentimes the only difference between one report and another is a couple of values in the filtering - "All invoices after the first of the month" vs "All invoices after the fifteenth of the month and that aren't from Corporate." Instead of building separate requests for each of these, you can create "Ask At Runtime" filters that allow the user to specify particular parameters at runtime. You can choose to require those parameters, to allow modification of other parts of the filter condition, or to even add and delete more filter conditions.

04COOLFB.BMP The Foxfire! Filter builder dialog allows the user to specify how much of the filter can be modified when the request is executed.

Another slick feature is the ability to query the application for existing values for a data item - instead of trying to remember of the Accounting Department's abbreviation is "ACCT" or "ACTG", you can view of list of values, (and find out that users have been entering both abbreviations.)

It's time to mention several new features in Foxfire! 3.0: outer join support, top N values, and externally available filter builder. The first two are found under the Special dialog (next to the Filter button). As you know, outer joins aren't directly

supported in FoxPro. However, you can build one and two table outer joins with a simple visual interface that shows the user what will happen according to which choices they make. You can also create output such as the often requested “Top 10” ranked values reports.

You can also call the filter builder in Foxfire! from your own applications. You simply populate the data item list with the values from the table you want to select from, call the filter builder, and let the user have their way with creating conditions as they desire. Once done, the filter builder returns a “FOR” string that you can use to winnow down the available records as requested.

It’s now time to move onto Output, or to paraphrase a major software company, “Where do you want your output to go today?” Not only are there an abundance of output options, including columnar, form, and master/detail reports, as well as labels, spreadsheets, mail merge, ASCII text file and DBF files, but you have extensive control over printer selection, default output options, and user-definable overrides. Furthermore, Foxfire! provides additional hooks into the report generation process so that you can run your own programs before and after the data selection and after the report output.

So that’s a quick tour for creating a request - but we’re not done yet. You can run a batch of requests unattended - including the entry of “ask-at-runtime” values and printer direction - through the batch builder, and you can run requests direct from your application (without going through the Foxfire! Request Manager) by calling Foxfire! with a series of parameters.

04COOLBB.BMP The Batch Builder allows the user to run a series of requests unattended.

Competitive Advantages

I wouldn’t think of creating a custom application without including Foxfire! It provides me with four very tangible benefits. These benefits are:

- A core set of capabilities that virtually every output-requiring application needs,
- The ability to let users make their own minor modifications instead of bugging you to “put the phone number before the company name on this report,”
- The empowerment of users (and the resulting benefits that accrue when a user takes ownership of an application), and
- A highly configurable architecture that enables you to set Foxfire! up to work with your application foundation regardless of its intricacies.

First, even the most basic Foxfire! installation, with no customization or tailoring, provides fucntionaltiy that you need for virtually every application that requires output. For example, how many hours have you wasted on setting up printer drivers? How about creating an interface that held a list of reports from which the user could pick and choose? How about the ability to batch a series of reports to run unattended? Perhaps you’ve tried to build a simple filter builder once you’ve found the RQBE dialog didn’t work in your applications. And most every developer has tried to present the user with an ‘options’ dialog that allowed the user to choose the sort order and the destination of a specific report. These are all built in - and with a great deal more robustness than the tool you’ve tried to handcraft yourself.

We can plug Foxfire! into a custom application in about 20 minutes by copying about a dozen core files into our data dictionary directory and then running the setup wizard to populate the data items and join tables. At this point, we have access to all of the above features - and they work the same way each time.

Probably the least rewarding task that we deal with as developers is having to respond to the incessant requests for minute modifications to reports. “Could you change the order of this report?” “We need to see the account number as the first column, not the second column.” “The header for this report should include the company name.” And so on. Incorporating Foxfire! into your applications allows you to give the users the ability to make changes like these by themselves.

Not only does this get those pesky little tasks off your back, giving you the time you need to more rewarding work, but it also helps you from having to nickel and dime your customers or users for little jobs. Finally, by enabling the users, they can make the changes immediately, instead of having to wait for you to get back to them.

I’m not suggesting that you’ll never hear from your users again, just that you’ll be relieved of 80 to 90% of those tasks. Foxfire! is designed to take most of the load off your shoulders, placing the mundane maintenance tasks in the hands of the users, so that you can concentrate on the demanding tasks that you can add the most value to.

The next benefit is the empowerment of the users by allowing them to create, modify and duplicate existing reports. This is a different point than the previous one - what I’m saying here is that by providing an ad hoc report generator, you give the users the ability to get at their data. They take ownership of the application, and their data, and become partners, not subjects. This managed query environment is what we’re looking for - allowing the users to mine their data in an iterative fashion according to their own needs and wants.

The fourth benefit is that you're not constrained to setting up your application in a certain way by adding Foxfire! to your tool chest. Foxfire!'s architecture is highly configurable - in fact, one of Micromega's primary design goals was to allow the developer to seamlessly meld it into their application. Calls to Foxfire! are run through a sophisticated configuration program that performs a broad set of functions that controls how Foxfire! operates.You can make modifications to this Foxfire! configuration program instead of having to modify Foxfire!'s source code.

The extent to which you can configure Foxfire! to fit into your application is nothing short of amazing. We use a specific environment for all of our custom applications, where we segregate source code, meta data, application-wide data, business-specific data sets, and both common and third party libraries all into separate directories. Since each data set is identical, we place all of the reports for the application in one directory, and point all data sets to that location. However, we also filter the requests so that a request created for a particular data set only shows up when the user is accessing that data set.

For example, we have a system that consists of a umbrella application that calls many department level subsystem applications. The data set was partitioned into multiple sets, with system wide data accessible to all and department level data accessible on a group by group basis. Simply by setting a series of filters in one of the Foxfire! configuration dialogs, we were able to configure Foxfire! so that all users had access to system wide requests (and their corresponding data items) but also were restricted to accessing the requests specific to their department. This application contains well over a thousand data items and close to a hundred requests, but each department only sees the information applicable to their part of the system.

Another customer of ours wanted a number of systems to be set up in identical fashion. Each system was designed for a specific customer of theirs, but each system's data sets had the same structure. We were able to set up a single group of Foxfire! requests that could be run against any of the systems, again by simply setting a data directory path in the configuration program. Furthermore, we used Foxfire!'s extensive user and group configuration capabilties to allow access to certain aspects of request creation and modification on a user by user basis.

Finally, yet another customer wanted a series of extremely complex reports that seemed as if they were going to be hand-coded. We were able to reduce the complexity of the output by breaking it into two parts. The first part was a "pre-Foxfire!" data selection program that made a first pass at filtering the data and doing some initial calculations. We then used this resulting data set to populate Foxfire! so that the users could handle the rest of the report creation themselves.

Where to get Foxfire!

Unlike many of the Cool Tools I've talked about, Foxfire! is a commercial product, not shareware or freeware. The pricing varies according to the capabilties of the version you need, but the Developers' Edition, the version you need to include Foxfire! in any number of your custom applications, royalty free, is $395. If you're still not sure, note that Micromega Systems includes a 60 day, no-hassle, full-refund policy with your purchase. You can also integrate Foxfire! with a vertical market application for a small royalty fee. You can get a fully functional demo disk that you can use to load your own sample data set (the only restrictions are that output is limited to 100 rows and printing hardcopy is disabled) by calling Micromega Systems at 800.453.6655, 415.346.6804, or emailing them at 73354.1745@compuserve.com.

Whil Hentzen is editor of FoxTalk and president of Hentzenwerke Corp, a Milwaukee, Wisconsin software development firm that specializes in strategic FoxPro-based business applications for Fortune 500 companies. 414.224.7654, CompuServe, 70651,2270.

Book of the Month

Following on the heels of last month's Book of the Month, Peopleware, is a tome by industry legend Larry Constantine. "Constantine on Peopleware" (Yourden Press, ISBN 0-13-331976-8) is a series of essays that originally appeared as columns in Software Development (formerly known as Computer Language Magazine).

You know how you're met with blank stares and utter amazement when you try to explain to a teenager these days that you listened to Bruce Springstein or saw John Travlota when you were in high school or college - over twenty years ago? Many of us who entered the industry as PC mavens are woefully unaware of the history of the pioneers - but Larry Constantine ranks as a member of the Hall of Fame. Known in the 1960's as the father of structured programming, he disappeared for a decade. When he came back, he edited Software Development magazine and authored a column that amused, angered, and aroused all who read it. Constatine on Peopleware is the collection of these editorials.

He addresses a wide range of software development issues. My favorites are his editorials on Cowboy Coders. He discusses his experience on a panel on structured vs unstructured programming; paired up against a manager from "Nanomush Software" who advocated chaos in the programming pits. "What kept programmers from reaching their full potential was managers who tried to impose standards, expectation, restrictions - stuctured methods, disciplined development, paper and pencil model building and software metrics are all unjustified impostions on the free artisic expression of our brilliant programming cowboys." This column, berrating the cowboy coder mentality, sparked an uprising in the development community, and provided the fodder for a series of columns.

Another classic editorial compares user interfaces with ski trails. Have you heard of the triphasic model of human interfaces? It describes how users - like skiers - have different needs duirng their lifespan. Initially, they need to ease and comfort of a bunny slope. Most users, and skiers, eventually need to graudate to something more difficult. On the other hand, only a few will graduate to the black diamond version of software - keyboard shortcuts and command macros. Nonetheless, the best software development will take into this concept into account.

Each column is just that - 3 - 5 pages - easily digestible while you're waiting for a half meg file download or while you're in the Private Room. Your outlook on this job will change and evolve while you're reading this book.