
Embedded Reporting

- ▶ *An Essential Application
Development Requirement*

Author: Marc Borbas
Contributor: Jaylene Crick, Colin Gray, Oz Greenberg, Jennifer Meegan, Eric Powers

Contents

Executive Summary	4
Why is Data Presentation So Hard?	5
To Code or Not to Code	6
Report Development Environment	7
Programmability	9
End User Controls	10
Flexible Deployment	11
Conclusion	12

Executive Summary

A graphical report is one of the most effective and familiar ways to present data to a business user. By converting data into business terms and adding formatting to mimic conventional business documents, you can increase the value and appeal of your application and the likelihood that it will be well received by your user base.

But if you're like most developers when they build an application, you have to balance your time between a variety of competing tasks. Data presentation often falls low on the priority list compared to core transactional and business logic. It also typically takes longer than planned, leaving you up against a tight project schedule with a lot of code left to write.

This document explores the various options you have for data presentation and will help you find the right mix of code and off-the-shelf components for your next project.

Why is Data Presentation so Hard?

Building great data presentation into your application seems like an obvious and universal need. You can increase the visual appeal of your application, communicate in a language and format that users understand, and present the most valuable thing your application collects—data. So why is complex data presentation one of the most frequent reasons cited for projects running over schedule or not meeting user needs?

Despite the importance of this function, most development frameworks and platforms provide only simple data presentation capabilities. You can use standard controls to project a query onto a web page, or you can use a charting component to generate a visual display of data. But what if you want to do both? What if you want to aggregate the data, but provide the option to show detail? What if you want users to be able to drill down through the chart? What if your users want to print it?

Chances are you're going to hit a wall with the standard components you have available, and you'll have to wrap them with a lot of code to meet even the simplest subset of user needs.

There's nothing wrong with code, except that what you thought you could do with a datagrid and a chart object just grew in scope and complexity. Over the years, we've seen customers build ASP/JSP reports to present relational data. And while the first few steps seem intuitive, the complexity quickly spirals out of control as additional users are added to the system, new technology is integrated, and reports need to be updated, customized, or added to meet growing user needs.

Let's take a look at how embedded reporting technology can help you dramatically reduce the amount of time you spend on data presentation, while preserving the flexibility you need to build great applications.

To Code or Not to Code

While designing a package tracking system, one of the company's most significant competitive advantages, you can bet that the developers at Fedex focused their energies on the high-priority tasks. You're always going to write code. The real question is, how do you maximize your coding time on the most important aspects of your applications, the aspects that often turn out to be the hardest and most unique?

Let's consider a scenario.

You've just been asked to post key customer accounting data onto your intranet site. The data is collected via the purchase order application you wrote a few weeks ago. All you need to do is write a couple of ASP-coded reports so people can use that data. You've coded five reports in total—including one that is scheduled to update automatically at the end of each day.

The report scheduled each night consists of a list of customers, grouped by company name. The report also includes personal contact information and hyperlinks to customer email addresses and web sites. One week later, you complete the reports and post them to your intranet site, giving the accounting department sole access to the reports.

Another week passes by and the marketing and sales departments hear about these reports and they want something similar, but with a few modifications. Sales would like the customer report to include telephone numbers. Marketing wants their customer report to include a chart of revenues per month by customer, grouped by region instead of by company name. Accounting has also identified that one of the five reports requires a lot of processing time, so they'd like it scheduled every week. They also let you know that when IT worked on the servers the day before, the reports were down for five hours and they were unable to access the data. This caused them to miss a couple of key deadlines for their auditors.

It looks like you'll need to delay your current project by at least one week to accommodate these changes. Plus you'll need to speak with your director about buying additional hardware to meet the needs of these additional users.

While all these needs are critical to the business, each one takes time to deliver—time you could spend solving tougher and more interesting problems. An embedded reporting toolkit provides the building blocks you need to meet end user needs in a consistent and efficient way, giving you more time to focus on other parts of your application.

Report Development Environments

It's easy to connect to a database. But what end users want is dynamic, actionable content presented in either static or ad hoc/parameterized formats. While you can typically build a few reports fairly quickly in ASP/JSP, it's unlikely that you'd invest the time to build a report design tool that would help you execute future requests faster. A powerful visual design tool is the first building block in a powerful embedded reporting toolkit. It will help you quickly build the reports your users need. Ultimately, it would be a powerful DHTML design and generation tool.

A highly summarized report can greatly assist a user or add insight into business process. Presentation-quality formatting using fonts, lines, graphics, and free-form layouts can provide your users with the exact look they need to view or post information in a professional manner.

Reporting technology also makes it easy to present your information in a format that people are familiar with—be it a cross tab, invoice, order form, etc. A visual designer supports the creation of virtually any report, style, or format required. With ASP/JSP, you are typically confined to table-based output. Banded or free form layout gives you the ability to quickly extract the information end users need and format it in a manner that helps to emphasize key areas of relevance.

A report development environment should also provide some of the key features that make IDEs powerful—dockable explorers that provide easy access to all parts of your report project, a repository to store commonly-used formulas, SQL statements, and design elements, and a set of expert tools to add interactivity to your reports.

This flexibility is available at the click of a button or wizard using report writers like Crystal Reports®. When coding via ASP or JSP—even when using web front-end designers such as Dreamweaver or FrontPage—you are looking at hours of additional development to add these features.

On the following page is an example of a fairly simple customer report, created with Crystal Reports and with JSP code:

- Five columns (two of them are content sensitive hyperlinks)
- One formula field
- A header for each column
- An alphabetical grouping by customer name
- Additional formatting for presentation quality

The first report below (figure 1) was created in 1.5 minutes using Crystal Reports. It was then published to the web as a dynamic report in just one minute. Similar steps using JSP code (figure 2) took an experienced developer 495 minutes. Even without incorporating the time it took to publish the report, the experienced JSP coder spent 210 minutes for the report design process.

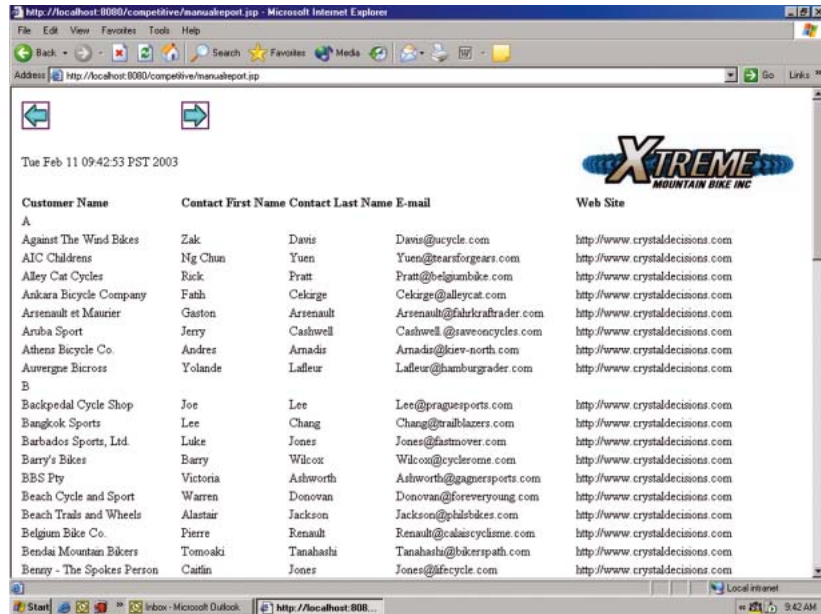


Figure 1: Report created with Crystal Reports.

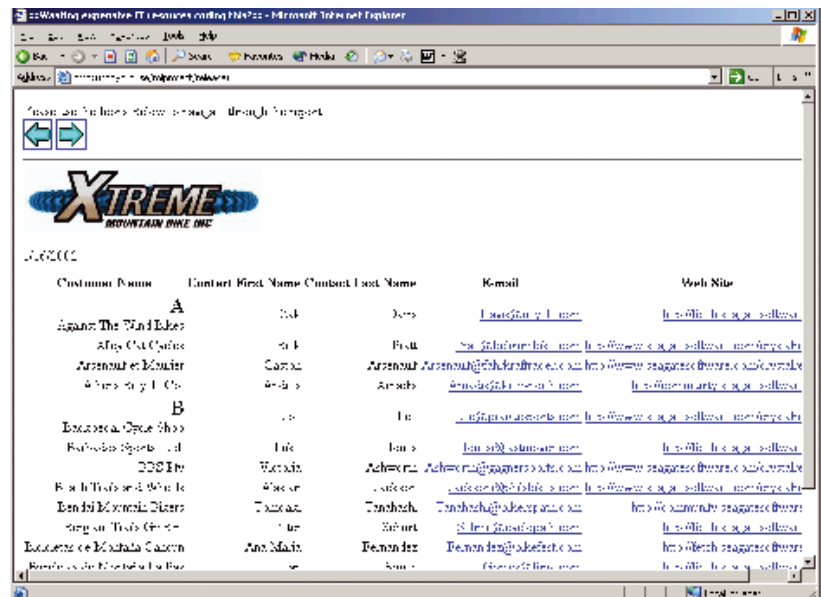


Figure 2: Report created with JSP code.

Programmability

One of the biggest risks of using an off-the-shelf component rather than custom coding is that you'll run into a limitation in the component's design that impacts your application. The second building block of a powerful embedded reporting toolkit is an extensive object model that ensures you won't hit a wall if you need to incorporate custom features or behavior into your application.

Programmability divides into three main areas: a document object model for fine-grained control over the report document and its processing, an event model for responding to user interaction, and a data interface for pushing custom data sources into a report.

The document object model can be used to modify a report at run-time or even build a new report from scratch within your application. This allows you to build a more flexible application that delivers tailored report views based on user input. You can add or remove columns, change sort order, or filter the report to meet specific user needs. Without this object model, you would have to accommodate all possible needs within your original reports, resulting in more complex reports or a larger number of reports.

An event model can be used to embed reports as part of a workflow in your application. You can capture user events (mouse clicks, refresh events, etc.), intercept the data in that particular area of the report, and define custom behaviors. This event model essentially adds reporting-specific events to what is already available in an ASP/JSP environment.

Finally, a data interface can accommodate your application's on-the-fly custom data sources (e.g. a shopping cart). You can easily format the data from these sources and offer conventional reporting functions (export, printing, pagination) to add value.

End User Controls

To maximize the value of the reports you create and embed, you will need to offer a range of intuitive controls so end users can easily navigate, search, and print those reports. The third building block of a powerful embedded reporting toolkit is a set of customizable end user controls for report consumption.

If your reports span multiple pages, navigation and searching become critical end user requirements. While page-based navigation may be adequate for smaller reports, you may want to include a tree control or a table of contents for larger drill-down reports. Keyword searching can help with simple queries (e.g. find the first mention of “John Smith”), but larger reports may require more complex searching (e.g. find all customers who spent over \$5000). Again, providing these capabilities on top of every report means you don’t have to build one-off reports to meet varying user needs.

At some point, most of your users will want to print a report or save it for offline reference. Printing ASP/JSP pages is unpredictable at the best of times and can be extremely challenging if your content has specific formatting or regulatory restrictions. Saving these pages as web archives is possible, but often requires additional user training. Reporting tools like Crystal Reports include robust printing and exporting to a variety of standard formats to help your end users easily keep snapshots of their reports for offline use.

While many of these controls are essential end user requirements, you may want to tailor their appearance and behavior to suit your application and users. Each of these controls should have a toggle on/off feature, and a customizable appearance to minimize end user training requirements and integrate seamlessly with your application.

Flexible Deployment

Ultimately, you'll need to deploy reports alongside your application., moving it from a development environment to a production environment or even redistributing it to other companies. The final building block in a powerful embedded reporting toolkit is a range of deployment options and features that are designed to work in your environment or your customers' environments.

The default deployment option for most developers is the application server. Most of your application is designed to run on the application server and your data sources are typically created as part of your application. You should be able to drop a reporting component onto your application server to provide a run-time environment for your reports. In a thick-client environment, you should also be able to redistribute a similar component to client desktops.

In the deployment model above, the application server is simultaneously running application logic and report processing, which may result in unsatisfactory performance if you have large data sets or complex reports. To address these issues, you may want to consider moving report processing to a dedicated web server (and using affinity to route requests to this machine) or deploy a separate report server.

Finally, your web application must provide end users with fast response times. Typically, the fewer number of hits to your database and the less time spent processing report results, the faster your application is able to send information to end users. You may also want to consider options for scheduling large reports, caching report results, and balancing load across multiple machines.

While one model may suit your application today, you will want to ensure you have a growth path should your load patterns or end user needs change.

Conclusion

Data presentation is one of the most important, yet under-estimated, aspects of application development. With data presentation, you can increase the visual appeal of your application, communicate in a language and format that users understand, and present the most valuable thing your application collects.

Like every aspect of your application, you have the choice to build from scratch or combine off-the-shelf components to deliver the right solution. The right embedded reporting toolkit can help you spend less time on the repetitive aspects of data presentation, while still providing the flexibility and power you need to build a great application.

Business intelligence (BI) vendors like Business Objects provide a range of different deployment options, from components that run in an application server container to fault-tolerant reporting architectures that provide scheduling and security. The right reporting tool for developer environments should include a powerful visual designer, rich developer interfaces, embeddable end user controls, and flexible deployment components for web and Windows applications.

With Crystal Reports—now part of the Business Objects family of BI products—you can enhance your existing integrated development environment (IDE) with re-usable components for developing reports and integrating them into your application. Your end users will benefit from highly formatted reports with intuitive navigation, printing, exporting features, as well as advanced report modification and searching controls.

For more information on Crystal Reports, visit the DeveloperZone at http://www.businessobjects.com/products/dev_zone/default.asp or the Crystal Reports section on the Business Objects website at <http://www.businessobjects.com/products/reporting/crystalreports/default.asp>.

**Americas**

Business Objects Americas
3030 Orchard Parkway
San Jose, California 95134
USA
Tel: +1 408 953 6000
+1 800 877 2340

Asia-Pacific

Business Objects Asia Pacific Pte Ltd
350 Orchard Road
#20-04/06 Shaw House
238868
Singapore
Tel: +65 6887 4228

Europe, Middle East, Africa

Business Objects, SA
157-159 rue Anatole France
92309 Levallois-Perret Cedex
France
Tel: +33 1 41 25 21 21

Japan

Business Objects Japan K.K.
Head Office
Yebisu Garden Place Tower 28F
4-20-3 Ebisu, Shibuya-ku
Tokyo 150-6028
Tel: +81 3 5447 3900

For a complete listing of our sales offices, please visit our website.

► www.businessobjects.com