Volume 11,No 1,April 2004

Volume 11, Number 1, April 2004

Technology Electronic Reviews (TER) is a publication of the Library and Information Technology Association.

Contents:

REVIEW OF: Tara Calishain and Rael Dornfest. (2003). *Google Hacks, 100 Industrial-Strength Tips and Tools.* Sebastopol, CA: O'Reilly.

REVIEW OF: Tara Calishain et al. (2003). *Google Pocket Guide*. Sebastopol, CA: O'Reilly.

by William Lund

Who would have thought that Google supported proximity searching? Or that the searcher can limit searches to specific websites, types of websites, or to portions of the web page? Or that Google can display or translate in a variety of languages? I certainly had no clue that Google had some of the more sophisticated features found in commercial search engines. *Google Pocket Guide* and *Google Hacks, 100 Industrial-Strength Tips and Tools* provides fascinating insights into the power and flexibility of Google, far beyond the one-size-fits-all search of the Google homepage.

The *Google Pocket Guide* is authored by Tara Calishain, Rael Dornfest, and D. J. Adams, whereas *Google Hacks* is authored by Tara Calishain and Rael Dornfest alone. Tara Calishain is the author of several books about the Internet, as well as articles on Google for O'Reilly, the publisher of both books. Rael Dornfest is an O'Reilly author of a variety of technical Internet related topics.

The *Google Pocket Guide* is a pocket-sized summarization of the larger *Google Hacks*, including only those portions of the larger work which deal directly with the user interface from a web browser. Included in the volume are the details of the syntax to invoke specialized features of the search engine. For example, when searching for "The Wind in the Willows" three of the words are stop words: "the" twice and "in". Some search engines will refuse to perform a search which includes stop words, but Google just ignores them. This may complicate a search for the specific phrase or title. However, if you really want to search a stop word, Google will permit this: enclose the phrase in quotes or place a "+" in front of each stop word to be included. This allows the user to force Google to include stop words, particularly in a title or phrase search, making the probability of success greater. Further, by using the "site:" modifier, you can limit your searches to a specific domain.For example "site:edu" will only search web pages from the .edu domain. Lastly, what is the difference between searching for "Apple computer" and "computer apple"? These two searches will return different results. The pocket guide explains why and how the user can maximize appropriate search results.

The pocket guide also includes detailed information on all of the services offered by Google, beyond the broad web searching we are all familiar with. For instance, Google offers searching of catalogs and on-line shopping sites for the shoppers among us, an archive of newsgroups extending back to the beginning of Internet time, searches of news feeds, of images, and of phone numbers by residence or business. All of these services are described, albeit briefly, in the pocket guide.

Google Hacks is divided up into 100 sections, each explaining a different insight or interface to Google. The first chapter covers the search interface and is essentially identical to the pocket guide described above. The remaining chapters include information on special services and collections, and creating interfaces to Google with or without the application program interface (API). The API sections include both code snippets and full-blown applications. This book does

not include a CD, however, the larger code examples can be found on the O'Reilly web site. On the lighter side, there is a section on Google games and pranks including "Google Whack", "GooPoetry", and "Creating Google Art."

The last chapter, is the most useful for webmasters who want to have some control over how Google views and indexes their pages. For instance, how can you help Google to appropriately index your pages so that they can be found through basic searches? How should your website be structured so that Google can find your content? What do you do if you do not want Google (or other well-behaved search engine spiders) to index your site? At the heart of Google is the Page Rank algorithm, by which Google orders the results of a search. *Google Hacks* describes the algorithm and how a website designer can optimize the structure, content, and metadata.

Overall, if you are a basic Google user who wants to get the most out of its capabilities, the pocket guide is probably all you need to be able to ratchet up your use of Google. It thoroughly covers the hidden user interface features, some of which do not appear to be documented on the Google site itself. One criticism of the pocket guide is that it appears to be lifted directly from the larger *Google Hacks* book without trying to change its format. I expect a pocket guide to be heavy on tables and lists so that I can thumb quickly to the information that I want without a great deal of discussion or text. This is not the case here. Whole sections are lifted verbatim from *Google Hacks*. (The authors of the two works are the same so there is no plagiarism concern.)

If, on the other hand, you would like to take the Google APIs for a spin, developing your own interfaces using the Google search engine, *Google Hacks* is more appropriate. All of the information found in the pocket guide is available in *Google Hacks* and since the format is the same in the two books, there is little use in purchasing both books.

Both books are recommended reading, but for different audiences. If you are just going to use Google as a search engine, then the *Google Pocket Guide* is best suited for you. However, if you want to go beyond this, building your own interfaces or optimizing your website for Google indexing, *Google Hacks* will meet your needs.

A note for O'Reilly Safari users, only *Google Hacks* is available through your Safari subscription. To see the table of contents for either book, visit the O'Reilly website.

William Lund, Lee Library, Brigham Young University.

REVIEW OF: Arlen Feldman. (2003). *ADO.NET Programming*. Greenwich, CT: Manning.

by John Wynstra

ADO.NET is the latest in a long list of Microsoft data access technologies, enabling programmers to develop applications that can connect to databases for the purpose of retrieving, inserting, updating, and deleting data. ADO.NET is not a programming language in and of itself, but rather, it defines a set of objects with properties and methods that are used in conjunction with other programming languages such as VB.NET, C#, and C++. ADO stands for ActiveX Data Objects and .NET is the moniker branding Microsoft's most recent development tools and framework.

ADO.NET Programming is a how-to manual for programmers. It is well written, well organized, and easy to read (for a programmer).The author has a relaxed, almost conversational, approach as he goes about the business of imparting information and advice. He does a nice job of including relevant information without getting bogged down in unnecessary detail, and along the way he manages to interject a sense of humor into his text. The book is organized into seven parts and has numerous tables and diagrams to illustrate concepts concisely.

In Part 1 "ADO.NET Overview", the author gives a history of Microsoft data access technologies, and then he shows how ADO.NET differs from past technologies. Along the way, he covers XML concepts and explains how to set up the code samples that will be discussed throughout the book. The code samples are obtained by downloading them from the book's website available at http://www.manning.com/feldman (http://www.manning.com/feldman).

Part 2 and 3 contain the core material of the book. Part 2 is entitled "ADO.NET basics" and expounds on the .NET data providers covering connection objects, command objects, and datareaders amoung other things. Part 3 is entitled "The Dataset" and provides detailed information on one of the most powerful features available in ADO.NET. The Dataset makes it possible to realize the disconnected data model, a strategy which is at the heart of ADO.NET.

I appreciated the fact that author included a separate chapter for the OLE DB data provider in Part 2. Too many authors take the easy route of demonstrating the SQL data provider in depth and then give only cursory coverage to the OLE DB provider. The end result of that approach is that the reader is left trying to figure out necessary syntax details for themselves. That is not the case with this book. The few examples provided go a long way toward enabling the reader to begin to use the OLE DB data provider.

Along these same lines, the author's in-depth coverage of the DataSet is a valuable aspect to the book. This is by far the most powerful feature of ADO.NET and worthy of the 7 chapters in this section. By starting at the top level and systematically exposing the layers beneath, the author very clearly explains how the DataSet works and what its practical uses are.

Part 4 "Databound controls" is a little disappointing. The author could have given more coverage to the properties of controls themselves, along with more examples demonstrating techniques for presenting data via these controls beyond just how to bind the data to them. The author stated up front that this was not within his intended scope for this book, so it was not an oversight on his part, but including this would have made the book stronger in my opinion.

In Part 5 "XML and ADO.NET" the author turns his attention to XML. Since Microsoft has made support for XML a core goal within the .NET framework, it is very appropriate that a significant section of this book be dedicated to the topic. The material is pretty straight forward and nicely enhanced by many practical examples. The XML chapter found in Part 1 of the book serves as a primer for those not already familiar with XML.

Part 6 "Useful extras" contains miscellaneous advanced topics that do not easily fit in other sections of the book such as connection pooling and distributed transactions. In the final chapter titled "Recommendations and advice" he sums up some of his personal best practices, although he does a good job of dispensing this advice throughout the book.

The book's seven appendices serve as detailed reference for the following ADO.NET programming topics: ODBC data provider, Connection classes, Command classes, DataReader classes, DataTables, DataColumns, and DataRows. The methods, properties, and events (if any) of these objects are described here in detail using figures, tables, code samples, and narrative text.

The programming examples in this book are written in Microsoft's new C# programming language which was introduced as part of the .NET platform. Being a new language, C# has a smaller base of programmers than Visual Basic or C++. However, Microsoft has made a determined effort to design ADO.NET to be language neutral so that the API for ADO.NET Objects is basically the same across all the programming languages that Microsoft utilizes. This means that even a Visual Basic.NET programmer should be able to find a lot of valuable information in this book.

The code samples are easy to locate since each is set off with a ¼ inch gray title bar that sums up the purpose of the example and provides a listing number. In addition, the author annotates the code with numbered labels that refer to explanatory comments directly following each code sample. The examples are concise and are written to illustrate single concepts. This approach is advantageous compared to the approach found in some books of including fully developed code samples that incorporate multiple concepts into a single program of hundreds if not thousands of lines. The shorter code samples found in this book make it is easy for the reader to focus on exactly what the author is demonstrating.

This book combines practical insight with a lot of useful information in a nicely organized package. The author's style of communication is clear, succinct and relaxed. The topic is adequately covered for both the beginning programmer as well as the more advanced

programmer. I recommend this book as a valuable addition to the computer science/programming languages collection.

John Wynstra, Library Information Systems Specialist, Rod Library, University of Northern Iowa.

REVIEW OF: Bruce Tate. (2002). *Bitter Java*. Greenwich, CT: Manning.

by Yan Han

Java is a common object-oriented programming language used for building enterprise applications and server-side applications. This book is written about Java antipatterns for intermediate server-side Java programmers. Design patterns provide an effective way to guide software design. Antipatterns are negative sides of design patterns, which can cause undesirable outcomes such as spaghetti codes, ineffective algorithms and even failed projects. Antipatterns can be found in basic Java programming through to J2EE programming such as Servlets, Java Server Pages (JSPs), and memory management. Understanding antipatterns will help software architects and developers prevent and recover from these negative outcomes.

Organization

The focus of this book is Java antipatterns and how to eliminate or reduce them. A unique characteristic of this book is that each chapter starts with one or two paragraphs of the author's kayaking experience tied to the programming context. The description of the kayaking experience is easy to identify with and can be skipped for uninterested readers.

The book has 11 chapters arranged in 3 parts. Part 1 covers the fundamentals of design patterns, antipatterns, and Internet standards supporting server-side Java. Part 2, the core of this book, provides detailed antipatterns in Java programming, including Java Servlets, JSPs, Java memory management, and XML. Part 3 is an overview of the context in terms of coding standards guidelines and performance. Source code examples are provided in the book and are available for downloading from the publisher's website.

There are two interesting chapters that deal with cache and memory management issues in Java, while other chapters address various issues such as antipatterns and JSPs. Chapter 5 focuses on the cache management issues, with a sub-section designated for an introduction to the classic read/write problems. Chapter 6 discusses memory management issues in Java,

usually not found in other books. Many novice Java developers assume they need not worry about memory leaks, because there is a built-in garbage collection mechanism which manages objects in Java. However, the author makes strong arguments against this assumption and demonstrates the needs for memory management in Java. The author addresses this by introducing algorithms for garbage collection and pinpointing circumstances causing Java memory leaks.

Strengths

The author has 14 years' experience at IBM and founded a company specializing in Java consulting, education and services [1]. This book, published in 2002, is current and well written. Bruce Tate is also the author of another Java book called *Bitter EJB*.

Bitter Java is well organized and includes a detailed table of contents. The index is thorough and the bibliography is a plus for interested readers seeking for additional resources. The selected examples of Java code adequately address issues in the corresponding contents. The author effectively delivers the content by using different fonts for different concepts. For example, Italic typeface is used for new terms, and Courier typeface is used for Java code examples. The content is appropriate for the intended audience and the book is printed on quality paper with the right font sizes.

Weaknesses

There are two incorrect or incomplete concepts identified in the book. On p. 29, the author states that "A firewall is simply a machine placed between the Internet and the HTTP servers of a corporation…", which is not totally accurate. The firewall explanation on p. 285 is correct. "A firewall can be hardware and/or software, and it can function more than HTTP protocols". A firewall definition from a computer science dictionary or encyclopedia such as *Microsoft Computer Dictionary* would be helpful if it had been included. In addition, Figure 9.2 and the associated annotation on p. 269, describing the concept of access control among private, protected, and public classes, is not correct, because the protected classes concept of Java is different from that of C++ (Sun, 2003).

Some typos were also found in the book's source codes. A few Java file names are incorrect and therefore they cannot be compiled. For example, the Java class "CompoundJSPCommand", which has the incorrect file name "CompounndJSPCommand.java", should be "CompoundJSPCommand.java". It is recommended that one reads the errata of this book maintained by the publisher [2]. Moreover, the lack of certain supported classes and packages in the source codes makes them hard to be compiled without additional efforts. The intended audience will not benefit from un-compiled code. In addition, the author did not state the criteria for selecting the bibliography, and there are no annotations included for the entries. This information would be helpful for readers to identify high-quality resources.

Summary

Bitter Java is a fair Java programming book to address antipatterns for intermediate Java programmers. Although some errata are suggested to be fixed for future releases, the content and presentation are useful for the intended audience. Its unique cache and memory management topics should benefit other readers as well.

Notes:

[1] "The Interview: Bruce Tate, *Bitter Java*". Retrieved April 2004, from http://www.javaperformancetuning.com/news/interview036.shtml (http://www.javaperformancetuning.com/news/interview036.shtml)

[2] "Forum: Bitter Java". Retreived April 2004, from http://www.manning-sandbox.com/forum.jspa?forumID=16&start=0 (http://www.manning-sandbox.com/forum.jspa?forumid=16&start=0)

References:

"Classes and Inheritance" (2004) in *The Java Tutorial* [online]. Santa Clara, CA: Sun. Available from (http://java.sun.com/docs/books/tutorial/java/javaOO/accesscontrol.html (http://java.sun.com/docs/books/tutorial/java/javaOO/accesscontrol.html))

Microsoft Computer Dictionary. (2002). 5th ed. Redmond,Wash.: Microsoft Press.

Yan Han, Systems Librarian, University of Arizona Library.

REVIEW OF: John Viega, Matt Messier, and Pravir Chandra. (2002). *Network Security with OpenSSL.* Sebastopol, CA: O'Reilly.

by Katherine Villyard

OpenSSL is an open source (meaning, among other things, that the source code is freely available) implementation of SSL (Secure Socket Layer), a protocol that can be used to secure an application's network communications. Securing network communications is important to prevent electronic eavesdropping, especially in commercial transactions. OpenSSL secures

network communication through data encryption, using public key algorithms such as RSA. If a security protocol is applied incorrectly it will not effectively secure an application, however strong that security protocol may be.

This book is a technical reference for programmers on correctly and securely using the OpenSSL library, primarily in C, although Perl, Python and PHP are also addressed. This book is an introductory reference to OpenSSL, but the authors assume the ability to code in a programming language (preferably C, Perl, Python or PHP). Most of the examples are in C, as C has the most full-featured implementation of OpenSSL. Only twenty-four pages are devoted to Perl, Python and PHP. This makes sense, as the focus of the book is OpenSSL: instructing a reader in a specific language is beyond the scope of this book.

The authors are highly qualified to write about the subject of network security. John Viega has written multiple security tools and is the co-author of *Building Secure Software* (Addison-Wesley, 2001). Matt Messier has written security software and is a founder and director of windows development at Secure Software Solutions. Pravir Chandra, also of Secure Software Solutions, is an expert in language-level security.

Stylistically, the book is typical of O'Reilly manuals: it is very detailed (it is more useful as a programming reference than an introduction) and completely devoid of "talking down" to the reader. This is not a book for a beginning programmer to learn C, and a novice PHP programmer intimidated by C might be frustrated by the book's strong C focus. That said, the OpenSSL concepts illustrated with C examples do translate to the implementations for Perl, Python and PHP. Concepts such as public key infrastructure, public key cryptography, random number generation and hashes fall well within the scope of the book and are clearly explained.

Programmers seeking a reference on using the OpenSSL library in writing network applications will be well served by this book. A reader seeking a "for dummies" introduction to OpenSSL will most likely be frustrated, especially if they are expecting instruction in a programming language as well. Readers should consider some programming knowledge a prerequisite for this book.

PostScript:

Since the time I first wrote this review, vulnerabilities were found in some versions of OpenSSL. These vulnerabilities could lead to a denial of service attack. CERT (the U.S. Computer Emergency Readiness Team) recommends an upgrade to version 0.9.6m or 0.9.7d, which are not vulnerable. For more information, see http://www.us-cert.gov/cas/techalerts/TA04-078A.html." (http://www.us-cert.gov/cas/techalerts/TA04-078A.html.")(viewed on April 5, 2004)

Katherine Villyard, Georgia Perimeter College, Clarkston Campus.

About TER