

IEEE DISTRIBUTED SYSTEMS ONLINE 1541-4922 © 2005 Published by the IEEE
Computer Society
Vol. 6, No. 5; May 2005

Editor: Marcin Paprzycki, <http://www.cs.okstate.edu/%7Emarcin/>

Book Reviews: Java Tools and Frameworks for Web Application Development

Fernando Berzal, *University of Granada*

Art of Java Web Development

By Neal Ford

624 pages

US\$44.95

Manning Publications, 2004

ISBN: 1-932394-06-0

The overwhelming variety of available tools, frameworks, and languages makes selecting the most suitable tool for implementing new Web applications difficult. Fortunately, books such as *Art of Java Web Development* provide an indispensable guide to sort through the diverse ecosystem of Web application tools and frameworks.

It's virtually impossible to completely cover all the choices a developer can make nowadays, because you'd have to cover everything from scripting languages to application servers, relatively simple frameworks to content management systems. Consequently, *Art of Java Web Development* focuses on some popular alternatives for developers using the Java programming platform. If Java is your chosen language, Neal Ford overviews six frameworks and numerous tools you should be aware of before you start building Java Web applications from scratch.

Web applications' evolution

To start, the book follows Java Web applications' evolution since their early days. Not so long ago, people built Web applications writing "simple" servlets. Servlets are Java applications whose output is the HTML page you can see at your Web browser. You build servlets like other object-oriented applications. However, raw servlets include HTML tags within program logic, which makes GUI design and dividing work between programmers and graphical designers more difficult than necessary. Templates can be useful to separate presentation from logic, but they usually incur a high performance cost.

Java Server Pages, which are compiled into servlets, invert the mixed-presentation-and-logic problem, immersing program logic in the HTML code. Custom tag libraries are the next evolutionary step in this short but intense Java Web application history. Custom tags help reduce the amount of code embedded within JSP pages. If used properly, which often isn't the case, you can realize evasive goals such as better source code organization and easier maintenance in Web application development.

Ford reviews recent history, setting the stage for Web frameworks, then introduces the Model 2 design pattern as the right way to separate concerns in a Web application front end. Model 2 is nothing more than the adaptation of the well-known Model-View-Controller pattern to Web applications' stateless nature. Here, the model resides in a standard Java object (also known as POJO, plain-old Java object), a servlet acts as controller, and a JSP page provides the suitable view. This makes the best possible use of both servlets' and JSP pages' strengths. Model 2 frameworks, in particular, also tend to use the command design pattern to reduce controller servlets by providing a shared front controller for the whole application.

Web frameworks

Frameworks provide the common infrastructure most Web applications need, thus freeing the developer of having to continuously reinvent the wheel. As you might expect, developers frequently use the Model 2 design pattern.

The second part of *Art of Java Web Development* surveys tools commonly used in developing Java Web applications. Even though this Web Frameworks section includes chapters on six particular tools, they're not all frameworks, as Ford himself acknowledges. In fact, the six "frameworks" Ford discusses are an assorted set of tools. They provide Web developers with many possibilities from simple template libraries to XML publishing frameworks:

- Struts, a popular Model 2 framework that includes its own custom tag libraries, is configured via XML files, easily supports internationalization, and allows declarative validation.
- Tapestry is a more complex framework that hides many Web application details. HTML templates, XML page specifications, and Java backing classes combine to let the developer build componentized Web applications as if they were desktop applications.
- WebWork, one of Ford's favorites, is closer to Struts. Despite Ford's positive assessment of WebWork, I still don't think it's different enough from the more popular Struts to adopt it over Struts. A closer examination could tip the balance, but the book doesn't provide enough details.
- InternetBeans Express, which bundles with Borland JBuilder, isn't a Model 2 framework. It's a rapid-application-development-based framework, ideal for quick-and-dirty applications, that includes visual-component-library-like controls (for those who aren't familiar with Borland tools, the VCL is the component library bundled with Borland integrated development environments such as Delphi or C++Builder). Given their different scope, JBuilder also includes Struts apart from InternetBeans Express.
- Velocity isn't a Model 2 framework, either. It's just a template language that can replace JSP and operate in conjunction with other Model 2 frameworks.
- Cocoon is an XML publishing framework that includes a complex XML version of JSP. As a Web framework, it's similar to Struts, albeit much more complex for standard Web applications.

The chapters devoted to these tools and frameworks provide an eagle-eyed view of the benefits of using frameworks to simplify Web application development. To compare them on a level ground, Ford builds the same small application with the six different frameworks to illustrate each framework's pros and cons.

Even though Ford evaluates different frameworks and discusses his own criteria, readers will have to draw their own conclusions. Your particular context has a definite role in the framework selection process, and you'll have to weigh the benefits and disadvantages as they apply to your situation.

All the frameworks the book discusses are worth knowing, but they form a miscellaneous

group. They offer the developer completely different abstraction levels, from relatively simple Velocity templates and Java Standard Tag Libraries to high-level architectural frameworks such as Tapestry and Cocoon. A head-to-head comparison of similar frameworks (in scope and complexity) would be a much more significant contribution to the Java community, because it's nearly impossible to stay current on Web development's latest frameworks and techniques.

As a minor drawback, the book is too biased toward Apache-sponsored projects. For example, half of the frameworks Ford discusses are part of the Jakarta project (the Apache Software Foundation's Java section), while Cocoon was part of the Apache XML Project before it became an independent Apache project. Discussing other options would greatly improve the book's value. For instance, three potentially relevant candidates that a more comprehensive overview of Java Web development frameworks should include are the Spring MVC framework with AOP support (<http://www.springframework.org/>), the Java ServerFaces user interface component framework (<http://java.sun.com/j2ee/javaserverfaces>), and the SiteMesh framework for page layout (<http://www.opensymphony.com/>) (from OpenSymphony, as WebWork).

Best practices?

The last 240 pages of *Art of Java Web Development* are mostly independent chapters describing topics of interest for Web developers. Ford discusses how to solve common problems and introduces ancillary but important application development tools.

This section starts with a chapter on separating concerns in Web applications. In other words, it shows where to put things when organizing a Web application (for instance, how to use JavaBeans and Enterprise JavaBeans). Surprisingly, this chapter ends with a client-side validation example where JavaScript code is included in Java source code as String literals. Fortunately, Ford mentions the Jakarta Commons Validator at the very end to lessen the negative impact this practice might have if broadly used.

The book also discusses some implementation details that might affect Web application usability, such as viewing large tables one page at a time, sorting columns, allowing undo operations, and properly handling exceptions. The following chapters deal with disparate topics such as profiling, debugging, logging, and instituting a "session cop" to avoid storing unnecessary data in session variables, which could hamper application scalability.

Ford also treats pooling and caching, because they're fundamental techniques that ensure application scalability. However, better and more comprehensive treatments of these topics exist, such as Paul Dyson and Andrew Longshaw's *Architecting Enterprise Solutions* (John

Wiley & Sons, 2004) or Michael Kircher and Prashant Jain's *Pattern-Oriented Software Architecture: Patterns for Resource Management* (John Wiley & Sons, 2004).

Unfortunately, *Art of Java Web Development* doesn't include relevant references beyond publisher-specific self-promotion.

Ford briefly covers other topics in the last few chapters, including unit testing (with brief introductions to JUnit and JWebUnit) and Web Services with Apache Axis. Again, a nice Web Services introduction is spoiled by an unfortunate example employing the wrong granularity for exposing Web Services.

As you might have already noticed, I wouldn't miss this section too much. I'd prefer it if the book had a more comprehensive overview of frameworks for Web application front ends, and I'd look for design and implementation best practices elsewhere. However, novice developers will probably benefit from the advice included in these final chapters if they can digest it from a critical viewpoint.

Conclusion

Despite its minor shortcomings, this book is a valuable resource for developers involved in Java project tool selection and programmers interested in getting a good grasp of current implementation alternatives for Java Web front ends. The survey of different frameworks is certainly a time-saver. The overall presentation of Web application evolution and the key design decisions involved in their development is noteworthy. If you're acquainted with the Java programming platform and already know how Web applications work, this book might benefit you. Read it, ponder the alternatives, and decide your own course.

Fernando Berzal is an assistant professor in the Department of Computer Science and Artificial Intelligence at the University of Granada and a cofounder of iKor Consulting, an IT services and consulting firm. Contact him at berzal@acm.org.

Cite this article: Fernando Berzal, "Java Tools and Frameworks for Web Application Development," *IEEE Distributed Systems Online*, vol. 6, no. 5, 2005.