

# Rapid application development for J2EE using Adobe® ColdFusion® 8

## Table of contents

- 1 Six issues affecting web application development
- 3 The ColdFusion approach for rapid application development
- 3 Combining the power of J2EE with the simplicity of ColdFusion
- 4 The Business case for ColdFusion 8
- 6 Conclusion: A complement to Java

For many IT managers and business-focused decision makers, today's technology trends are presenting a unique and important inflection point. While Adobe ColdFusion software is widely used throughout a large majority of organizations today, the Java™ programming language is rapidly being considered or adopted in many of these organizations. As IT professionals investigate the use of ColdFusion versus Java, it is important to understand that ColdFusion is designed to be used in conjunction with Java, leveraging the powerful capabilities that the Java runtime offers while maintaining the hallmark benefits of rapid, low-cost development provided by ColdFusion. This white paper outlines specific considerations that technology and business managers should factor into their application development infrastructure choices and investments. These considerations provide a clear picture of the benefits of utilizing ColdFusion in conjunction with Java to build and deploy web applications of varying sizes and complexity. ColdFusion 8 deployed on a Java runtime continues to provide the most favorable total cost of ownership for the majority of web applications today.

## Six issues affecting web application development

Users have needs and various skills. Organizations have different technology infrastructures. Business strategies require alternative approaches for creating, deploying, and maintaining software applications.

As a result, a broad spectrum of application development strategies has arisen for enterprises, government agencies, higher education, and other sectors. A wide range of application types exist, from commercially developed software and enterprise-class mission-critical systems that manage strategic, fundamental, high-volume processes to rudimentary, single-user, standalone applications that increase personal productivity.

One of the foremost classes of applications is web applications. These applications use Internet networking protocols and web application servers to deliver processing and functionality, typically through a simple, ubiquitous web browser. While no class of applications shares every characteristic, there are several common issues with web applications.

### **Issue #1: The developer productivity paradox**

Web applications are placing unrelenting demands on application development teams. Regardless of industry or target market, customers' and constituents' expectations for web access to information and services continues to grow. Buyers want self-service ordering. Management teams want instant access to the latest sales and financial information. As a result, there are larger backlogs for developers, so traditional tools and methodologies are feasible for only the largest-scope applications. For legacy applications, development cycles of 9 to 18 months were once the rule. Today developers need to deliver applications in a matter of weeks. Compounding the problem is that, in virtually every sector, IT budgets are under continuous pressure. Collectively, today's organizations are expected to create more applications with less time and money.

### **Issue #2: The skills shortage**

Another reality for IT managers is the dearth of elite programming talent. Staffs have varying levels of proficiency. Java, Visual Basic, HTML, and ColdFusion experience varies. If the choice is limited to Java, there is the risk of encountering a lack of skilled programmers. By standardizing on lowest common denominator tools, you forsake enterprise-class power for simplicity that your entire team can leverage. Another idea may be to use a small range of development tools that harness the varying skill levels of your team. If you do that, you may have to segment projects to maximize each contributor's skills. Elite programmers will need to effectively collaborate on complex code with other application developers in your organization.

### **Issue #3: Technical requirements of web applications**

By their very nature, web applications have shorter lifecycles and must be adaptable to more volatile business conditions and strategies. Therefore, they must be anchored by technology that provides the highest level of flexibility. These web applications—typically customer-facing systems available 24 hours a day, 7 days a week—must be reliable. As a result, you need to select the most effective technology for each developer—and achieve the correct balance between power and simplicity.

### **Issue #4: The TCO imperative**

A proper total cost of ownership (TCO) calculation should factor in such aspects as initial license costs, training time and expenses, costs to cover a skilled developer, and application maintenance and support. Less quantifiable factors, such as code comprehensibility or vendor viability, might not fit a TCO model but are also important. Having skilled developers and effective web technologies is important, but for IT managers it ultimately boils down to the TCO for a given tool, methodology, or strategy.

### **Issue #5: The role of Java in web applications**

Java is an enduring web application platform that has earned a central role in virtually every enterprise-class IT strategy. Through an open and standard component-based model, standardized integration, improved middleware development, and wider portability, Java can be the basis for more reliable, reusable, and flexible applications. Without question, Java is an essential requirement for high-end web applications.

It is important to acknowledge that, for several reasons, Java is not suitable for every developer, application, or situation. Although it is easier to learn than predecessors such as C++, achieving proficiency in Java is a costly and difficult proposition. Gartner estimates that learning Java takes 60 days of training and 10 months of full-time usage at an average cost of \$75,000 plus salary. Even if professionals were available, it would not be cost-feasible. As the complexity of the application decreases, the cost of using Java to create that application increases.

### **Issue #6: Time to market and application complexity**

To maintain alignment with business needs and expectations, application developers must be able to control complexities that result in extended development cycles. At the same time, users demand greater application sophistication and faster time to market for those applications. These conflicting goals are a key reason why Java is not suitable for every project. Java typically creates a level of code complexity that can significantly lengthen an application development cycle. Since Java is a low-level system programming language, developers typically must write many more lines of code (compared to higher level languages) to perform tasks—code that must be thoroughly tested. Java does not include the built-in services (such as search services or presentation features) and high-level abstractions that would make it accessible to less-experienced programming talent. Development teams require wizards, guides, and other productivity tools that can be implemented quickly. This “less is more” approach accelerates application development and simplifies subsequent maintenance and change cycles.

### **The ColdFusion approach for rapid application development**

Adobe ColdFusion 8 is the fastest, easiest way to build and deploy powerful Internet applications. Using proven tag-based scripting and the rich set of built-in services in ColdFusion 8, web application developers can easily harness the power of Java—without the complexity.

Many organizations today have chosen Java as the core technology to build and deploy high-end web applications. A large percentage of web applications require faster, more productive development cycles, so companies are choosing ColdFusion to extend their Java technologies. Combining ColdFusion with Java provides the best of both worlds—power and productivity.

### **Combining the power of J2EE with the simplicity of ColdFusion**

The runtime environment of ColdFusion is fully compatible with Java 2 Enterprise Edition (J2EE). This major architectural shift in ColdFusion enables it to leverage the benefits of J2EE fully while retaining the simplicity, power, and low TCO for which ColdFusion is renowned. ColdFusion 8 continues to build upon this powerful architecture.

ColdFusion applications can be deployed either on a standalone ColdFusion 8 server or on leading enterprise Java application servers, such as IBM WebSphere, BEA WebLogic, and JBoss Application Server.

Besides greater deployment flexibility, leveraging the power of the J2EE platform brings several additional benefits to ColdFusion developers:

- **Improved performance:** ColdFusion applications run as compiled Java bytecode, increasing application performance and overall server capacity.
- **Increased security and reliability:** Administrators can now isolate applications from one another by creating multiple instances of ColdFusion on a single server.
- **Greater extensibility:** ColdFusion can integrate with components built in Java, including JSPs, JSP tag libraries, servlets, and EJBs.

As a result, ColdFusion 8 is ideal for web applications that require data access, security, and scalability in a fast development cycle—systems for which the complexity of Java is not pragmatic—while leaving your specialized Java team free to focus on larger, more strategic systems.

At the higher end, where you might want to create a complex system with redundancy and failover, ColdFusion 8 is the ideal complement to Java. It provides complexity-shielding abstractions, tools, and guides to enable more developers to contribute to the development cycle and to collaborate with skilled team members who are knowledgeable in Java.

## **The business case for ColdFusion 8**

### **Advantage #1: Time to adoption**

A key strength of ColdFusion 8 is how quickly developers can leverage the power of J2EE and take advantage of the productive ColdFusion environment. The intuitive, tag-based ColdFusion Markup Language (CFML) requires fewer lines of code by handling low-level programming tasks automatically and simplifying code reuse—a paradigm that is very similar to languages with which web developers are already familiar. Typically, a new developer can be proficient in ColdFusion with just one week of training and four months of full-time use—far less than Java would require.

### **Advantage #2: Time to market**

The fully integrated application services of ColdFusion 8 eliminate significant amounts of code that Java programmers would otherwise have to write. Estimates show that a ColdFusion application requires 88% fewer lines of code than an identically functioning Java application, and 54% fewer lines than an identically functioning Microsoft .NET application. That translates into faster time to market and lower maintenance costs.

### **Advantage #3: Leveraging the power of J2EE**

The innovative architecture of ColdFusion 8 combines the scalability, reliability, and power of the Java platform with the simplicity of web scripting. You can also take advantage of complete extensibility through custom tag libraries and reusable components—as well as full support for JSP, servlets, and EJBs. ColdFusion 8 includes options for native deployment to give you the benefits of J2EE without forsaking the productivity and time to market advantages of ColdFusion 8.

### **Advantage #4: Native ColdFusion services**

Native services offered by ColdFusion 8 boost developer productivity, requiring far less code to be written as compared to Java. For example, developers can leverage the powerful, built-in structured business reporting, document generation, Verity text search, graphing/charting, and rich forms into applications. ColdFusion connects to a wide variety of back-end data sources and protocols through high-level abstractions that leverage underlying Java integration application programming interfaces (APIs). ColdFusion also provides rich presentation services using Adobe Flash® technology for displaying information to many different client types now accessing web applications.

### **Advantage #5: Extensibility**

Leveraging existing infrastructure is important to your business. ColdFusion 8 enables you to create or consume web services easily; invoke COM, CORBA, .NET assemblies, or Java objects; execute servlets; access LDAP and other security mechanisms; execute stored procedures; access FTP files; generate e-mail; invoke other web pages; access mail servers; and much more. ColdFusion helps you extend existing systems and technologies, enabling you to take advantage of your other technology investments. ColdFusion supports a wide range of deployment options—including leading Java application servers such as Macromedia® JRun™ from Adobe, IBM WebSphere, BEA WebLogic, and JBoss Application Server—in addition to running on leading operating systems.

### Advantage #6: Lower TCO

Collectively, the previous advantages translate into another compelling benefit: lower total cost of ownership. Quite simply, you can do more with less. By adding ColdFusion 8 to your technology suite, you can solve application problems faster and for far less overall investment. Although you won't use it for all your applications, ColdFusion 8 helps you deliver a significant percentage of your applications faster and at a lower cost than building them in native Java.

In application development, TCO is a critical metric. Defining and measuring TCO remains elusive and open to some debate. As IT managers grapple with investment decisions concerning their development strategies, the following table can be a useful starting point for making a decision. The numbers included in the table represent a small-to-medium-size application project. Enter your own estimates to determine your TCO business case.

DESCRIPTION	JAVA		COLDFUSION 8		YOUR ESTIMATE	
Cost of person-month	\$8,500		\$6,000			
Technology training	2	Person-months	0.25	Person-months		Person-months
Project scoping, specification, and design	3	Person-months	1	Person-months		Person-months
Development cycle (includes coding and testing)	6	Person-months	2	Person-months		Person-months
Annual maintenance	2	Person-months	0.5	Person-months		Person-months
Total IT manpower	13	Person-months	3.75	Person-months		Person-months
First-year IT labor costs	\$110,500		\$23,625			
Annual cost of developer equipment	\$500	Per-person-month	\$300	Per-person-month		Per-person-month
Total expected first-year costs	\$117,000	\$26,100				
Code profile	100,000	Lines of code	20,000	Lines of code		Lines of code

### Advantage #7: Vendor commitment/stability

When you accept a project, you try to minimize the risk of failure? ColdFusion 8 mitigates this risk in several ways. First, ColdFusion 8 is faster. Shorter project horizons are less risky. It's a lower cost alternative, which reduces your risk. ColdFusion has been validated not only by industry analysts and experts, but also by a broad and thriving developer community that has embraced the technology for many years. Finally, it's regularly upgraded by Adobe, a large, stable software company with a lengthy track record of success and innovation. Selecting Adobe ColdFusion 8 is a safe decision.

## **Conclusion: A complement to Java**

Ultimately, it's not a case of ColdFusion versus J2EE. Instead, view ColdFusion 8 as an essential complement to any organization's Java strategy. It should be the first choice in a large percentage of your applications when the level of application complexity does not merit the use of Java. Except for instances in which you require low-level access to define and tweak applications, ColdFusion 8 is often a better choice for web application development, combining the power of Java with productivity and lower total cost of ownership. For more information on Adobe ColdFusion 8, and its role with Java and J2EE check out the ColdFusion 8 datasheets and white papers page at [www.adobe.com/products/coldfusion/whitepapers](http://www.adobe.com/products/coldfusion/whitepapers) or the Java and J2EE Architecture page on the ColdFusion Developer Center at [www.adobe.com/devnet/coldfusion/java\\_j2ee.html](http://www.adobe.com/devnet/coldfusion/java_j2ee.html).



**Adobe**

**Adobe Systems Incorporated**  
345 Park Avenue  
San Jose, CA 95110-2704  
USA  
[www.adobe.com](http://www.adobe.com)

Adobe, the Adobe logo, ColdFusion, Flash, JRun, and Macromedia are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Java is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries. All other trademarks are the property of their respective owners.

© 2007 Adobe Systems Incorporated. All rights reserved. Printed in the USA.  
95009826 7/07