Introduction

by Jim Inscore and Nicholas Kassem

SINCE its introduction more than two years ago, the Java 2 Platform, Enterprise Edition (J2EE), has rapidly established a new model for developing distributed applications. This model is based on well-defined components that can automatically take advantage of sophisticated platform services. These components can be developed according to standard guidelines, combined into applications, deployed on a variety of compatible server products, and reused for maximum programmer productivity. This model is intended to both standardize and simplify the kind of distributed applications required for today's networked information economy. The success of the J2EE platform is in large part due to the success of this model.

Today, all leading application server and enterprise information system vendors have adopted the J2EE standard and introduced products based on the J2EE platform specification. Application architects and developers have come to rely on the J2EE standard to help them solve the various design challenges that face them day to day.

While the fundamentals of the J2EE platform are relatively easy to describe, mapping these features to architectural issues in the design of distributed applications requires deeper understanding and careful decision making. Although the J2EE standard offers a simplified programming model compared to previous alternatives, the platform isn't monolithic. Certain features require that architects and developers weigh their options before making design decisions and be prepared to re-think those decisions as they uncover new challenges. That, in turn, requires some understanding of the design motivations behind the platform and of the trade-offs involved in applying specific design features to a specific architectural problem.

Different implementations of the J2EE platform may provide distinguishing characteristics that improve their performance or development ease in particular areas. However, the level of abstraction provided by the J2EE standard enables common themes to be developed, explained, and explored and certain common design guidelines to be developed. That's what Java BluePrints is all about. It answers questions like:

- What's the best way to apply each type of J2EE component?
- Where does it make sense to use Java servlets and where to use JavaServer Pages?
- What's the best way to factor business logic between entity beans and session beans?
- How do you choose between container-managed and bean-managed persistence when using entity beans?
- What are the design and performance trade-offs between choosing a distributed architecture and one based on local interfaces?
- In this increasingly security-conscious world, how do you design distributed applications to be accessible to users who need them and secure from unwanted intrusion?

Before the remainder of this book takes you more deeply into these and other details of J2EE application architectures, this chapter gives you a look at some of the design motivations behind the J2EE platform. It describes the high-level benefits of the J2EE platform and discusses ways that using it as the underlying architecture for distributed applications makes sense for a variety of application requirements.

1.1 Challenges of Enterprise Application Development

Timing has always been a critical factor when organizations adopt new technologies, and the accelerated pace of the information-driven business model puts greater emphasis on response times. Organizations need to be able to project enterprise systems into various client channels, and to do so in a way that's reliable, productive, and capable of sustaining frequent updates to both information and services. The principal issue is how to keep up with today's business challenges—whatever

they may be—while maintaining and leveraging the value of existing information assets. In this environment, timeliness, productivity, security, and predictability are all absolutely critical to building and maintaining momentum. A number of factors can enhance or impede an organization's ability to deliver custom enterprise applications quickly and to maximize their value over their lifetime.

1.1.1 Programming Productivity

The ability to develop and deploy applications is key to success in the information economy. Applications must go quickly from prototype to production and must continue to evolve even after they are deployed.

Productivity is thus vital to responsive application development. Providing application development teams with standard means to access the services required by multitier applications and standard ways to support a variety of clients can contribute to both responsiveness and flexibility.

The current divergence of technologies and programming models is a destabilizing factor in Internet and other distributed computing applications. Traditional Web technologies such as HTML and Common Gateway Interface (CGI) have provided a mechanism for distributing dynamic content, while back-end systems such as transaction processors and database management systems have provided controlled access to the data to be presented and manipulated. These technologies present a diversity of programming models: some based on well-defined standards; others on more ad-hoc standards; and others still on proprietary architectures.

With no single application model, it can be difficult for teams to communicate application requirements effectively and productively. As a result, architecting applications becomes more complex. What's more, the skill sets required to integrate these technologies aren't well organized for effective division of labor. For example, CGI development requires coders to define both content and layout of a dynamic Web page.

Another complicating factor in application development time is the choice of clients. While many applications can be distributed to Web browser clients through static or dynamically generated HTML, others may need to support a specific type of client or to support several types of clients simultaneously. The programming model needs to support a variety of client configurations, with minimum effect on basic application architecture and on the application's core business logic.

1.1.2 Integration with Existing Systems

Much of the data of value to organizations has been collected over the years by existing enterprise information systems. Much of the programming investment resides in applications on those same systems. The challenge for developers of enterprise applications is how to reuse and commoditize these existing information assets.

To achieve this goal, application developers need standard ways to access middle-tier and back-end services such as database management systems and transaction monitors. They also need systems that provide these services consistently, so that new programming models or styles aren't required as integration expands to encompass various systems within an enterprise.

1.1.3 Freedom of Choice

Application development responsiveness requires the ability to mix and match solutions to come up with the optimum configuration for the task at hand. Freedom of choice in enterprise application development should extend from servers to tools to components. The wide range of J2EE compatible solutions available today and in the future ensures the maximum freedom of choice.

The availability of choices among server products gives an organization the ability to select configurations tailored to their application requirements. It also provides the ability to move quickly and easily from one configuration to another as internal and external demand requires.

Access to the appropriate tools for the job is another important choice. Development teams should be able to adopt new tools as new needs arise, including tools from server vendors and third-party tool developers. What's more, each member of a development team should have access to tools that are most appropriate to their skill set and contribution.

Finally, developers should be able to choose from a ready market of off-the-shelf application components to take advantage of external expertise and to enhance development productivity.

1.1.4 Response to Demand

When designing large-scale distributed applications, both availability and scalability are key considerations. The more easily and automatically that an application can handle changes in use patterns and system configurations, the better. Systems that

require any redesign, recoding, or redeployment to achieve either availability or scalability will limit flexibility and diminish expected performance.

To scale effectively, systems need to be designed to handle multiple client interactions with ease. They need mechanisms for efficient management of system resources and services such as database connections and transactions. For highest availability, they need access to features such as automatic load balancing and failover, without any effort on the part of the application developer. Applications should be able to run on any server configuration appropriate to anticipated client volumes and to easily switch configurations when the need arises. Support for clustered application deployment environments contributes to achieving many of these goals.

1.1.5 Maintaining Security

More than ever, information systems security is on the minds of IT managers and system architects. That's because protecting information assets to maximize their value can jeopardize that very value. Traditionally, IT departments have been able to maintain a relatively high level of control over the environment of both servers and clients. When information assets are exposed in less-protected environments, it becomes increasingly important to maintain tight security over the most sensitive assets, while allowing seemingly unencumbered access to others.

One of the difficulties in integrating disparate systems is providing a unified security model. Single sign on across internal application and asset boundaries is important to creating a positive user experience with the applications. Security needs to be compatible with existing mechanisms. In cases where customers need to access secure information, the mechanisms need to maintain high security (and user confidence) while remaining as unobtrusive and transparent as possible.

1.2 The Platform for Enterprise Solutions

The J2EE platform represents a single standard for implementing and deploying enterprise applications. During its first two years, the J2EE standard's success has transformed the marketplace for distributed computing products. This success is largely due to the fact that the J2EE platform has been designed through an open process, engaging a range of enterprise computing vendors to ensure that it meets the widest possible range of enterprise application requirements. As a result, the J2EE platform addresses the core issues that impede organizations' efforts to main-