



Chapter 8

Onward

Java provides a solid infrastructure for component development and deployment. The next challenge for the software industry is to build a higher-level component infrastructure using standards such as XML.

8.1 Where We Are

This book explains how to develop components and assemble them within a Java virtual machine. Viewed at this level, component development is driven by two key concepts: loaders and types.

Java's class loader architecture provides the means to locate and load components into a process. When they are used carefully, class loaders guarantee security, manage versioning, prevent name collisions, and enable side-by-side deployment of changing components. Loaders also provide a hook from which custom behavior or metadata can be extracted from or added to classes. Even JNI is just a special loader that loads non-Java components in a local process.

Loaders use Java's type information to validate dependencies between classes at runtime. Equally important, containers and other service code can manipulate components via their type information, in order to provide services such as serialization, object/relational mapping, XML data binding, and dynamic proxies.

Taken together, loaders and type information provide great control over the bind mode of application decisions. In addition to compile-time binding and runtime dynamic binding, you can generate the bindings you need, which will give you runtime flexibility and static-bound performance.





8.2 Where We Are Going

Enterprise applications span multiple processes and machines, and they must be able to communicate with disparate hardware and software over the Internet. The current buzzword in enterprise application development is “web applications.” Just like standalone Java applications, web applications need to be assembled from components. Therefore, all of the concerns of this book apply to web applications, but with a new twist: web applications need a longer reach.

Web applications must enable communication that reaches all products, and all vendors. No one vendor will define the protocols of web applications; they require industry standards implemented atop solid component platforms. XML will provide the standards, and Java will provide the platform.

As a Java developer, there are several things you should do to prepare for a web application world:

1. Learn servlets. A Java servlet is a network class loader. Servlets load code in some other process across the network in response to a request.
2. Learn XML and the Infoset. XML provides the conceptual model and the serialization format for web application data.
3. Learn XML schemas. Schemas add type information to XML data.
4. Learn XPath. XPath will be the query language of web applications.
5. Learn the Web Services Description Language (WSDL). Along with schemas, WSDL is the other half of XML type information. Where schemas describe data, WSDL describes functionality, that is endpoints and message formats.
6. Participate in the Java Community Process (JCP). Most of the technologies of web applications are too new to have standard Java APIs, but that will change swiftly. The JCP will define APIs so that web applications do not have to be programmed directly.

Java is well positioned to be a dominant platform for web applications. Java’s class loaders, type information, and security model will map well to the loaders and type systems of web applications, however those may evolve.





8.3 Resources

There are a huge number of XML books already in print. However, the ways in which XML technologies will combine to enable web services are not completely understood at the present time. To keep track of the XML world, begin at the home page for the World Wide Web Consortium [W3C]. To join in the discussion of how Java and XML should work together, begin with the home page for the Java Community Process (JCP).



