Chapter 11: Packaging and Deployment

Similar to a physical world supply chain, Java developers are able to deploy their code across any device using packaging. This process makes it easy for users to be able to run the java programs compiled into a package, without ever seeing the underlying code associated with those programs. After the code is developed, the application is then compressed for production deployment. Inside these compressed bundles, is a numerous amount of Java classes that point to a specific file in the directory. One format of packaging a java application is called Java Archive, or JAR. All of these class files and their related assets are bundled into a ZIP file. Inside a JAR also resides a manifest file called "METAINF/MANIFEST.MF", which defines the JAR's attributes for unpacking. JARs are included in standard JDKs, and JARs specifications are documented in the JDK documentation. JEE also has documentation that provides uses for JARs. Next, the book talks about Web Modules, which are as small as a deployable application can be in regards to JEE. These contain mostly static content, but also include web resources and components. Upon development of the web modules, they can then be packaged into a web archive, or WAR. Similar to a JAR, this contains a manifest file that handles the attributes of the module, and is present in the WEB-INF directory. Inside that directory, web.xml is usually present and is used to layout the structure of the web application as well as point to resources required by said application. This format is known for being flexible to the developers' needs, and can be easily deployed through web containers. Tomcat and Jetty are a few examples of popular web containers that are used across the globe. Moving on, an enterprise archive (EAR) holds multiple other archives, typically JAR and WAR files, and a manifest file called application.xml. Each of these archives are given a security role, and the EAR is responsible for

authenticated the roles of these archives. Because EAR is intended for enterprise environments, they must be used with an enterprise application server. Using this server opens up the JEE specification to many more possibilities than a simple web module. Popular examples of application servers include JBoss, WebSphere, and WebLogic Server. RAR, or resource adapter module, is a less used option for archiving java applications and allows connectivity to Enterprise Information Systems (EIS). These systems often run legacy services such as ERP, MainFrame, Queue, etc. A RAR provides a more open alternative to JBDC drivers, but functions in almost the same way without being limited to accessing relational databases.

These types of archives are not just limited to Java development. For example, there are multiple tools that allow JARs and WARs to be created from other frameworks and resources. The Play framework allows for the implementation of Scala resources to reside inside the created WAR. Warbler is another popular application that allows developers to create JARs and WARs based on ruby code. In client-server web development, Java and JEE packaging frameworks serve their purpose well and have proven their usefulness in speeding up development and delivery of the application to the users. For many years, there has not been much innovation in the world of JEE and Java packaging because the framework just works and hasn't really needed to change to provide a better experience for developers. Packaging can also keep some hefty resources off of the user's machine and handled on a dedicated web server. In this case, the WAR deployed to the user may contain a reference to a server WAR that contains the API, HTML, CSS, and JS of the desired application. Multiple EARs can be deployed to keep the release organized and to separate web containers, and two of the same EARs can be deployed to other modules.