Let’s start with buzzwords. J2EE provides:

- A component-based approach to design, development, assembly, and deployment
  - Many components are reusable
- A multi-tiered distributed application model
  - Means that different parts of the applications naturally reside on different machines
- A unified security model
- A flexible transaction control
- Web services support through integrated data interchange on XML-based open standards and protocols
  - Communication among components is standardized

Typical tiers, components, and mapping to hardware

Client tier

- J2EE applications support clients of different kinds:
  - Web clients
    - A Web browser that is capable of displaying pages in mark-up languages (HTML, XML, etc)
    - Do not perform complex computations -> often called thin clients
  - Applets
    - Small client applications that execute in the JVM run by the Web browser
    - May perform moderately complex computations
  - Application clients
    - Applications that directly communicate with objects in the business tier

Web tier

- Servlets
  - Java classes that dynamically process requests and construct responses
  - Use the HTML protocol
  - The Java version of CGI scripts
- JSPs (Java Server Pages)
  - Text-based documents that combine mark-up structure and Java code
  - Essentially, are used by J2EE to generate servlets
- Code components (classes, often JavaBeans)
  - Used by JSPs and servlets

Business tier

- Called this way because it contains logic for the application
- Enterprise Java Beans (EJBs):
  - Concentrate on processing of data from client programs and the storage system
- Three kinds of EJBs:
  - Session beans
    - Supports transient “conversations” with clients
    - Removed when the client finishes the communication
  - Entity beans
    - Represent persistent data (stored in the DB)
  - Message-driven beans
    - A more complex type of session beans, using Java Message Service

Lecture topics

- J2EE architecture
- Description of the project
J2EE containers

- Writing a multi-tiered distributed application from scratch is a daunting task
  - Lots of features have to be supported on the low-level
    - Multi-threading
    - Persistence
    - Transactions
    - Resource pooling
- Containers in the J2EE architecture provide this low-level support for user-defined components
  - A container defines an interface between a component and the low-level platform-specific functionality that supports the component
  - This is why components have to implement specific interfaces
- Container services are configurable
  - The same application can behave differently depending on where it is deployed

Examples of container services

- The security model allows configuration of a Web application so that only selected kinds of users can access a specific functionality
- The transaction model lets you treat a number of method calls as a single transaction
  - Effects of method calls can be unrolled
- Naming and directory lookup services
- Remote connectivity
  - Call methods on remote objects as if they were in the same virtual machine

Types of containers

- EJB container
  - Manages execution of EJBs
- Web container
  - Manages execution of servlets and EJBs
- Application client container
  - Manages execution of application client components
- Applet container
  - Manages execution of applets
  - Consists of a browser and a plug-in

Packaging

- A J2EE application is delivered in an Enterprise Archive (EAR) file
  - A JAR file with extension .ear
  - Contains J2EE modules
- A J2EE module consists of
  - One or more J2EE components for the same container type
  - A deployment descriptor
    - An XML document describing deployment settings
    - E.g., for an EJB, can specify access authorizations
  - Four types of J2EE modules:
    - EJB modules (.jar extension)
    - Web modules (.war extension) - JSPs, servlets, HTML, GIF
    - Resource adapter modules (.rar extension) - classes, libraries, documentation, etc.
    - Application client modules (.rar extension)

J2EE development roles

- J2EE product provider
  - The companies that make/sell the J2EE platform
    - Operating system, database system, application server, etc.
- Tool provider
  - Creates development and packaging tools used by the developers, assemblers, and deployers
- Application component provider
  - Creates and packages Web components, EJBs, applets, and application clients
- Application assembler
  - Assembles the component JAR files into an EAR file
- May edit deployment descriptors for individual components
- Application deployer and administrator
  - Configures and deploys the application
  - Administers the infrastructure on which the application runs
  - Is responsible for system-wide security settings

J2EE APIs

- EJBs
- Java Database Connectivity (JDBC)
  - Database connectivity (lets you invoke SQL commands)
- Servlets
- JSP
- Java Message Service (JMS)
  - For sending asynchronous messages among components
- Java Naming and Directory Interface (JNDI)
  - Directory operations, e.g. search for objects on the network using their attributes
- Java Transaction API (JTA)
  - Enables transactions (commit, rollback functionality)
J2EE APIs, cont.

- JavaMail
  - For sending email notifications
- Java API for XML Processing (JAXP)
  - XML parsing and manipulation
  - Supports DOM, SAX, XSLT
- Java API for XML Registries (JAXR)
  - Support for two Web registry standards
- Java API for XML-Based RPC (JAX-RPC)
  - Implements XML-based remote procedure calls
  - Low-level API used by JAX-RPC

J2EE APIs, cont.

- J2EE Connector Architecture
  - Supports creation of resource adapters
  - Used to interoperate with EISs
- Java Authentication and Authorization Service (JAAS)
  - Allows configuration managers to create groups of users and assign subsets of the application functionality to them

XML

- eXtended Markup Language
- A standard for structured text documents
- Tags are used to classify data
  - Attributes provide additional information inside tags
- All configuration files (deployment descriptors) in J2EE use XML

```<message to="evil.overlord@pit.hell" from="evil.minion@raise.hell" subject="Mission accomplished">
  <text>
    SCO victorious.
  </text>
</message>```

Web modules

- Web resources are J2EE Web components and static content files (e.g., images)
- A web module is the smallest deployable and usable unit of:
  - Web resources
  - Server-side utility classes (including EJBs)
  - Client-side classes
- Web modules have a well-defined structure
- Web modules can be deployed as an unpacked file structure or packaged in a WAR file

Dynamic reloading of Web applications

- Downtime for enterprise applications can be very costly
- Yet, a typical software upgrade requires a re-install
- J2EE enables dynamic reloading: upgrades are done on a running application
  - Can change deployment descriptors
    - E.g., introduce a new type of user, with associated access rights
  - Can change code!
    - JSP, .class files, etc.
- The application server checks for changes and re-deploys the application automatically

Typical Web module structure

Dynamic reloading of Web applications

- Downtime for enterprise applications can be very costly
- Yet, a typical software upgrade requires a re-install
- J2EE enables dynamic reloading: upgrades are done on a running application
  - Can change deployment descriptors
    - E.g., introduce a new type of user, with associated access rights
  - Can change code!
    - JSP, .class files, etc.
- The application server checks for changes and re-deploys the application automatically
Servlets

- A servlet is a Java class using the request-response programming model.
- Commonly used with the HTTP protocol.
  - Each request contains a URL identifying a Web component or a static object (HTML page, image file, etc.)
  - The J2EE server converts the request to an HTTP request object and gives it to the servlet identified by the request URL.
  - The servlet fills in an HTTP response object.
  - The J2EE server converts the HTTP response object to an HTTP response and sends it to the client.

Servlet lifecycle

- Lifecycle of servlets is controlled by the servlet container.
- When the container receives a request mapped to a servlet, it:
  - Checks if the servlet exists. If not, the container will:
    - Load the servlet class.
    - Create an instance of the servlet class.
  - Call the init method of the servlet instance.
  - This method is defined by the servlet programmer and contains any code necessary to load the initial data used by the servlet.
  - Invokes a service method of the servlet, passes the request and response objects to this method.
- Containers can remove servlet instances.
- Application code can have listener objects that are notified of events in servlet lifecycle.
- E.g., you can log every request to a specific servlet.

Servlet service methods

- For HttpServlet, methods doGet, doPost, doPut, doDelete, doOptions, doTrace
- public class BookDetailsServlet extends HttpServlet
- { public void doPut(HttpServletRequest request, HttpServletResponse response)
- { try {
- String bookId = request.getParameter("bookId");
- if (bookId != null)
- { String title = request.getParameter("title");
- BookDetails bd = new BookDetails(title);
- try { bd = new BookDetails(title); } catch (BookNotFoundException ex) { throw new ServletException(ex); }
- bd.print();
- } catch (IOException | ServletException ex) { throw new ServletException(ex); }
- } catch (IOException ex) { throw new ServletException(ex); } } }

JSPs

- JSP technology lets you specify scripting language commands inline with formatted text.
- Used by the J2EE server to generate responses to clients’ requests.
- Essentially, the server executes the scripting commands in a JSP page and inserts the produced output in the dynamic page.
- Format the text (HTML, XML) is left as is.
- The JSP 2.0 standard uses Java as the scripting language, but in future other languages will be supported.
- JSPs are converted into servlets in the application server.

JSP page composition

- A JSP is composed of:
  - Template data
    - Static presentation content (HTML, XML)
  - Directives
    - JSP elements that are interpreted at translation time
      - Tag a JSP compiler to include other files in the compilation of a JSP.
      - Define attributes about the JSP page being translated.
      - Define libraries of custom elements used in the JSP.
  - Scripting elements
    - JSP elements that define
      - Variable and method declarations.
      - Expressions to be evaluated.
      - Scripts (blocks of commands).
  - Action elements
    - Tags, e.g. forward requests to other resources, look up JavaBean objects, etc.

JSP example

```jsp
<%@ page session="false" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<h1>Hello, world!</h1>
```

...
JavaBeans

- A JavaBean is a Java class that follows certain design conventions
- JavaBeans have properties
  - A property can be read-only, write-only, or read-write
  - A property does not have to correspond to a field
  - For each readable property, the bean must have a method of the form PropertyClass getProperty() {...}
  - For each writable property, the bean must have a method of the form PropertyClass setProperty() {...}
- A JavaBean must have a constructor with no arguments
- Special JSP tags make it convenient to use JavaBeans with JSPs

EJBs

- Beans on steroids
  - EJB container adds a lot
- Types of EJBs
  - Session beans
  - Entity beans
  - Message-driven beans

Session beans

- A session bean represents a single client inside the J2EE server
- The client invokes the session bean’s methods
- A session bean is not shared and not persistent
- Two types of session beans
  - Stateful
    - The state of the bean is preserved as its fields
  - Stateless
    - Although a bean may have fields, they are not used to hold data across method invocations on this bean
    - Usually, these are more efficient than stateful beans