Enterprise JavaBeans (EJB)
What are EJBs?

- They are components that can be connected to form a system
  - They can represent data
  - They can represent behavior
- Usually, EJBs fall into only one of these categories
- They are typically used in the server tier
- EJBs can be persisted
- EJBs can interact with other EJBs
Advantages of EJBs

- EJBs are reusable components
  - Can be reused in different parts of the system
  - Can be packaged into libraries and sold
- EJBs Can be combined visually using development IDEs
  - E.g. Visual Age, Visual Café
- EJBs provide convenient abstractions so it do not require you to write:
  - Multi-threaded, multiple access code
  - Database access code (e.g. JDBC)
  - Network communication code (i.e. it uses RMI) for client/server communication
  - Network communication code for EJB to EJB communication
  - Transaction management code

- EJBs from different businesses can interact easily
  - This is because of their well-defined interfaces
EJB in the Big Picture of J2EE

Where is EJB?
EJB™ Role: Application Assembler

ejb-jar (.jar)

Deployment descriptor

Deployment tools

Web jars (.war): servlets, JSP...

Application Jar (.ear):
EJB Communication

- EJBs use IIOP as the wire protocol
  - Therefore, EJBs are compatible with RMI (i.e., RMI over IIOP) and CORBA libraries
  - Thus, you could write an EJB client in any language that supports CORBA
EJB as Client/Server Middleware

- Think of EJBs as just another Client/Server middleware like RMI, CORBA, or Web Services
  - The EJB instance itself is the server
  - EJBs have clients (objects that call its methods)
- One complication is that EJB clients can be:
  - Java Applets and Applications (using RMI or CORBA)
  - Non-Java applications (using CORBA)
  - JSPs and Servlets (using RMI or CORBA)
  - Other EJBs (using RMI or CORBA)
In enterprise systems, EJB clients are usually: Servlets, JSPs, or Other EJBs
EJBs & Multi-Tiered Architectures

- Applet
- Web Page
- EJBs
- Servlets & JSPs
- 3rd Party EJBs
- Database
How EJBs Change Things

- EJBs are most suitable for developing business logic and data manipulation
  - If all of the business logic operations and data manipulations are done using EJBs, the JSPs and Servlets will be focused mainly on displaying the results of those operations

- Session EJBs: Used to represent system behavior (i.e. business logic)
  - e.g. Storing products to purchase in the shopping cart

- Entity EJBs: Used to represent & manipulate system data
  - e.g. Finding products that match a search term
Application Servers

- Containers where EJBs (and JSPs and servlets) are executed
- Provide EJB functionality, including:
  - Persistence through databases (using JDBC)
  - Transactions (using Java Transaction Service)
- Can provide advanced features, including:
  - Load balancing
  - Database connection pooling
- Here are the major application servers:
  - SJS AP, WebLogic (BEA), Internet Application Server or iAS (Oracle), WebSphere (IBM)
Alternatives to EJBs

- Web Services are one of the technologies competing with EJBs
  - Web services use the SOAP protocol to exchange information with some server
    - SOAP uses an XML format to exchange request and response information via HTTP
    - Due to SOAP's well-defined protocol, Web Services can be used to exchange information between businesses (B2B)
  - Web services provide one or more remote method that can be accessed easily from other applications
Alternatives to EJBs

- CORBA objects provide some functionality similar to EJBs:
  - Persistence (of CORBA object data)
  - Transactions (between CORBA objects)
  - Security (between CORBA objects)

- CORBA and EJBs are closely related, in fact, they use the same wire protocol:
  - IIOP

- In some sense, EJBs can be considered to be an enhanced version of CORBA
  - Except that EJBs can only be created in Java
EJB Types

Entity and Session Beans

Session Beans
- Statefull
- Stateless

State Management

Entity Beans
- Bean-Managed
- Container-Managed

Persistence Strategies
Types of Enterprise Beans

- **Session beans:**
  - Also called business process objects
  - They represent the business logic of the system
  - Their lifetime is usually an entire session
    - When a session is done, the session bean expires
    - i.e. Session bean instances exist as long as a specific user is using the system

- **Entity beans:**
  - Also called business data objects
  - They represent persistent data
    - Often the data persistence is managed through a database, using JDBC
Subtypes of Session Beans

- **Stateful:**
  - Used for operations that require multiple requests to be completed
  - Maintain data between requests

- **Stateless:**
  - Used for operations that can be performed in a single request
  - Do not maintain persistent data between subsequent requests from a given client
Entity Beans Explained

- Entity beans represent data in the system
  - In addition, entity beans are used to search for, modify, create and delete data
  - Usually, this data resides in a relational database
  - Each entity bean typically represents a single row in some database table

- An entity bean instance exists as long as the data is being used
  - When the EJB client is done with the instance, the entity bean instance usually returns to a bean pool

- The client for an entity bean is typically a session bean, since behavior usually involves the manipulation of data
Subtypes of Entity Beans

- **Bean-managed persistence:**
  - The entity bean handles its own persistence
    - Often via JDBC (or SQL/J) to a database
  - The bean author is required to write persistence-management code into the bean code itself

- **Container-managed persistence:**
  - The entity bean’s persistence is automatically maintained by the EJB container
  - This is the easiest way, and often EJB containers do a better job because they provide extra features like connection pooling, load balancing, etc.
  - This method is known to be extremely reliable, since CMP code is usually well tested
  - Persistence logic is kept in “declarative code” in the EJB deployment descriptor
Session and Entity Beans

Entity and Session Beans—Typical Architecture
An EJB Autopsy

- The remote interface
  - Describes the interface provided to EJB clients
- The enterprise bean class
  - The implementation of the bean
- The home interface
  - Describe how client can create, find, and remove EJB instances
EJP Autopsy

EJB Architecture
Where are Stubs?

RMI over IIOP
EJB™ Role: Bean Provider
The Remote Interface

- Describes the interface provided to EJB clients
- Must extends javax.ejb.EJBObject
- This interface usually provides a number of accessor methods (getters and setters) for the bean’s fields, as well as all business methods
The Enterprise Bean Class

- The implementation of the bean
- This is where the methods exported in the remote interface are defined
- Business logic and data operations occur here
- EJB classes must implement one of the following interfaces: javax.ejb.SessionBean, javax.ejb.EntityBean
The Home Interface

- The home interface describes any methods not requiring access to a particular bean instance
  - Methods for creating, finding, and deleting bean instances
- Must extend javax.ejb.EJBHome
EJB Naming Conventions

- Enterprise bean class:
  - `<name>Bean`, e.g. `CustomerBean`
- Home interface:
  - `<name>Home`, e.g. `CustomerHome`
- Remote interface:
  - `<name>`, e.g. `Customer`
EJB Client Operation

- An EJB client uses an EJB by first locating its home object
  - The methods on this home object are declared in the home interface
  - The home object is located using JNDI
    - The client tells JNDI what name the EJB goes by, and JNDI gives a home interface for that EJB
- Once a home object is obtained, the client calls some home methods to access the EJB
  - e.g. The client may call “create” to create a new instance, “remove” to delete an instance, “findXYZ” to search for EJBs.
References
