L06c. Enterprise Java Beans

What is a Java Bean?

- A Java Bean is a reusable software component that has many Java objects in a bundle so that the Java Bean can be passed around easily from one application to another application for reuse.
- Some of the challenges for different enterprises to provide together a common service are to maintain and evolve:
  - Service Interoperability and compatibility.
  - Service Scalability.
  - Service Reliability.
  - Service cost of operation.
- Such cross-enterprise services (e.g. airline reservation system, gmail, internet search engine, etc.) are referred to as Giant Scale Services (GSS).
- The Object Technology facilitates the following:
  1. Structuring of an operating system at different levels
  2. Structuring of Distributed Services, providing customers various options based on cost, convenience, guarantees, etc.
  3. Handling resource conflicts that might occur between simultaneous requests across space and time coming from several different clients.

N-Tier Applications:

- Distributed Giant Scale Services are also called as N-tier applications because the software stack of an application comprises of several different layers:
  - Presentation Layer: Responsible for painting the browser screen and generating the web-page based on your request.
  - Application Layer: Responsible for the application logic that corresponds to what the service is providing.
  - Business Logic Layer: Corresponds to the way fees/prices are calculated.
  - Database Layer: Accesses the database that contains the information that is queried and updated based on the user request.

- N-tier applications must handle the following:
  - Persistence of data/actions.
  - Transaction properties.
  - Data caching.
  - Clustering related services/data.
  - Security.
  - Concurrency: Exploit parallelism across several simultaneous requests.
  - Reusing components: Portions of the application logic are reused in different components in order to serve simultaneous requests from several different clients.

Structuring N-Tier Applications:

- A Java Bean is a unit of reuse and contains a bundle of Java Objects to provide a specific functionality, e.g. a Java Bean may provide the shopping cart functionality.
- A Container is a Protection Domain implemented in a Java Virtual Machine (JVM) and it packages and hosts a related collection of Java Beans to provide higher-level functionality.
- An Application Service is constructed by using multiple Containers, typically present on different servers and used in a distributed manner.
- Mnemonic: Java Objects → Java Beans → Containers → Application Service.
- The JEE (Java Enterprise Edition) framework has 4 containers for constructing an application service:
  - Client Container: Resides on a web server and interacts with client browser.
  - Applet Container: Resides on a web server and interacts with client browser.
  - Web Container: Contains a Presentation Logic. Responsible for creating web pages to be sent back to the client browser.
  - EJB Container: Manages the Business Logic that decides what needs to be done to carry out the client browser requests. It also communicates with the Database server to read/write data corresponding to the client browser requests.

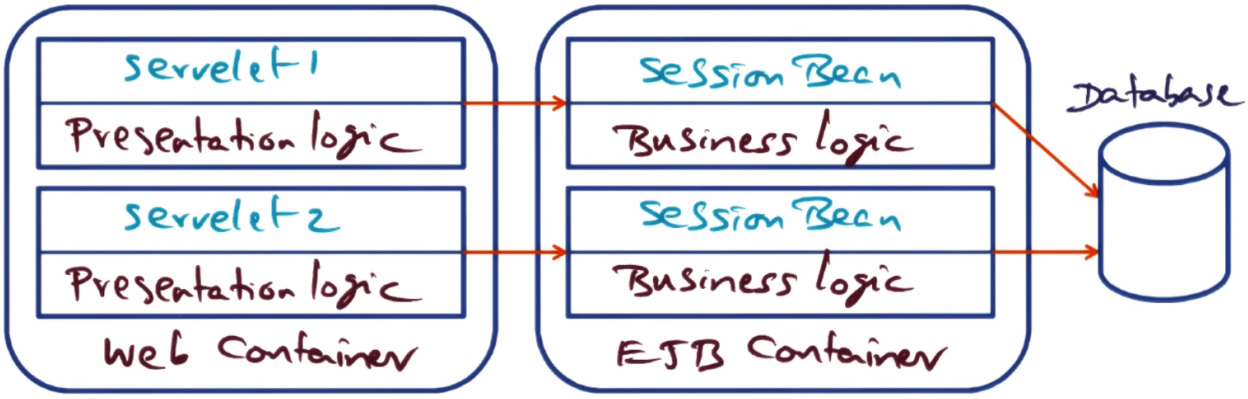
Types of Java Beans:

- Entity Bean: Persistent Objects with Primary Keys so that they can be easily retrieved from a database.
  - An Entity Bean may be a row of a database.
  - There are two types of persistence:
    1. Bean-managed Persistence: Persistence managed by the Bean.
    2. Container-managed Persistence: Persistence managed by Container.
- Session Bean: A Bean associated with client-server session.
  - There’re two types of Session Beans:
    1. Stateful Session Beans: Remember the state associated with the session across multiple sessions.
    2. Stateless Session Beans: The state is thrown away at the end of each session.

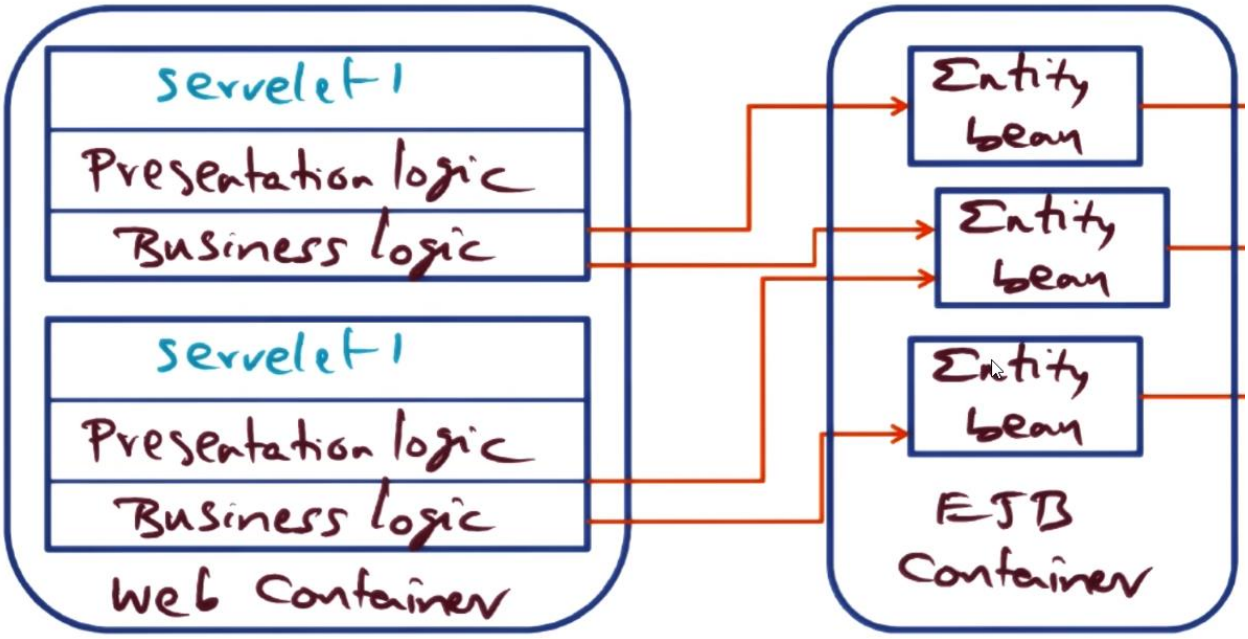
- Message-driven Beans: Useful for asynchronous behavior. An example would be receiving messages of interest that are typically event-driven (e.g. stock ticker information, newsfeed, RSS feed, etc.).
- Each Java Bean type denotes a particular functionality. Each Java Bean can be constructed into 2 forms, based on the granularity level:
  - Fine-grained Java Beans: Provide more concurrency in dealing with individual requests.
  - Coarse-grained Java Beans: Provide less concurrency, but keep the business logic simple.
  - Thus, the tradeoff in structuring N-tier applications is to choose either:
    1. Fine-grained granularity: More concurrency + complex business logic.
    2. Coarse-grained granularity: Less concurrency + simple business logic.

Java Beans Design Alternatives:

- The Client Container and the Applet Container are in the Web-server so we will not consider them. Instead, we'll only consider the Web Container (Presentation Logic) and EJB Container (Business Logic) in the several design alternatives below.
- A Servlet corresponds to an individual session with a particular client.
- Coarse-grained Session Bean:
  - A Coarse-grained Session Bean is associated with each Servlet and serves the needs of a Client.
  - Each Client is associated with one Session.
  - Pros:
    1. Minimal Container services needed from the EJB Container: The EJB Container coordinates concurrent independent sessions.
    2. Business Logic is not exposed beyond the corporate network since the Business Logic is contained in the EJB Container and not in the Web Container.
  - Cons:
    1. The Application Structure is similar to a Monolithic kernel.
    2. There is very limited concurrency for accessing different parts of a database in parallel. Hence, Coarse-grained Bean structure represents a lost opportunity in exploiting parallelism.



- Data Access Object:
  - The Business Logic is pushed to be in the Web Container containing Servlet and Presentation Logic. Similar to having a 3-tier software structure of (Servlet – Presentation Logic – Business Logic).
  - All Data Access happens through Entity Beans, which have Persistence characteristics. That is, Data Access Object (DAO) is implemented using Entity Beans.
  - Entity Beans can have Container-Managed Persistence or Bean-Managed Persistence.
  - An Entity Bean can represent the granularity of either one row of a database or a set of rows.
  - Multiple Entity Beans can work in parallel for a single client-server session.
  - The EJB Container contains these Entity Beans.
  - Pros:
    1. There is an opportunity for the Entity Bean to cluster the requests from different clients and reduce accesses to the database server across several different client requests that are temporally happening at the same time.
    2. The granularity of the Data Access Object (DAO) determines the level of concurrency desired in constructing the application service. This provides reusability opportunities.
  - Cons: Business Logic is exposed outside the corporate network because it was move from the EJB Container to the Web Container.



- Session Beans with Entity Beans:
  - The Web Container contains only the Servlet and the Presentation Logic associated with the Servlet.
  - The Business Logic sits along with the Session Façade and Entity Bean in the EJB Container.
  - A Session Façade is associated with each Client Session. The Session Façade handles all data access needs of its associated Business Logic.
  - The DAOs are implemented using multiple Entity Beans (having CMP/BMP) so that we get Concurrency and can reduce data accesses across different client requests.
  - The Session Bean communicates with the Entity Bean using Java RMI or local interfaces.
    1. Using local interface makes the communication faster since no network communication is used
    2. Using RMI makes the communication flexible enough to be used anywhere in network.
  - Pros:
    1. No network communication between Business Logic and Entity Beans.
    2. Business Logic is not exposed beyond the corporate network.
    3. There is an opportunity for the Entity Bean to cluster the requests from different clients and reduce accesses to the database server across several different client requests that are temporally happening at the same time.
    4. The granularity of the Data Access Object (DAO) determines the level of concurrency desired in constructing the application service. This provides reusability opportunities.
  - Cons: We're causing additional network access to do the service we want for the data access and that can be mitigated by co-locating the Entity Bean and the Session Façade in the same EJB Container.