1 Introduction to INT316D and JEE

In this chapter we introduce the student to our INT316D (Internet Programming) module and JEE (Java Enterprise Edition) which is also known as J2EE (Java 2 Enterprise Edition). In our discussion we cover the following:

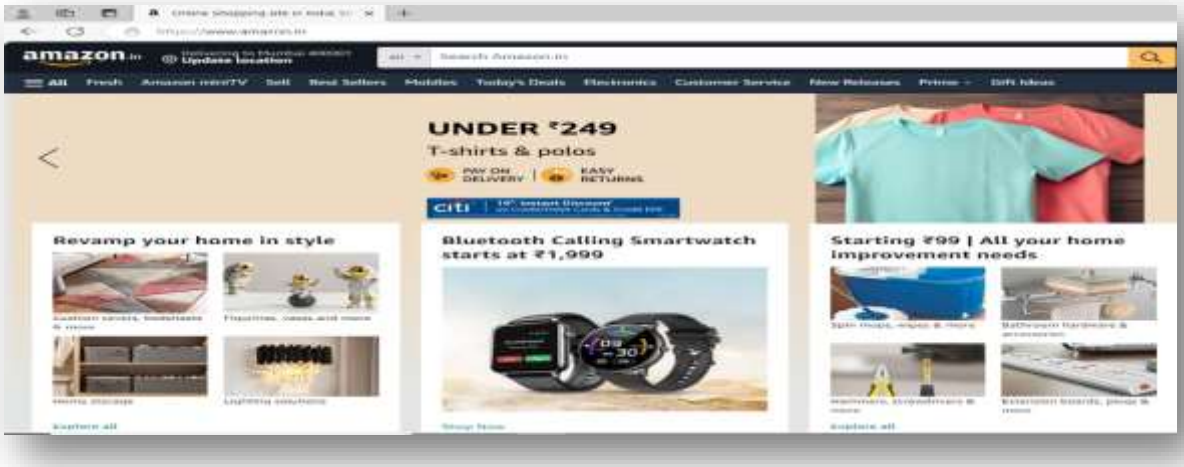
- Purpose of the module.
- How we accomplish the module's purpose.
- Relevant software for the module.
- The JEE framework.

1.1 What is the purpose of INT316D?

The purpose of this module is to capacitate students with the requisite skills to develop enterprise web applications that are **distributed**, **scalable**, **fault-tolerant**, **highly-available**, and **secured** using the **Java Enterprise Edition** (JEE) framework. These web applications can either be developed using an IDE such as **NetBeans** or a simple editor such as **Notepad++**. Ultimately the web applications are deployed in a **Web Server** such as **Tomcat** or an **Application Server** such as **GlassFish**. The user then accesses the web applications through a browser.

1.1.1 Enterprise

By enterprise applications we mean business applications. These are e-commerce systems that allow companies to trade over the internet. An example of such a web application is Amazon.



1.1.2 Distributed

By distributed we mean that the components of the web application are not located in the same JVM (Java Virtual Machine), they are distributed over different machines to accomplish a task. This has great benefits in that even during disaster, the system continues working (Never put all your eggs in one basket!!!).

1.1.3 Scalability

Scalable applications continue to function even when new components are added to the system. They don't require the whole system to be rewritten just because a new component is being added.

1.1.4 Fault-tolerant

By fault-tolerant we mean that the enterprise web applications can handle exceptions that may occur during program execution. This makes the enterprise web applications to be robust, meaning, they fail safe in the midst of unexpected error conditions.

1.1.5 High availability

Highly availability simply means the enterprise web applications are available 24/7. This high availability over the web is due to the deployment of the business applications in servers which are normally kept running all the time.

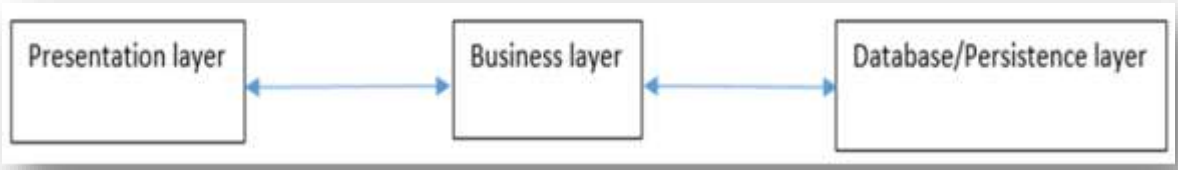
1.1.6 Secured

By secured we mean that the web applications we create implement security measures. In a secured system, users are authenticated and authorised to access resources based on the role they play in the system.

1.2 How do we accomplish the module's purpose?

We develop enterprise web applications that conform to the **3-tier** model of creating software. The 3-tier model simply says our web applications should consist of

components that can be categorised into three tiers/layers, namely the **presentation layer**; **business layer**, and the **database/persistence layer**. The figure below shows the 3-tier model.



1.2.1 Presentation layer

The presentation layer is mainly responsible for data presentation (entering and viewing of data). Through this layer users of the system are allowed to enter data, and also view processed data. In this module we are going to use the following presentation layer components:

- HTML
- JSP
- JSTL
- Servlets
- JavaScript

1.2.2 Business layer

The business layer is responsible for the processing of data. In this layer the actual business logic is performed. Say we want to determine the sum of two numbers. The summation of the numbers is the business logic, and it is done in this layer. We mainly use the following components to perform business logic:

- POJOs (Plain Old Java Objects, which are classes);
- EJBs (Enterprise Java Beans);

1.2.3 Persistence/Database layer

The persistence layer is responsible for database interaction. Through this layer, data is stored in the database and retrieved from it. We use entities (special EJBs) to interact with databases through an API called JPA (Java Persistence API). There are

various RDBMS (Relational Database Management System) available out there such as **Derby**, **MariaDB**, **Oracle**, **Postgress**, **MySQL** etc. Any of these can be used for database interaction with web applications.

1.3 Typical web applications to be developed

The web applications to be developed in this module will conform to the 3-tier model. We will use two PCs. The first PC will be a client that makes requests and then receives responses. The requests are made through a browser and the responses get rendered by the same browser.

The second PC will be a server that receives requests and send back responses to the client. The server will interact with the database. The protocol of communication between the client and the server will be **http**. The figure below shows the infrastructure of a typical web application we will develop.

Time permitting, we will change the infrastructure to the diagram below.

In this architecture we deploy the components of an application in distributed machines. Web components are kept in their own machine, different from EJBs.

1.4 Software to be used in the module

In this module we are going to use **NetBeans** to develop web applications, **GlassFish** to deploy web applications, and **Derby** for data management. NetBeans 8 comes along with GlassFish and Derby.

1.5 What is JEE?

JEE is a set or collection of APIs (class libraries) used to develop business applications. The APIs are also known as standards. Through the JEE framework, we are able to develop business applications that are **distributed**, **secured**, **robust**, **scalable**, and **highly available**. There are other frameworks in existence for developing business applications. Some of the frameworks are:

- Structs;
- Spring; and
- Hibernate.

The JEE APIs are implemented in containers or servers. Servers are divided into two, depending on which part of the enterprise APIs they support or implement. If a server only implements the web part of the APIs, we call it a **Web Server**. If it supports both the **Web** and **EJB** components, we call it an **Application Server**. So servers are classified in accordance to the JEE standards or APIs they implement or support. As a result they are sometimes called Reference Implementors (RI).

1.5.1 Web Server

A Web Server is a container that implements or supports the web components or APIs of JEE. Some of the web components of JEE are the following:

- Servlets;
- JSPs;
- Filters;
- Listeners; Lite; and
- Web Services

An example of a web server is **Tomcat**. You can get Tomcat from the following location:

Apache Tomcat® - Welcome!

1.5.2 Application Server

An Application Server is a container that supports both the web and EJB components or APIs of JEE. Some of the EJB APIs/components of JEE are the following:

- EJB;
- JPA;

An example of an Application Server is **GlassFish**. You can get GlassFish from the following location:

https://javaee.github.io/glassfish/

Other than GlassFish, we also have the following popular Application Servers:

WildFly/JBOSS

https://www.wildfly.org/

TomEE

https://tomee.apache.org/

1.6 History of JEE

Enterprise Java started in 1999 under the supervision of Sun. The company released J2EE 1.2 in 1999 with 10 specifications/standards/APIs in support of business applications development. Subsequent versions of Enterprise Java were released by the Java Community Process, an open organization started in 1998 for the eveolution of Java. JCP constitutes of company representatives, universities and private individuals with the aim of propelling Java forward.

After 1999, the versions of JEE evolved as follows:

September 2001:

J2EE 1.3 was released with 13 specifications/standards/APIs.

November 2003:

J2EE 1.4 was released with 20 specifications/standards/APIs.

May 2006:

Java EE 5 was released with 23 specifications/standards/APIs.

December 2009:

Java EE 6 was released with 28 specifications/standards/APIs.

May 2013:

Java EE 7 was released with 38 specifications/standards/APIs.

September 2017:

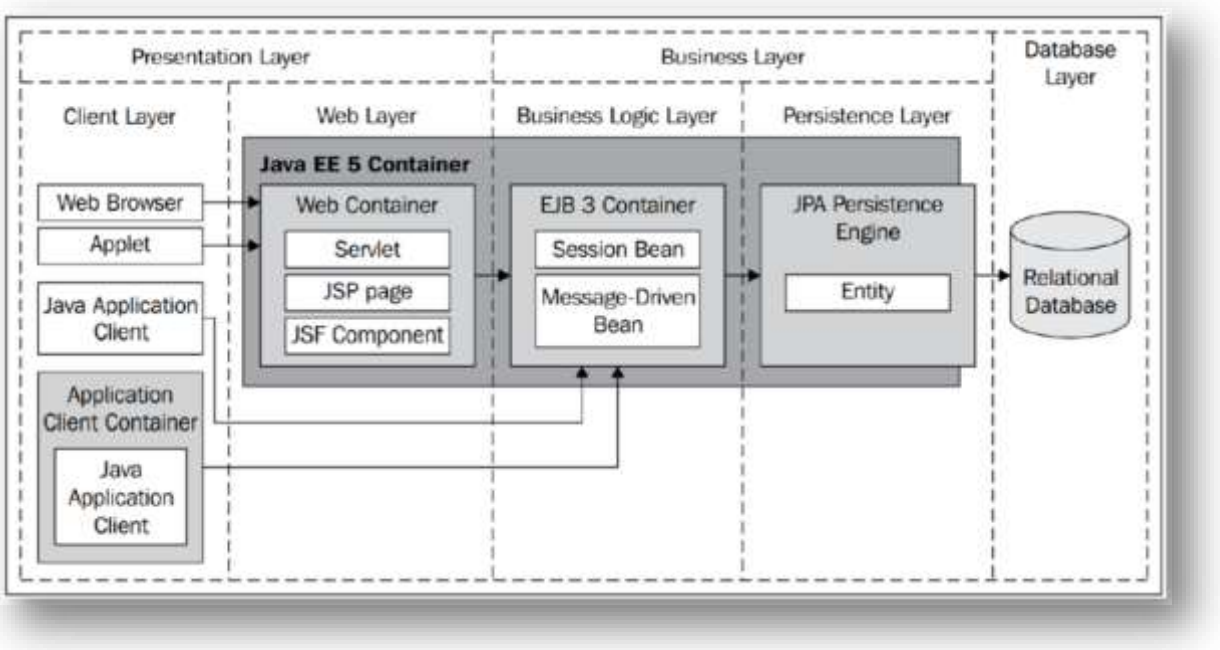
Java EE 8 was released with 40 specifications/standards/APIs.

For detailed information, check the link below from Oracle:

Java Platform, Enterprise Edition (Java EE) | Oracle Technology Network | Oracle

1.7 JEE architecture

The figure below shows the JEE architecture.



In the Java EE architecture diagram depicted above, the commonly known 3 layer model has become 5 layers. This is so because the presentation layer is subdivided into the client layer and the web layer. The business layer is also divided into the business logic layer and the persistence layer. In practice, the distinction between client/web and business logic/persistence layers is not always made. The JEE architecture is simply referred to as **n** layer or multi-layer architecture.

The Web Browser and Applet can talk directly to the web container. The web components can then talk to the EJB container, which can talk to the database. The Java Application Client can run as a standalone Java class with the main method or be deployed in the Application Client Container of the server as a Java Client project. The application client can talk directly to the EJB container, which in turn can talk to the database.

1.8 Benefits of using JEE compliant servers

The business web applications we create are ultimately deployed on the server. The server gives us the following benefits:

- The applications are available 24/7, so long the server is running.
- The server runs the deployed applications for us. The applications we develop don't have the **"main"** method. It is the server that runs them.
- The server manages the life cycle of components such as **Servlets** and the **EJBs**. This means the instantiation and destruction of these components is done by the server, not us.
- The server supports multi-threading, we don't need to create threads for every request we get.
- The server supports both declarative and programmatic security.
- The server supports seamless communication with ports. We don't need to write code that listens to ports.