Java EE - Big Picture

Saturday, 4 August 2018 12:54 AM

Source: https://app.pluralsight.com/library/courses/java-ee-big-picture

Java has four Editions:

1. Java SE (Standard Edition)
2. Java EE (Enterprise Edition)
3. Java ME (Micro Edition)
4. Java Fx (Effects)

IMPORTANT ->All java editions consists of JVM and a set of APIs.
JVM is a program for a particular OS and Hardware which runs java applications.
And the set of APIs are software components that can be used to create other software components or applications.

Java SE

- Core java programming platform
- Contains all core APIs and Libraries that any Java developer should learn.
- Java SE APIs provide core functionality of Java Programming language.
- They define everything from basic types to high level classes for networking, security, GUI, XML parsing etc.
- It includes the following as part of **JDK**:
  - JVM
  - Development tools
  - Deployment and monitoring systems.
  - Class libraries and toolkits commonly used in java aplications.

Java ME

- It is a subset of Java SE for mobile Devices.
- Provides small footprint JVM to run on sensors, IOT devices, set-top boxes etc.
- Most suitable for IOT and embedded applications.

Java FX

- Java FX is a platform for creating rich internet applications using a light weight **user-Interface** API.
- It uses Graphics Cards(Hardware accelerated graphics) .
- Has modern look and feel.
- Has high level APIs to connect to remote services.

Java EE

- Java EE extends Java SE. So SE is subset of EE.
- It provides environment for Developing and running Enterprise applications.
- This includes:
  - Network services
  - Web Services
  - All kinds of large-scale, multi-tier, reliable and secure applications.
- If any application demands scalability and distribution than Java EE is much better than SE.
- Java SE, Java ME, and Java FX are usually clients of Java EE applications.

What is an Enterprise Application?

An application which is:

1. Scalable
2. Reliable
3. Secure
4. Multi-tiered.

Multi-tiered

In **Multi-tiered Application**, the functionality of application is separated into isolated functional areas called tiers. We can have one to many tiers depending upon complexity of our application.
Usually an enterprise application has:

1. Client Tier
2. Middle Tier
3. EIS Tier (Enterprise Information Tier)

Client Tier

It can be browsers, but also java applications running on Java SE, ME

It can be browsers, but also Java applications running on Java SE, ME or FX.

Middle Tier

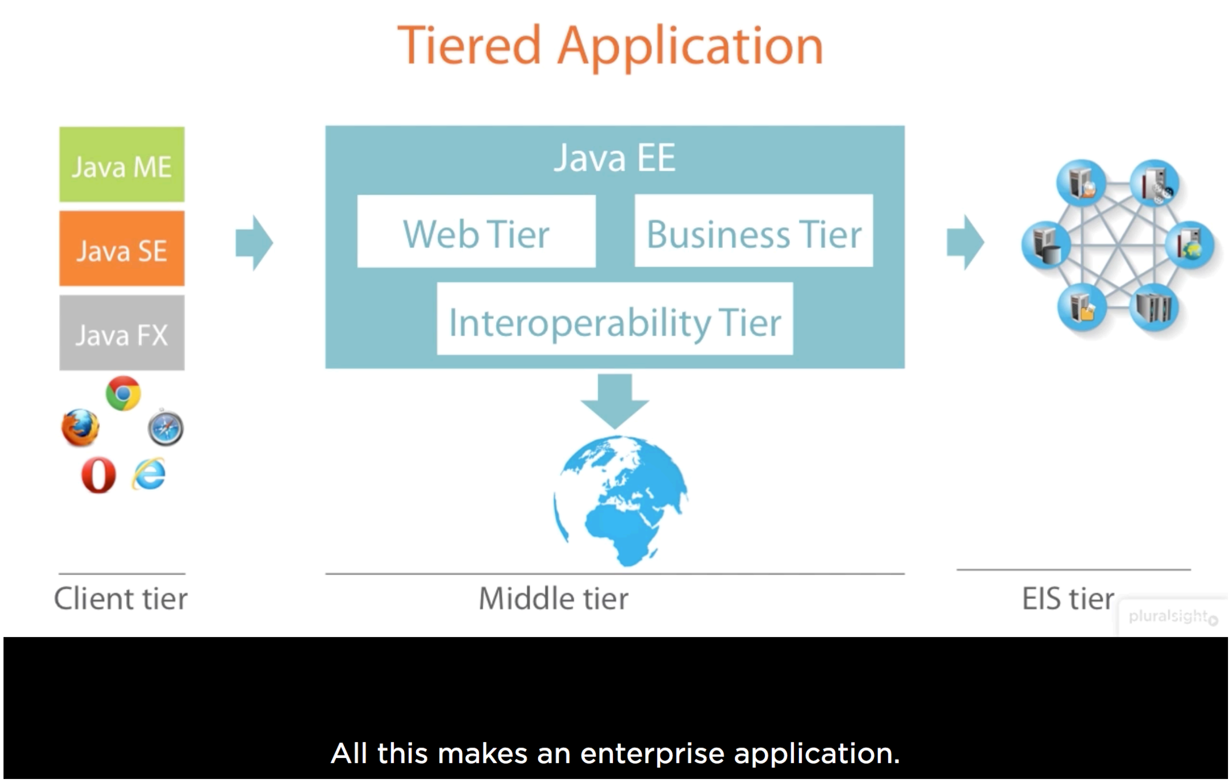
Here we find Java EE platforms with its own tiers:

i. **Web Tier:** Components that control interaction between web clients and business tier. It creates content dynamically in various formats for client. Controls flow of screen of pages, maintains user session etc.
ii. **Business Tier:** It handles the request and processes application data, string to permanent storage device in **Data Tier**. Here we have all the business logic.
iii. **Interoperability Tie**r: allows interaction with external services using messaging(like JMS) or web-service.

EIS Tier

Consists of:

- Database service
- Enterprise resource planning
- Main frames etc.



All this makes an enterprise application.

Java SE vs Java EE

Like when we need a list of object we don't go and implement collections but use APIs.

Similarly when we need enterprise applications we do not apply them but

Similarly when we need enterprise applications we do not apply them but use APIs for messaging, Transactions etc.
Java EE code is run inside container that brings extra services such as transactions, messaging, persistence, etc.
Java EE gives extra services on top of Java SE.
It reduces complexity by providing APIs, programming model , runtime environment etc. allowing developers to concentrate on business requirements.

Java EE programming model

- It emphasizes **convention over configuration**.
  - **Convention over configuration**(also known as coding by **convention**) is a software design paradigm used by software frameworks that attempts to decrease the number of decisions that a developer using the framework is required to make without necessarily losing flexibility.
  - The container takes default decisions and about our code. If default don't suit us then we can change them by using XML of Annotations. This uses **Metadata** to deviate from convention. This Metadata is understood by container.

What is Java EE?

Java EE Architecture

- Java EE is centered about a runtime environment usually called a **container**.
- Container provides services to the services it hosts, such as
  - Life cycle management
  - Dependency injection
  - Distributed Applications and scalibilty.
  - Concurrency etc.
- These components use well defines contracts or APIs to communicate with java infrastructure and with other components.
- Components needs to be packaged in a standard way before being deployed to the container.
- Once deployed there are certain protocols that can be used to access these components.
  - Container
    - Container is a run time environment. Example: TomEE,

IBM Web Sphere, Jboss etc

        - Its primary role is to hide technical complexity.
        - enhance and support portability to the application it hosts.
        - Developers can focus on writing business code and leave resource management, multithreading, resource pooling etc. to container.
        - A container manages components and these components can be static and dynamic web pages that process requests and constructs responses.
        - Components can also be server side java classes that handle business code and process data.
        - The container provides services to components such as:
            - Security
            - Transaction Management
            - Naming
            - Remote Connectivity.
        - Since Java EE provides configurable services, components on same application may behave differently based on where they are deployed.(Example: using Spring Profiles.)
        - Configuration is isolated for each component.
    - Metadata
        - Metadata can take form of Annotations that is embedded in code or can be an XML which is external to it.
        - This extra information is given by component to the container,.
        - The container will configure appropriate service to component at deployment time and runtime.
        - Thus component need not to create services. It uses them.
        - This is the concept of **Inversion Of Control.** The container takes control of our business code and provides technical service
    - Packaging
        - Thus an application is an aggregation of components which are web pages, web resources, business components, database access components, XML or annotation configuration etc.
        - For application to be executed we need these components to be assembled together in an archive and deployed to container.
        - The packaging has a standard format so that it can be executed in any format increasing portability.

        - We can deploy multiple applications to a container and container can isolate each other from one other.
        - To allows these applications to communicate with each other it defines a list of protocols.
    - Protocols
        - HTTP
        - HTTPS
        - RMI
        - RMI over IIOP
        - The clients of these can be web pages, Web Apps, Rest Soap Services, Mobile apps, Browsers etc. i.e. anything on web tier or even other Java EE apps.

Java EE Limitations

- Cannot leverage HTML, CSS, JS.
- Not suited for IOT devices but can communicate with them.
- Not suited for Non-relational databases and big data.
- Not suitable for Data Science: Machine learning, Data Mining, Visualization, Pattern recognition etc.
- Best choice to make micro-services.