JEE7 Features

Java EE 7: Key Drivers and USP

Developer Productivity

- ***Annotated POJOs***, which have proven to be the cornerstone of ease-of-development paradigm since Java EE 5, continue to rule the roost and further penetrate Java EE
- ***Contexts and Dependency Injection (CDI)*** reputation of being the 'magic glue' has been taken to the next level – it's now 'default' in Java EE 7 and its goal is to drive cohesiveness throughout the entire EE platform
- ***JMS 2.0*** (JSR 343) has been completely '*revamped*'. A 'new version' of the API known as the '***Simplified API***' is more succinct, easy-to-use and will ensure that developers have to deal with 'significantly *lesser*' amounts of '*boilerplate*' code

HTML5 Focus

- ***JAX-RS 2.0*** (JSR 339) – The RESFTful API in Java EE 7 has been further enriched, including addition of a brand new *'client' side API, asynchronous capabilities,* Servlet *Filters and Interceptors*
- ***Java API for WebSocket*** (JSR 356) provides a 'easy-to-use' and powerful 'high level' abstraction for developers to write low latency, real-time and feature rich *WebSocket driven applications*
- ***JSON Support*** – The Java API for JSON Processing (JSR 353) has finally 'standardized' JSON processing. No more 'third party' libraries required for dealing with JSON.

Fulfilling Enterprise Demands

- ***Concurrency Utilities*** (JSR 236) API now makes it possible to fire your own threads in Java EE 7 (forbidden prior to this) in a controlled/managed fashion
- ***Batch Applications for Java Platform*** (JSR 352) – Brand new specification in Java EE 7, providing standard way to write efficient '*bat ch processing*' applications

The '***brand new***' specifications which were introduced in Java EE 7 are

- Java WebSocket API 1.0 (JSR 356)
- Java API for JSON Processing 1.0 (JSR 353)
- Concurrency Utilities 1.0 (JSR 236)
- Batch Applications for Java Platform 1.0 (JSR 352)

Java WebSocket API 1.0 (JSR 356)

- Built on top of the *WebSocket* protocol which is an **IETF** standard (***RFC 6455***)
- WebSockets allow ***full-duplex***, ***bi-directional*** communication b/w client and server over a single TCP connection
- *Client can send a message to server* at any point after the connection is established
- *Server can send messages to its connected clients/peers* without any 'explicit request' from them i.e. totally independent of the client
- The WebSocket API in Java EE 7 supports ease-of-development by allowing support for *intuitive annotations* which*transform simple POJOs* into Server or Client web socket end points. The framework also supports 'Programmatic' endpoints which works on the basis of
- Capability to *intercept WebSocket lifecycle events* with the help of annotations
- Support for text and binary messages along with health check (ping-pong) messages

The Java API for JSON Processing (JSR 353)

- Provides a '*standard*' way to *parse and generate* data in JSON format
- Developers are *not* 'required' to use *third party* JSON libraries (Jackson etc), however, they can choose to 'plugin' an implementation of their choice if need be
- Modeled on similar lines to the JAXP API and has support for Streaming API and DOM API
- As the name suggests, the ***Streaming API*** helps process/generate JSON in a streaming fashion (similar to the StAX API from JAXP). It is a low level API which works on the '*events*' obtained from the JSON data stream
- The ***DOM API***, creates an *in-memory Java object model* for the JSON data (similar to the XML DOM API). It's a easy to use high level API
- Support for a '*binding*' API (***JSON-B*** similar to JAXB) is in progress and should be 'standardized' in Java EE 8

Batch Applications For the Java Platform (JSR 352)

- Suitable for execution of 'long running' ***bulk jobs*** which do not require human intervention and can be scheduled as per requirement e. g. ETL jobs, end of day jobs etc
- Provides a complete *programming model* for 'batch' oriented applications
- Defines a ***Job Specification Language*** which is the basis for defining 'jobs' within an XML. This *Job XML* captures the entire batch process.
- Supports two contrasting 'processing' mechanisms.
- ***Chunk Style Processing*** involves, 'reading' the input 'bulk' data, 'processing' it and finally 'writing' it as the final step in the process. All these operations are abstracted through specific interfaces, namely, ItemReader, ItemProcessor and ItemWriter
- A '***batchlet***' style processing model is also supported by this API which defines' tasks' which once completed, mark the end of the job

Concurrency Utilities for Java EE (JSR 236)

- Earlier editions of Java EE forbid initiating application specific threads using the Java SE concurrency API.
- The new specification provides a standard way to support custom 'concurrency' constructs from within applications.
- Leverages/extends the *Java SE concurrency utilities* and provides '***Managed***' versions of these APIs
- Now, developers can *create threads* in a '*managed*' and 'controlled' manner using the ***javax.enterprise.concurrent.ManagedThreadFactory***
- ***javax.enterprise.concurrent.ManagedExecutorService*** can be leveraged to fire tasks in an '*asynchronous*' fashion
- This API also allows developers to *schedule* periodic tasks via the ***javax.enterprise.concurrent.ManagedScheduledExecutorService***

That was it as far as the 'fresh' specifications are concerned. Here is the *revamped* specification list

- Java Message Service API 2.0 (major revamp for the first time in 10 years)

- JAX-RS 2.0 (RESTFul API)
- Enterprise JavaBeans 3.2
- Java Persistence API 2.1
- Contexts And Dependency Injection 1.1
- JavaServer Faces 2.2
- Java Servlet 3,1
- Interceptors 1.2
- Bean Validation 1.1