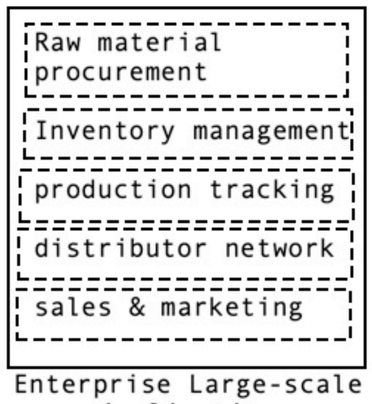


Enterprise Large-scale Application

[built on Monolithic Application Architecture , packaged & deployed as a single deployable artifact]

1. security
2. resource pooling (connection pooling)
3. global transactions
4. application logging
5. clustering/scalability
6. caching
7. load balancing
8. messaging services
9. monitoring
10. memory management
11. schedulers
12. auditing

Most of these features are quite commonly required while building enterprise class-level applications.
For eg.. Connection Pooling or Security is not that we only need aspart of our application, it is a common requirement for every application like
  1. pool the db connections so that we can grab and release the connections quickly thus reducing the performance or memory in terms of acquiring and releasing them
  2. similarly securing an application by authenticating the user and granting them access to the features of the application based on the roles of the users is not an specific requirement for our application rather it is an common requirement

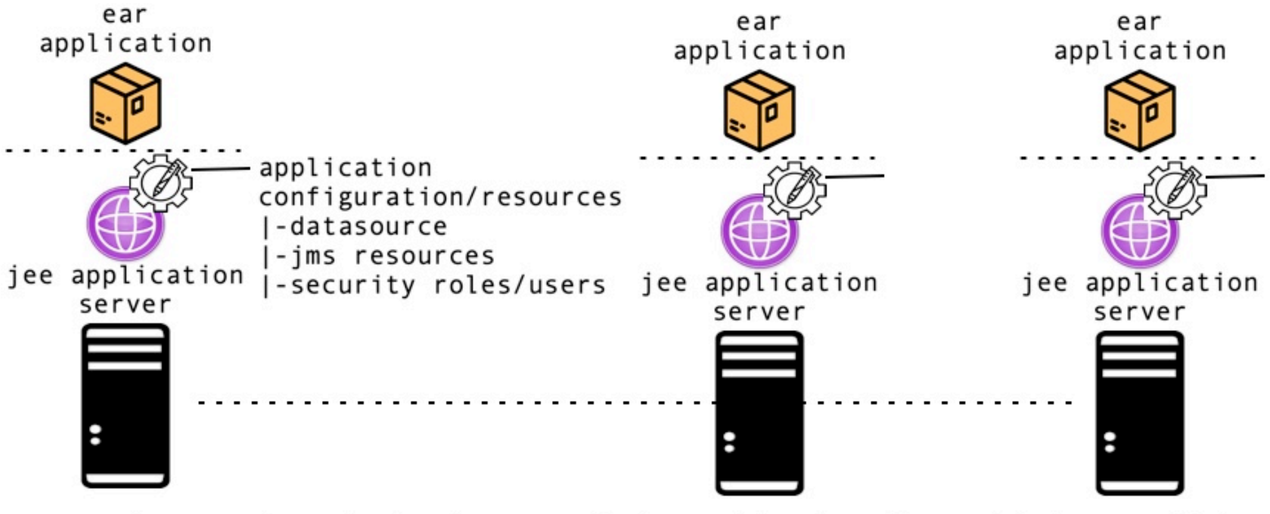
so if the developer has to build these common features by his own, the amount of time and complexity it takes in building these features would be very high and results in huge cost of development as well.
So to help us in quickly building the enterprise stack application the jee platform has been introduced by Sun Microsystems or Oracle.

All these cross-platform services which are common across all the enterprise applications are provided by the jee platform itself interms of
1. api
2. server infrastructure (jee containers)

1. security = jee container / javax.security api
2. connection pooling = jee container takes care of pooling the connections and we need to use DataSource api for using the connections from the pool
3. global transactions = javax.transaction api and every jee container supports global transactions
4. application logging = every java application server supports app logging
5. clustering/scaling = jee containers supports clustered deployments
6. caching = most of the jee containers providers server-side caching (for eg.. weblogic server supports: coherence cache)
7. loadbalancing = jee servers provides built-in http loadbalancer component
8. messaging = jms support for creating messaging datastructures like queues/topics are supported
9. monitoring = jee container takes care of monitoring and managing the applications across the cluster
10. memory management = jee servers supports memory management by default
11. schedulers = jee has scheduler api through which we can build and run batch applications
12. auditing = jee server supports auditing (access logs/audits)

If we deploy the enterprise large scale application by packaging into an single deployable artifact onto a Single Server/Machine, it will not be able to scale and handle the traffic load, we need to deploy it on a cluster of machines.

To deploy the application on cluster of machines we need to setup the application server on each machine, and need to add relevant resources/configurations on each of these servers like
1. datasource configurations
2. jms resources
3. security roles/users configurations
4. logging configurations
etc
and then we need to manually deploy the application on all the servers across the cluster.
Performing all of these above activites on all the nodes of the cluster manually seems to be a very tedious job and requires labours efforts in running the system, to overcome all of these challenges in running an enterprise application on a cluster of machines, the jee application servers are introduced.



Jee platform not only provided cross-cutting services in development of the application along with it provided necessary support interms of deployment/ rollout of the application onto the production-grade environments through the help of jee containers.