Mail\_**EJB**\_JS\_CSS\_HTML\_JQ\_AJAX\_Security(WS-RS)\_TransactionManagement\_FewAnnotations

Logging\_Junit\_PDFGeneration🡺in detailed

**EJB**

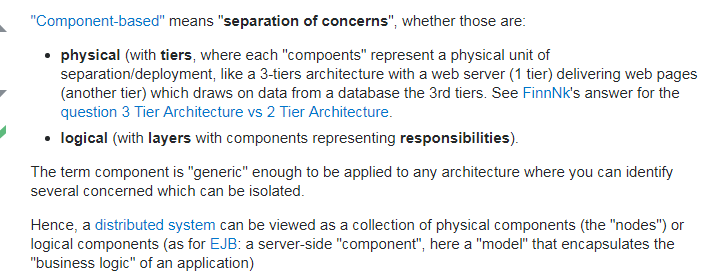
1. **E**nterprise **J**ava **B**eans (EJB) is a development architecture for building highly scalable and robust enterprise level applications to be deployed on J2EE compliant Application Server such as JBOSS, Web Logic etc.

Note 🡺 EJB IS A SERVER SIDE COMPONENT SO ITS GETS DEPLOYED ON THE SERVER THEN WHAT ABOUT THE JS/JQ WHERE THIS CODE GETS DEPLOYED NEEDED MORE INFORMATION ON THIS

THAT IS

EXAT DIFFERENCE BETWEEN SERVER SIDE COMPONENT AND CLIENT SIDE COMPONENT

[Component-based architecture vs. Distributed system](https://stackoverflow.com/questions/1621560/component-based-architecture-vs-distributed-system)

****

**MS 🡺 so can we say that MVC(I think mvc is architecture not the design pattern) itself is a component based architecture?**

# [What is the difference between framework and architecture?](https://stackoverflow.com/questions/2190625/what-is-the-difference-between-framework-and-architecture) 🡺

# Stack overflow

# I would like to know the difference between framework and architecture. for example: dotnetnuke is the framework and mvc is the archite

# 

# Or

# Simply put -- architecture is theory, framework is implementation.

# 

# 

# What is a design pattern

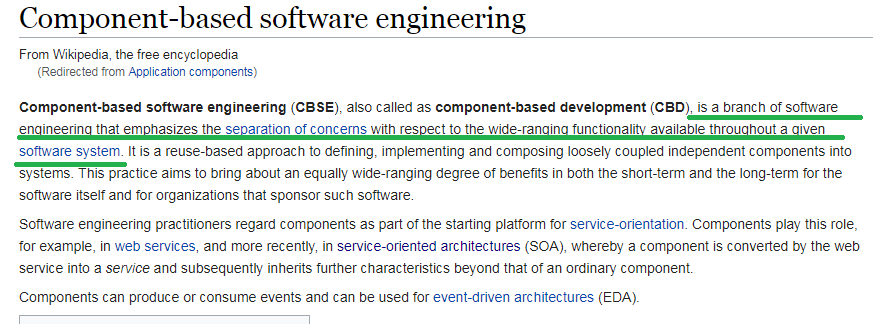
# 

# Difference between design pattern and architecture?

# 

1. Unlike RMI, middleware services such as security, transaction management etc. are provided by **EJB Container** to all EJB applications
2. middleware services 🡺

from wikipedia

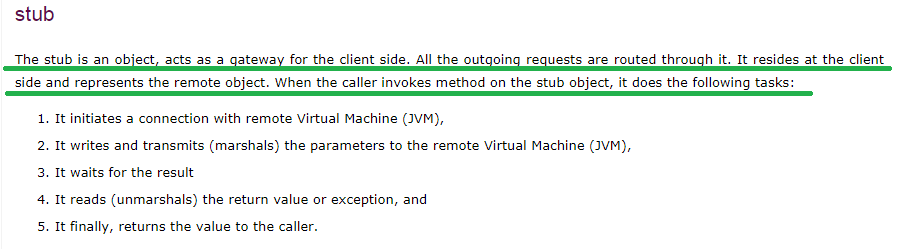
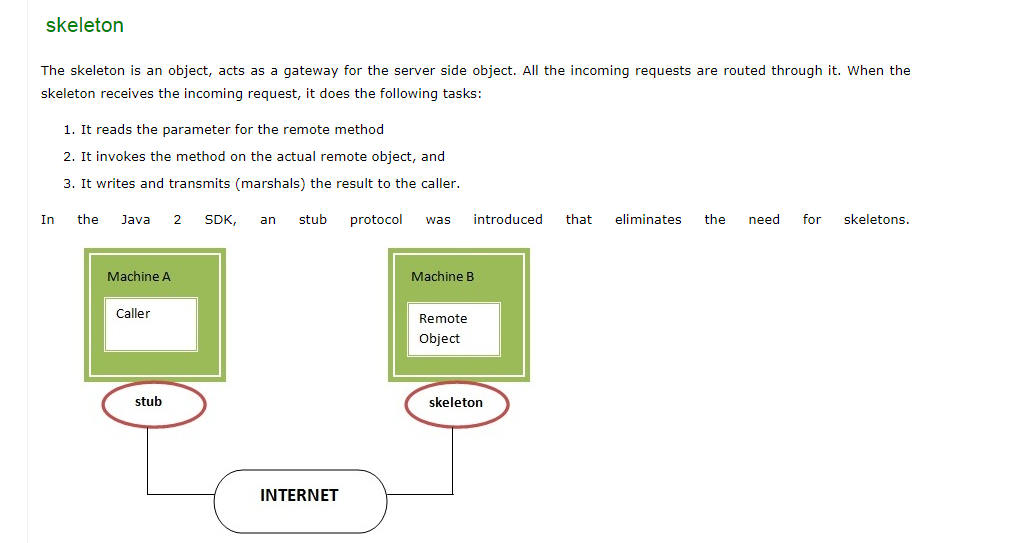
1. **Middleware** in the context of [distributed applications](https://en.wikipedia.org/wiki/Distributed_application) is [software](https://en.wikipedia.org/wiki/Software) that provides services beyond those provided by the [operating system](https://en.wikipedia.org/wiki/Operating_system) to enable the various components of a distributed system to communicate and manage data. Middleware supports and simplifies complex [distributed applications](https://en.wikipedia.org/wiki/Distributed_application). It includes [web servers](https://en.wikipedia.org/wiki/Web_server), [application servers](https://en.wikipedia.org/wiki/Application_server), messaging and similar tools that support application development and delivery. Middleware is especially integral to modern information technology based on [XML](https://en.wikipedia.org/wiki/XML), [SOAP](https://en.wikipedia.org/wiki/SOAP), [Web services](https://en.wikipedia.org/wiki/Web_service), and [service-oriented architecture](https://en.wikipedia.org/wiki/Service-oriented_architecture).
2. **A service-oriented architecture (SOA) is a style of software design where services are provided to the other components by**[**application components**](https://en.wikipedia.org/wiki/Application_components)**, through a**[**communication protocol**](https://en.wikipedia.org/wiki/Communications_protocol)**over a network. The basic principles of service-oriented architecture are independent of vendors, products and technologies.**[**[1]**](https://en.wikipedia.org/wiki/Service-oriented_architecture#cite_note-1)**A service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently, such as retrieving a credit card statement online**.
3. 
4. The current version of EJB is EJB 3.2. The development of EJB 3 is faster than EJB 2 because of simplicity and annotations such as @EJB, @Stateless, @Stateful, @ModelDriven, @PreDestroy, @PostConstruct etc.
5. EJB is a specification provided by Sun Microsystems to develop secured, robust and scalable distributed applications.
6. distributed applications🡺

 get information about distributed applications, visit [RMI Tutorial](https://www.javatpoint.com/RMI) first.

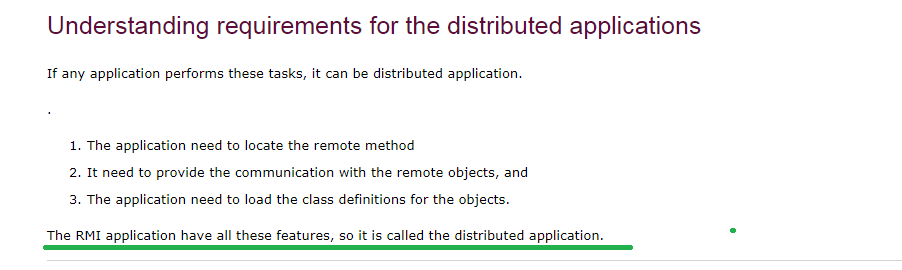
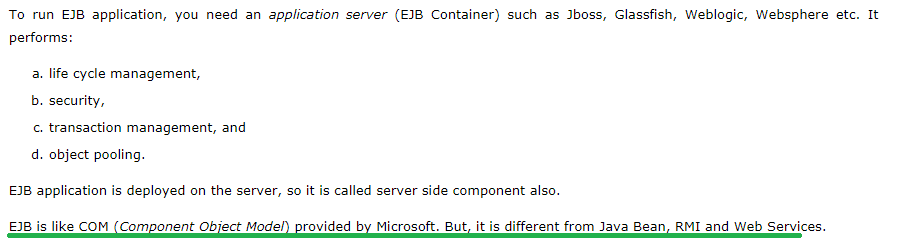
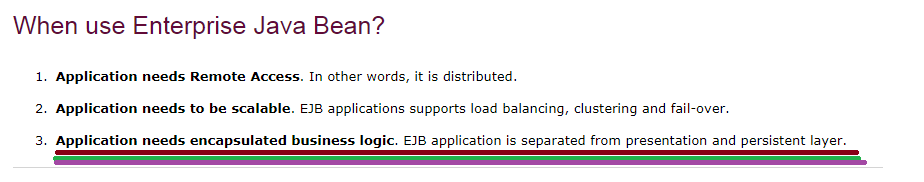
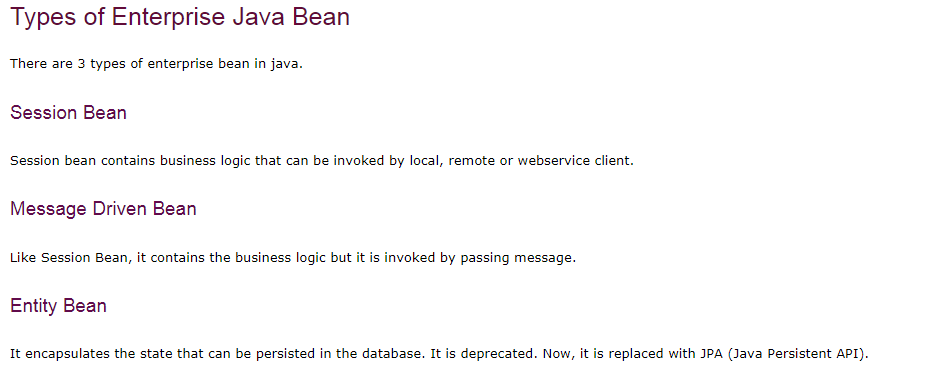
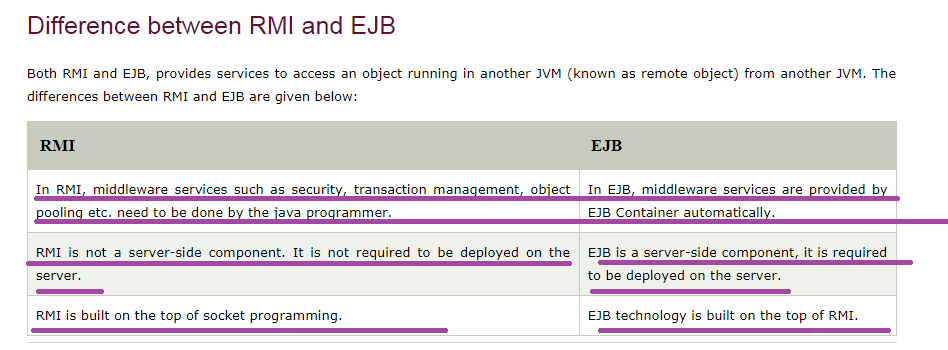
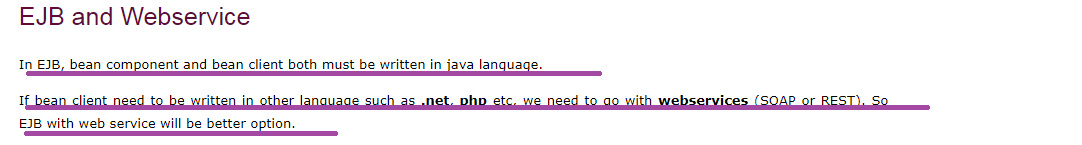
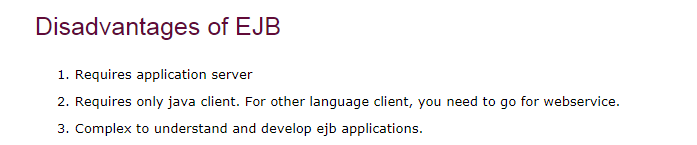
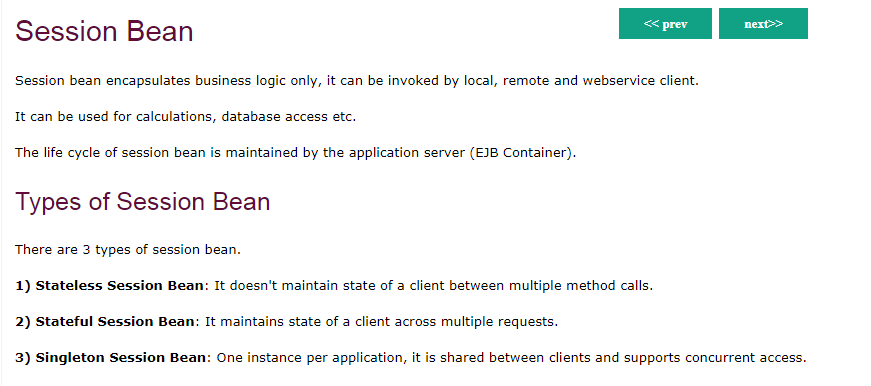
[RMI Tutorial](https://www.javatpoint.com/RMI)

1. The **RMI** (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java.
2. The RMI allows an object to invoke methods on an object running in another JVM. 🡺 MS THIS IS WHAT MEANS DISTRIBUTED APPLICATION

The RMI provides remote communication between the applications using two objects *stub* and *skeleton*.

1. 
2. 

**If any application performs these tasks, it can be distributed application.**

1. 
2. 
3. 
4. 
5. 
6. 
7. 
8. 

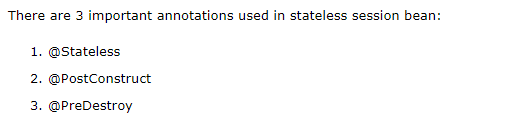
# Stateless Session Bean

1. **Stateless Session bean** *is a business object that represents business logic* only. It doesn't have state (data).

In other words, *conversational state* between multiple method calls is not maintained by the container in case of stateless session bean. 🡺 NEEDED MORE INFORMATION ON “MULTIPLE METHOD CALLS”

The stateless bean objects are pooled by the EJB container to service the request on demand. 🡺 MS I THINK ITS SINGLETON SCOPE

It can be accessed by one client at a time. In case of concurrent access, EJB container routes each request to different instance.

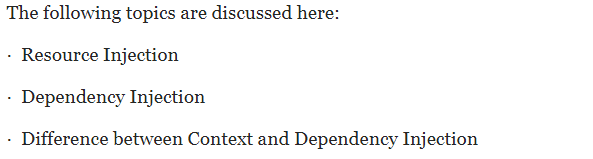


In our real world ours is ejb3 only and the annotations that we had used is

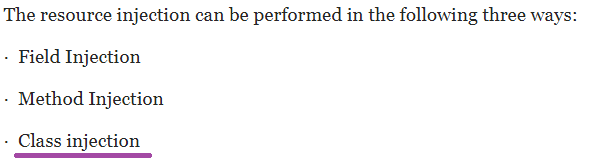
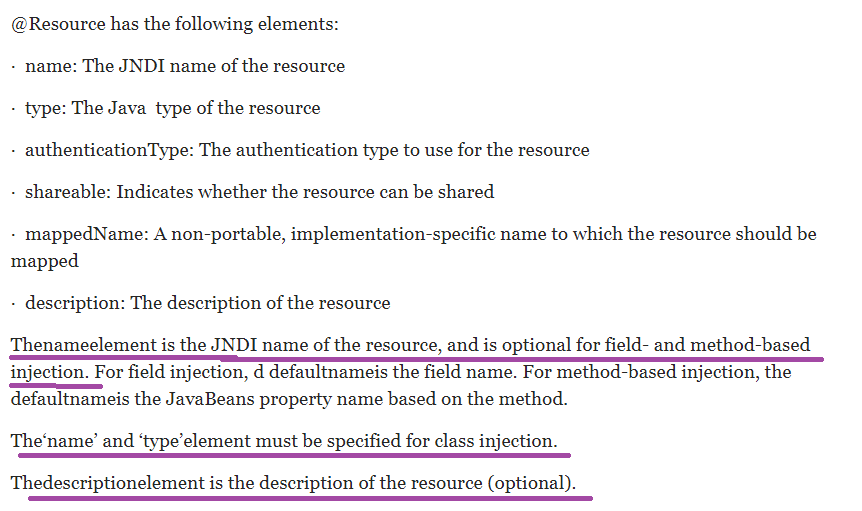
* 1. @Stateless 🡺ejb @
  2. @LocalBean🡺ejb @ 🡺
  3. @Resource 🡺javax.annotation.Resource
  4. @Inject🡺javax.inject.Inject

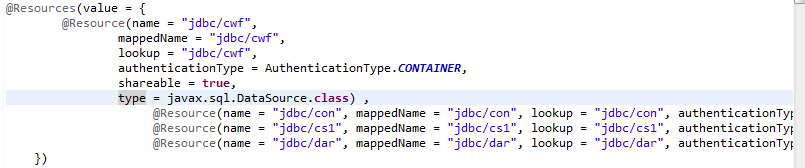
Now let’s see more about the @Resource and @Inject 🡺 the website I had referred is 🡺 <https://dzone.com/articles/resource-injection-vs>

# Resource Injection vs. Dependency Injection Explained

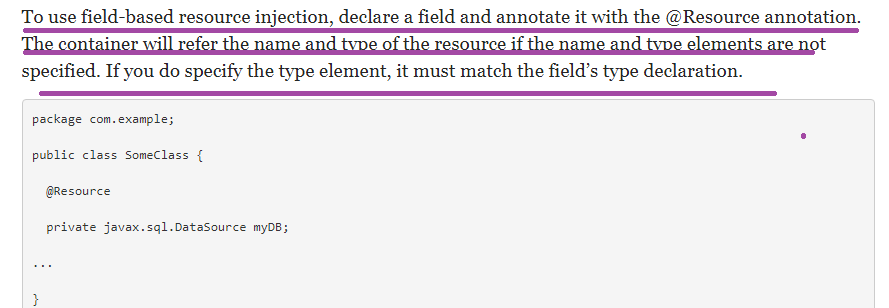
1. Java EE provides injection mechanisms that enable our objects to obtain the references to resources and other dependencies without having to instantiate them directly (explicitly with ‘new’ keyword).
2. We simply declare the needed resources & other dependencies in our classes by drawing fields or methods with annotations that denotes the injection point to the compiler.
3. **The container then provides the required instances at runtime. The advantage of Injection is that it simplifies our code and decouples it from the implementations of its dependencies.**
4. 

### **Resource Injection**

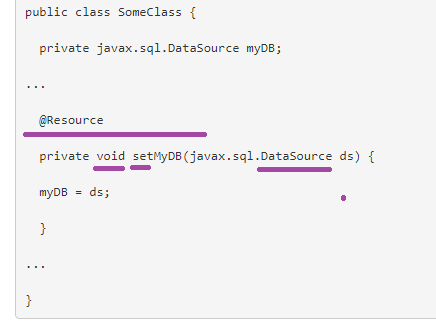
1. **One of the simplification features of Java EE is the implementation of basic Resource Injection to simplify web and EJB components.**
2. **Resource injection enables you to inject any resource available in the JNDI namespace into any container-managed object, such as a servlet, an enterprise bean, or a managed bean. For eg, we can use resource injection to inject data sources, connectors, or any other desired resources available in the JNDI namespace.**
3.  🡺 IN OUR REAL WORLD WE HAD USED CLASS INJECTION
4. Now, the **javax.annotation.Resource** annotation is used to declare a reference to a resource. So before proceeding, let’s learn few elements of **@Resource** annotation.
5. 



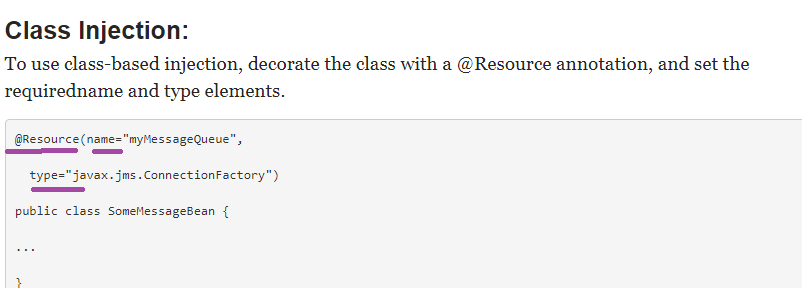
### Field Injection:

1. 

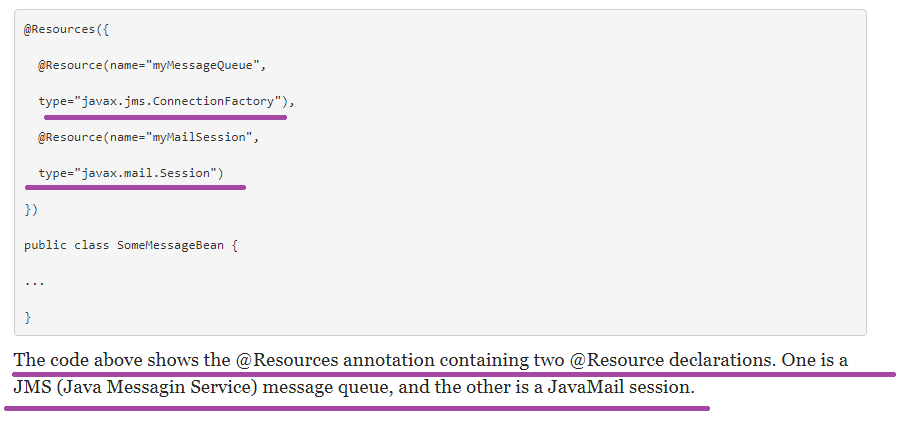
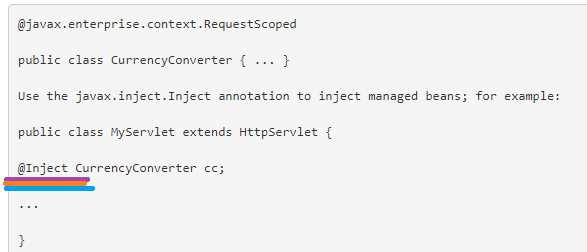
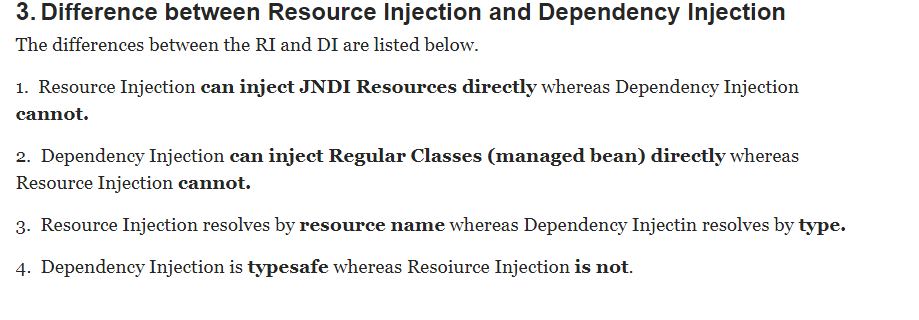
### Method Injection:

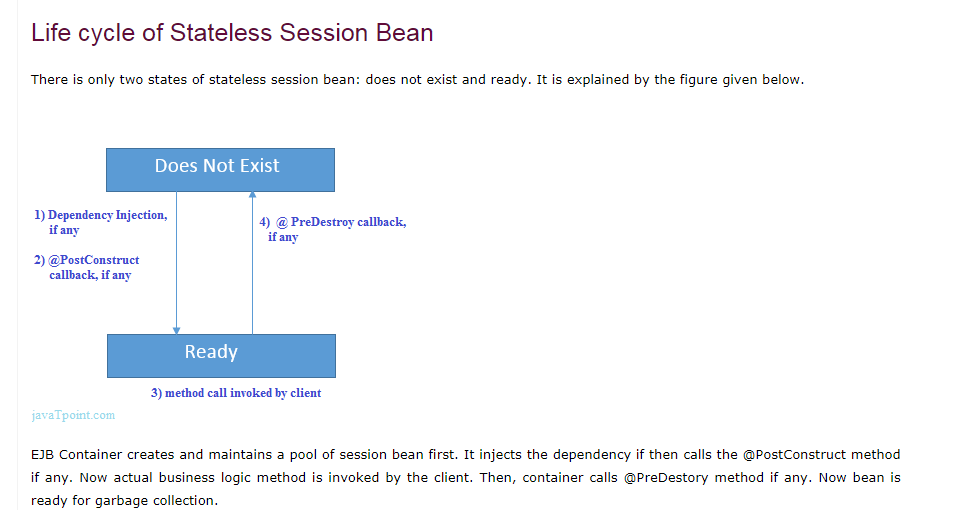
1. To use method injection, declare a setter method and preceding with the @Resource annotation. The container will itself refer the name and type of the resource if in case it is not specified by programmer. The setter method must follow the JavaBeans conventions for property names: the method name must begin with set, have a void return type, and only one parameter (needless to say :P).
2. 

### Class Injection:

1. 

### Declaring Multiple Resources

1. The **@Resources** annotation is used to group together multiple **@Resource** declarations for **class injection only.**
2. 
3. **Dependency Injection**
4. **Dependency injection**enables us to turn regular Java classes into managed objects and to inject them into any other managed object (objects wich are managed by the container)
5. 
6. 

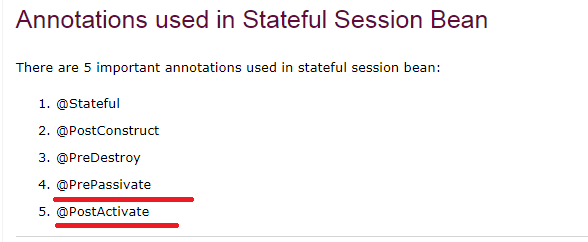


AN EXAMPLE OF STATELESS BEAN IS NOT GOOD IN TPOINT

# Stateful Session Bean

**Stateful Session bean** *is a business object that represents business logic* like stateless session bean. But, it maintains state (data).

In other words, *conversational state* between multiple method calls is maintained by the container in stateful session bean.



NOTE🡺 NEEDED MORE INFORMATION ON @PrePassivate AND @PostActivate

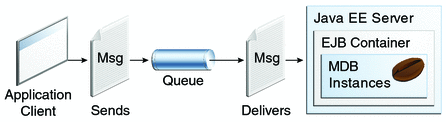
HTML PARSING, SMIP ON CONVERSION

Needed a very clear example

# JMS Tutorial

# Message Driven Bean

* 1. A message driven bean (MDB) is a bean that contains business logic. But, it is invoked by passing the message. So, it is like JMS Receiver.
  2. A message driven bean receives message from queue or topic
  3. A message driven bean is like stateless session bean that encapsulates the business logic and doesn't maintain state.

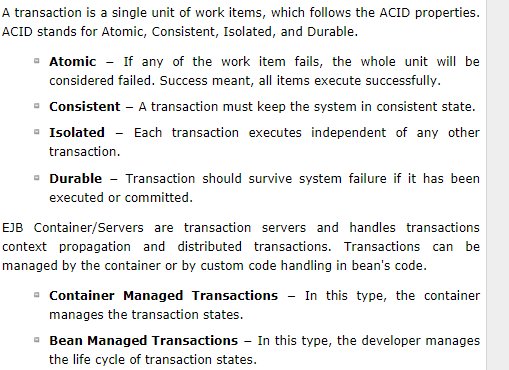
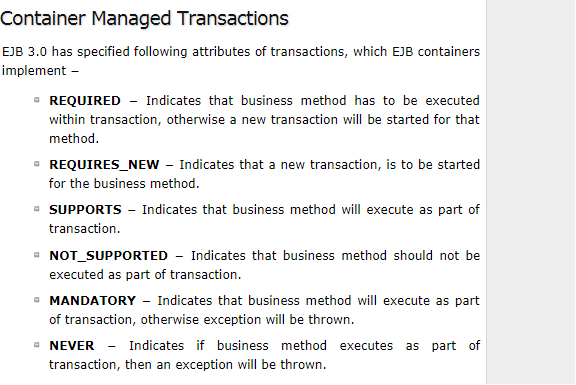
1. 

# Entity Bean in EJB 3.x

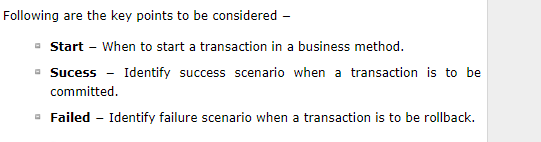
1. Entity bean represents the persistent data stored in the database. It is a server-side component.
2. In EJB 2.x, there was two types of entity beans: **bean managed persistence** (BMP) and container managed persistence (CMP).
3. Since EJB 3.x, it is deprecated and replaced by JPA (Java Persistence API) that is covered in the hibernate tutorial.
4. In hibernate tutorial, there are given hibernate with annotation examples where we are using JPA annotations. The JPA with Hibernate is widely used today.
5. EJB life cycle apart from Stateless
6. Now let’s see one more EJB Annotation 🡺@TransactionAttribute
   1. @TransactionAttribute 🡺 javax.ejb. TransactionAttribute
   2. Syntax example🡺

@TransactionAttribute(TransactionAttribute.REQUIRES\_NEW/NOT\_SUPPORTED)

PUBLIC RETURNtYPE METHODnAME(){}

* 1. Uses🡺@TransactionAttribute determines the way to handle transactions. But when you choose REQUIRES\_NEW it will always create a new transaction for each method invocation
  2. Explanation from Tutorials point🡺
* 
*  🡺 ms in our real world we had uses container managed transaction
* 

## **Bean Managed Transactions**

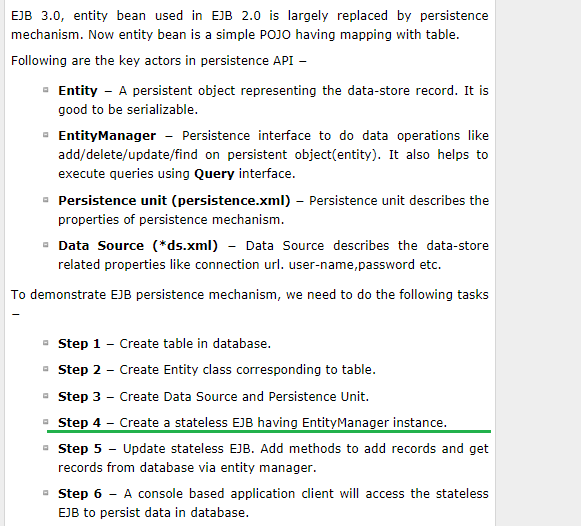
* In Bean Managed Transactions, Transactions can be managed by handling exceptions at application level. 🡺 done by developers
* 
  1. **Points from** 🡺
* <https://stackoverflow.com/questions/24378733/ejb-transaction-attribute-overriding-in-method-level>

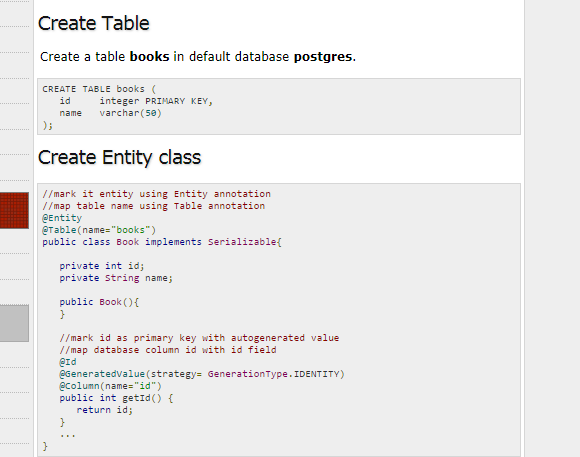
# [EJB Transaction attribute overriding in method-level](https://stackoverflow.com/questions/24378733/ejb-transaction-attribute-overriding-in-method-level)

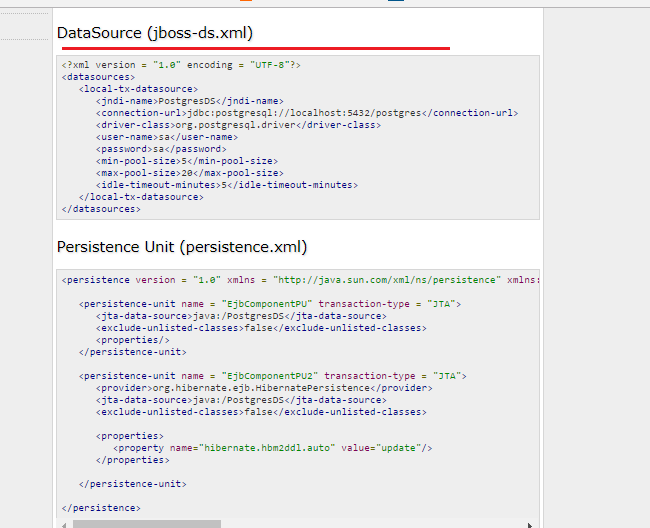
# Question from the above website is 🡺[EJB Transaction attribute overriding in method-level](https://stackoverflow.com/questions/24378733/ejb-transaction-attribute-overriding-in-method-level)

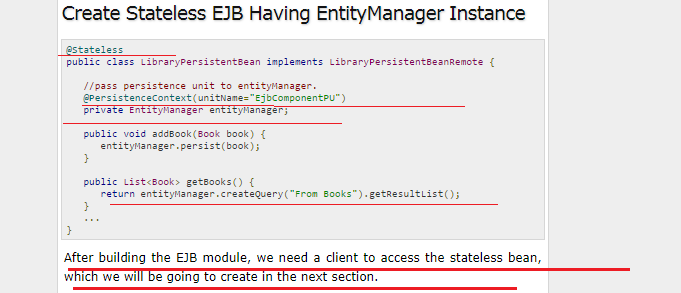
# Example🡺

# EJB - Persistence









For complete example see the tutorials point

EJB Timerservice🡺

1) our real world is completely different from this tutorial point

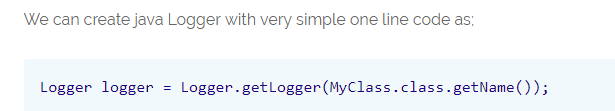
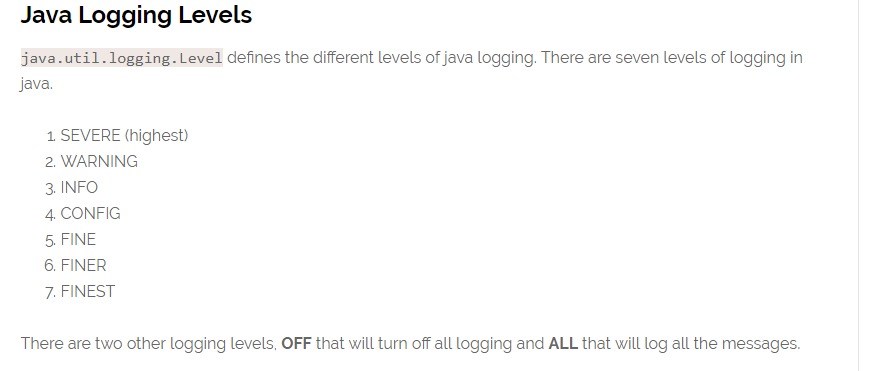
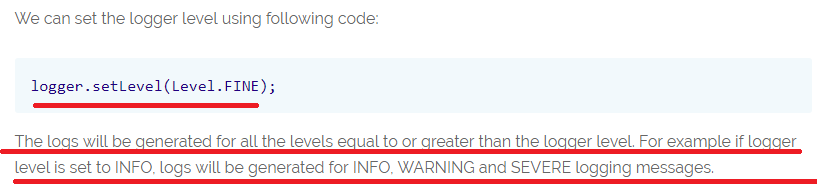
2) We had used @Schedule 🡺 javax.ejb.schedule

3) And also our timer class is annotated with

* @Stateless
* @Resource 🡺 for type attribute as datasource
* @RunAs(“TECH\_USER”) 🡺javax.annotation.security.RunAs

Logging

<https://www.journaldev.com/977/logger-in-java-logging-example>

1. **Java Logging** API was introduced in 1.4 and you can use java logging API to log application messages.
2. java.util.logging.Logger is the class used to log application messages in java logging API. 🡺 we had used this only in our real world
3. 
4.  🡺 but in our real world we had logger.entrying(“”) and logger.exiting(“”) also rt? 🡺 and note this entry and exit content was not getting printed on to the log file
5. 

### Java Logging Handlers

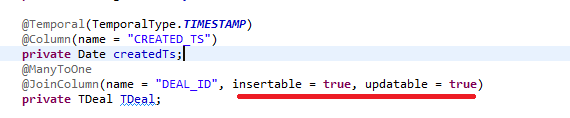
1. We can add multiple handlers to a java logger and whenever we log any message, every handler will process it accordingly. There are two default handlers provided by Java Logging API.

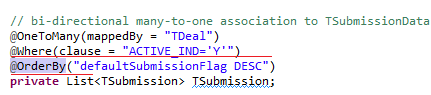
* **ConsoleHandler**: This handler writes all the logging messages to console
* **FileHandler**: This handler writes all the logging messages to file in the XML format.

1. We can create our own custom handlers also to perform specific tasks. To create our own Handler class, we need to extend **java.util.logging.Handler** class or any of it’s subclasses like StreamHandler, SocketHandler etc.

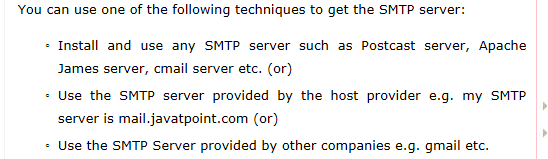
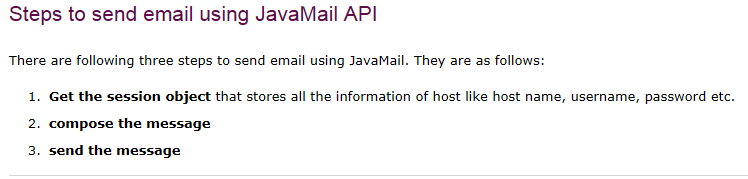
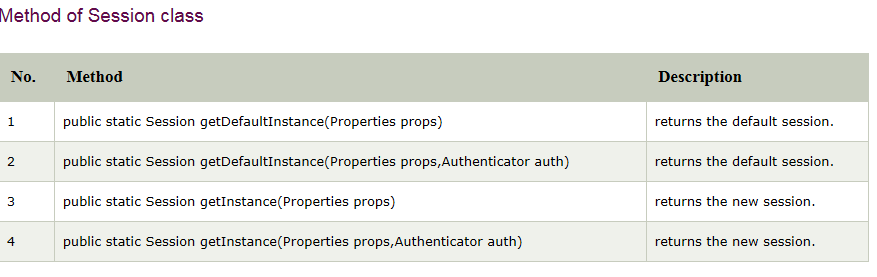
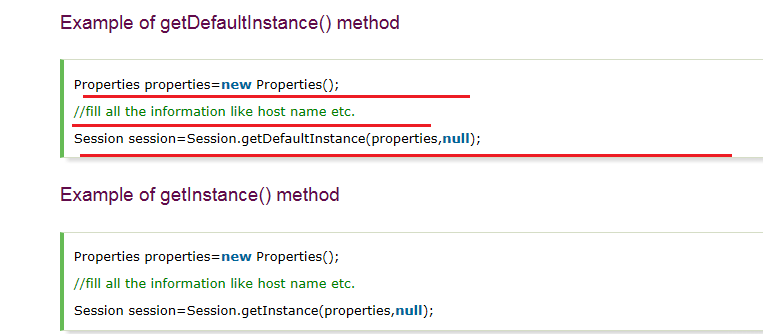
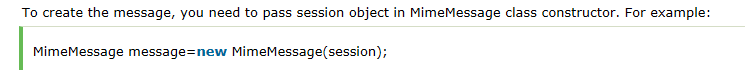
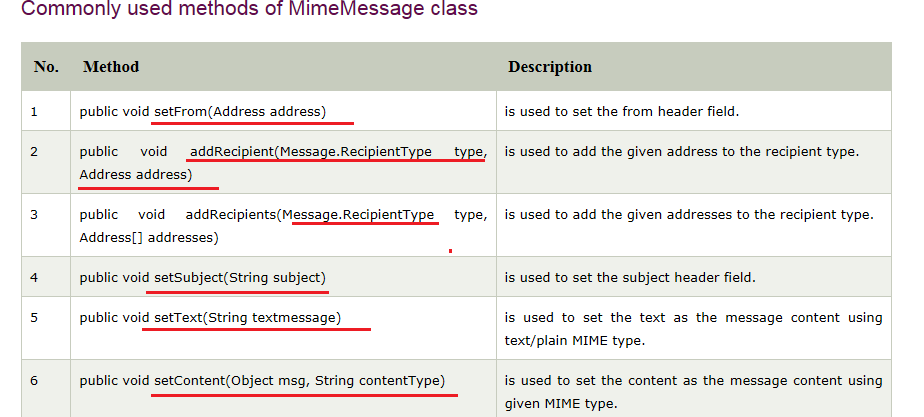
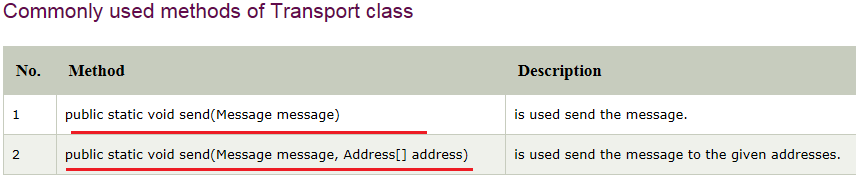
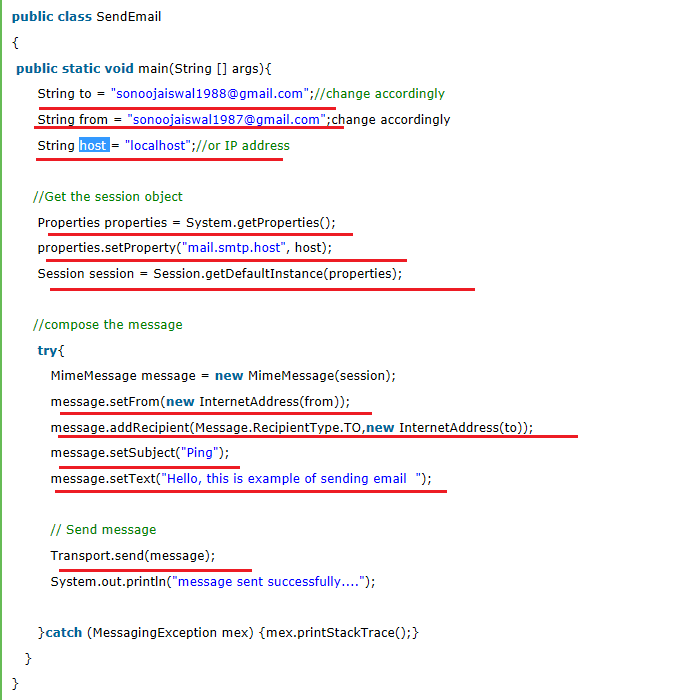
JPA

1. @PersistenceContext(“Persistence Unit Name”) 🡺 this is what we use to get our EM object in our app
2. Now lets see few of the attributes of @JoinColumn 🡺 In our case one deal can have n no of submission is relationship is one to many between deal and submission and

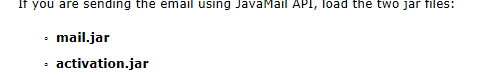
Many to one between submission and deal 🡺s

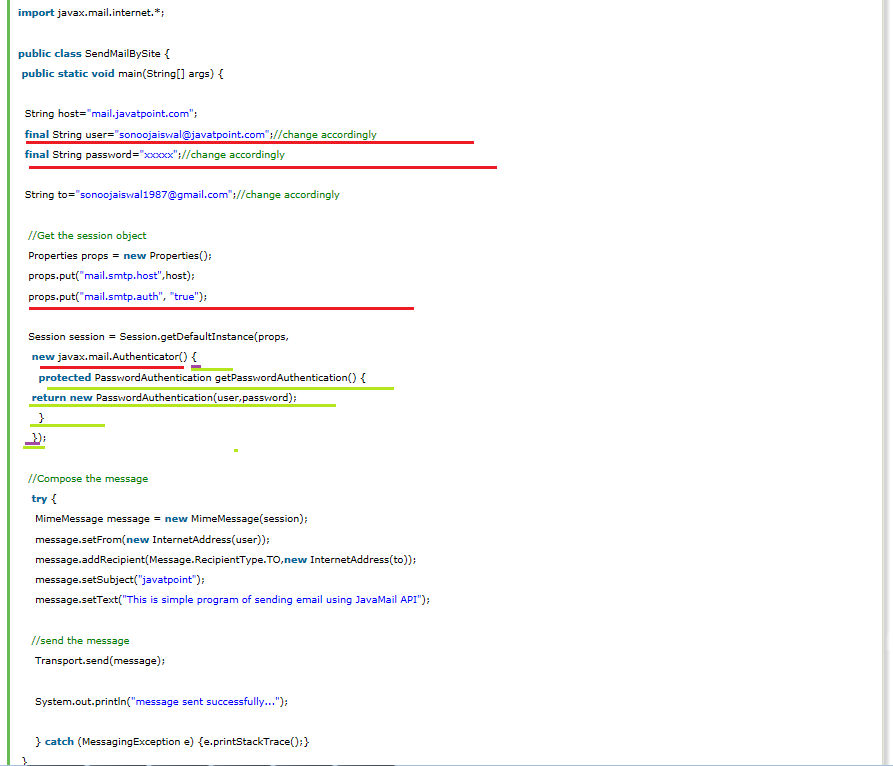
1. @SequenceGenerator already we had seen🡺 see earlier notes
2. @OrderBy(“name of the property that should be ordered”)🡺 is there is any advantage or reason for adding @orderBy or needed to know how this works
3. 
4. A

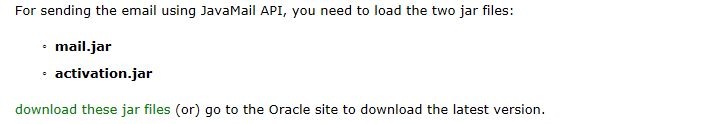
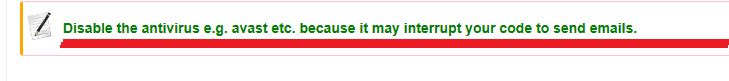
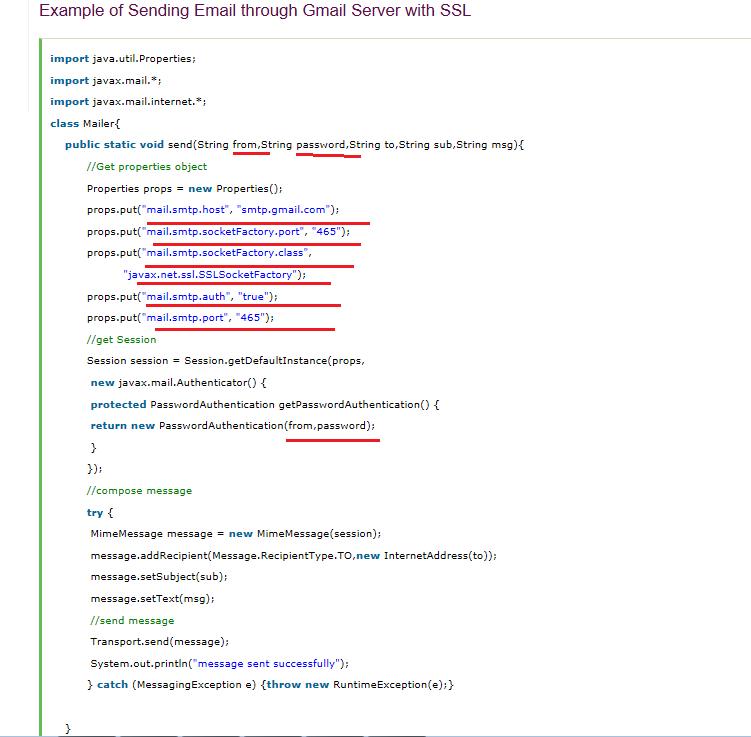
Mailing

1. We use JavaMail API to send an email
2. There are various ways to send email using JavaMail API. For this purpose, you must have SMTP server that is responsible to send mails
3. 
4. 
5. Get the session object 🡺The **javax.mail.Session class** provides two methods to get the object of session, Session.getDefaultInstance() method and Session.getInstance() method. You can use any method to get the session object.
6.  🡺 ms here 2nd parameter null indicates any authentications🡺 an example is covered below
7. **Compose the message 🡺**
8. The javax.mail.Message class provides methods to compose the message. But it is an abstract class so its subclass javax.mail.internet.MimeMessage class is mostly used.
9. 
10. 
11. 
12. **Send the message 🡺The javax.mail.Transport** class provides method to send the message.
13. 
14. **Simple example of sending email in Java**
15. In this example, we are going to learn how to send email by SMTP server installed on the machine e.g. Postcast server, Apache James server, Cmail server etc. If you want to send email by using your SMTP server provided by the host provider, see the example after this one
    * + - 1. For sending the email using JavaMail API, you need to load the two jar files: **mail.jar** and **activation.jar**
          2. 
16. **Example of sending email in Java through SMTP server provided by the host provider**

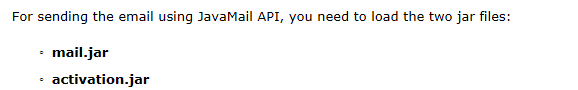
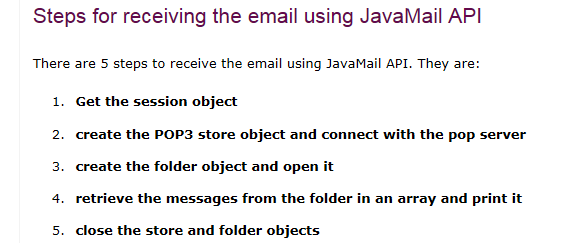
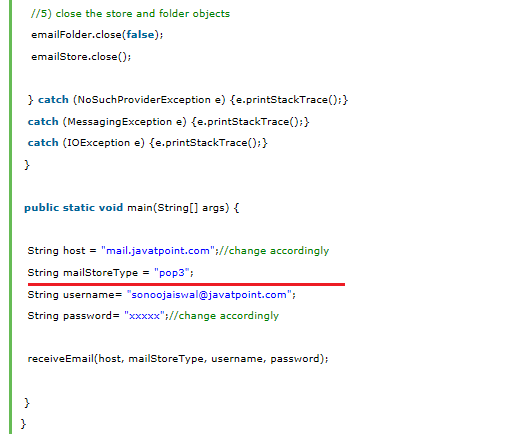
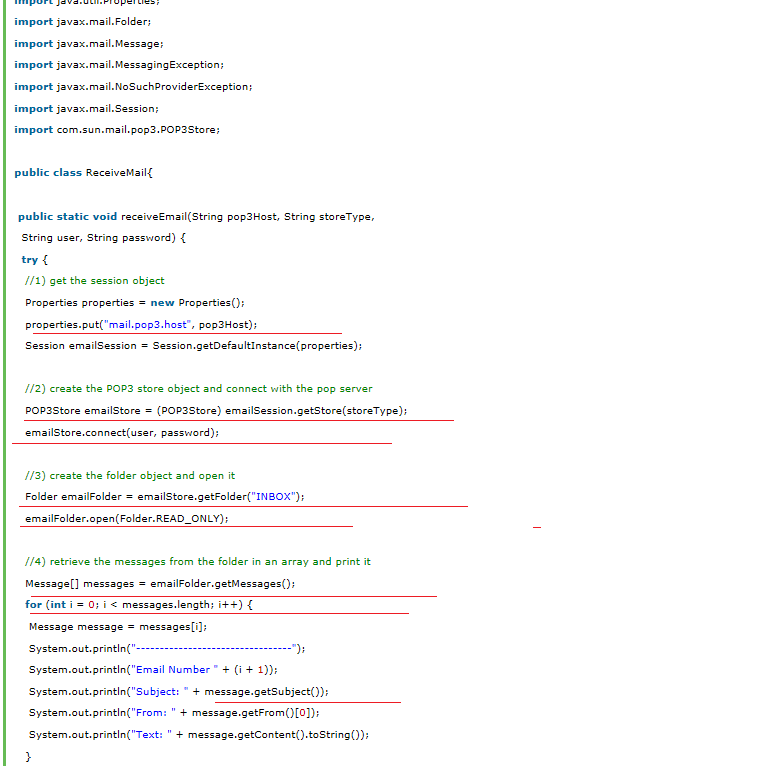
**If you are using the SMTP server provided by the host provider e.g. mail.javatpoint.com , you need to authenticate the username and password. The javax.mail.PasswordAuthentication class is used to authenticate the password.**

🡺[download these jar files](https://www.javatpoint.com/src/mail/mailactivation.zip) or go to the Oracle site to download the latest version.

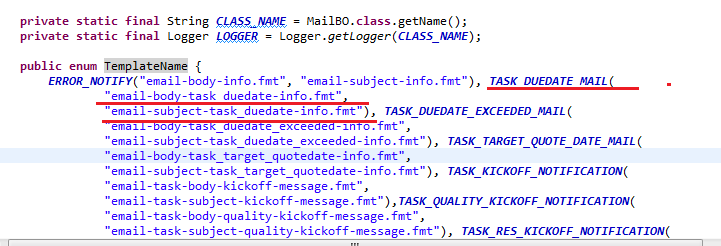
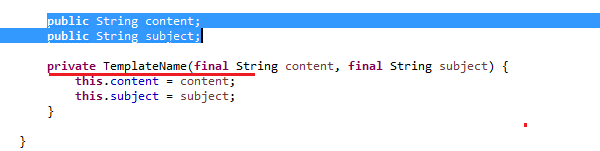
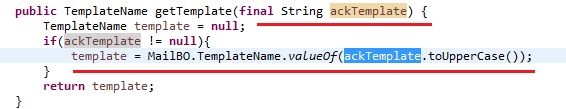
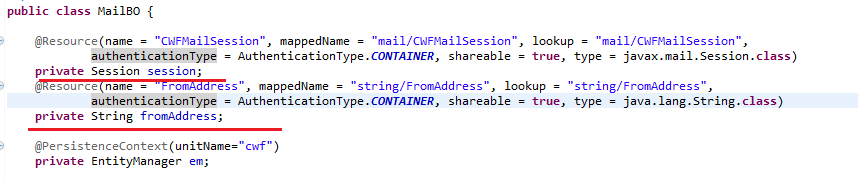


1. **Sending Email in Java through Gmail Server**
2. We can send email by using the SMTP server of gmail
3. Here we will learn how to send email through gmail server by SSL (Secured Socket Layer)
4. SSL is basically used for security if you are sending email through gmail server.
5. Steps
6. 
7. 
8. 
9. **Resolving AuthenticationFailedException 🡺**Click on this link and click on turn on radio button to allow users to send mail from unknown location. <https://www.google.com/settings/security/lesssecureapps>

**Receiving email in Java**

1. For receiving email Store and Folder classes are used in collaboration with MimeMessage, Session and Transport classes.
2. Now let’s see the steps for receiving an email
3. 
4. 
5. 

**NOTE🡺 NEEDED MORE INFORMATION ON THE “STORE” AND “POP SERVER”**

1. An example on our real world
2. Here in this case we have a enum inside a class🡺
3. 
4. 
5. 
6. 



