1. **Difference between SQL and PL/SQL**

|  |  |
| --- | --- |
| SQL | PL/SQL |
| Simple query language | Procedural Language along with Simple Query Language Features |
| Suitable when we execute single update / insert / and delete operations on database | Suitable for executing the multiple inserts / updates / deletes on database |
|  |  |
| It doesn’t not support programming | It supports programming |
| Loop:  It doesn’t loop through the records | Loop:  It can loop through the records and manipulate the single record at a time |
| Exceptions:  Exceptions cannot be handled | Exceptions:  Exceptions can be handled |
| Code Secure:  It doesn’t stored in database and cannot be encrypted | Code Secure:  It can be stored in database and can be encrypted |
| Complex Data Types:  Cannot work with complex data types such as records, objects, collections,cursors,arrays | Complex Data Types:  It can support the complex data types such as records,objects,arrays,collections,cursors |
| It specifies what to do | It specifies what to do and how to do |
| It is used to code the queries, DML and DDL statements | It is used to code the blocks of code ,triggers, stored procedures, functions and packages |
| It can be embedded into Pl/SQL | It cannot be embedded into the SQL |

1. **What is SQLInjection and how to handle it?**

It is a code injection technique, used to attack the data driven applications in which malicious SQL statements are inserted into an entry field for execution. It is an technique which malicious users can inject the SQL commands into the SQL statement via web page input in the web page request. SQL injection is one of the top web page vulnerabilities and it affects the web application security.

**Considers this server code,**

txtUsrId=request.getPameter(“userId”)

txtSqlQuery=”select \* from User where userId=”+txtUsrId

**Scenerio’s when SQLIjection happens for the above server code:**

1. Based on 1=1 is always true

When we input user id field as ,

User ID : 154 or 1=1

The query looks ,

Select \* from User where userId=150 or 1=1

1=1 always true in sql execution , so returns all user details from the table User (including username and password)

1. Based on “”=”” is true

Hackers can easily login to the application by simply inserting the “ or “”=”” in the user anme and password field.

User Id : “ or “”=””

Password : “ or””=””

It will create the query like this ,

Select \* from User where userId=”” or “”=”” and password=”” or “”=””

It will return all the user name and password from the user table

1. Based on Batch SQL statements

User Id : 105 ; Drop Table Suppliers;

It will create query like this,

Select \* from User where userId=105 ; Drop table Suppliers;

Since these are separated by semicolon , both statements are executed and Supplier table will be dropped.

1. **How to handle the above SQL injections in java?**

In java , We have to use parameterized queries and can be done by using the PreparedStatement to execute the query as like,

PreperedStatement ps=con.prepareStatement(“select \* from user where userID=?”);

Ps.setString (1, txtUsrId);

Prepare statement setter methods always truncates the special characters in the input

Use callable statement to execute the stored procedures and functions. This statement itself is not vulnerable to SQLInjection.

1. **How to handle the SQL injections in ORM frameworks such as hibernate ?**

We have to use NamedQuery instead of using the normal query.

**Ex:**

Query namedQuery=session.createNamedQuery(“queryName”);

namedQuery.setString(1,txtUsrId);

List list=namedQuery.list();

1. **Explain JNDI ?**

It is an Java Naming and Directory Interface is an API (Application Programming Interface) used to access the services which is reside on the application server by its name. Which allows distributed applications lookup services in an abstract and resource independent way. To use the JNDI , we need JNDI classes and service providers. JDK itself has the service providers to access the following JNDI services

* LDAP (Lightweight Directory Access Protocol)
* RMI (Remote Method Invocation)
* CORBA – Common Object Request Broker Architecture
* DNS – Domain Name Service

**When it is used?**

It is used to set up the database connection pool in the J2EE Server. Any applications that is deployed in that server get the database connection using the JNDI name (“java/testJNDI”) without know about the database connection details.

**Advantages:**

When we move code to the different environment ( dev->QA->SIT->UAT->PROD ) we can use the same JNDI name in all environments and hide the database details. Applications doesn’t require any change when they promote to different environments. In this way, we can restrict the production database credentials to be shared to many users except server admin.

1. **How to use the JNDI to lookup and get the data source connection ?**

* Obtain the Initial Context

InitialContext ctx=new InitialContext()

* Use JNDI lookup to obtain the datasource reference

DataSource ds=ctx.lookUp(“JNDIDataSourceName”)

* Use the datasource refrence to get the connection

Connection con=ds.getConnection()

1. **Why do we need spring ? I have an java directly connect hibernate then why I need spring ?**

Spring is an loosely coupled framework , it provides loose coupling between plain java component and hibernate component (framework). It is an framework that helps to wire the difference components together. We can use spring, When we have lot of components , and wish to combine it in such a way and when we want to replace the one component with another based on different settings and different environments . It is also useful to control the lifecycle of the java bean , via IOC container. It provides easy unit testing , since all the fields has the setters methods which can be easily mocked by using the mocking framework. It a good framework for web programming. Spring supports less code to write . It provides an good segregation on service layer , web layer and business layer. Basically we can choose spring for creating the distributes application in a loosely coupled way and it provides easy maintanence.

We can use spring only we have only J2EE server or Servlet Container but not having the Application Server.

1. **What is join ? How many types are available ?**

Join is an SQL technique used to retrieve records from more than one table in efficient manner.

There are four types of joins are available,

1. Inner Join :

Retrieve records from two table based on the conditions met.

Select c.firstName,c.lastName ,o.orderDate,o.orderAmount from Customer c inner join orders o on c.customer\_id=o.cust\_id

1. Left outer join:

Retrieve all records(rows) from left side table and only matched records from right side table based on the conditions specified. The left side table columns for the right side table records are filled with NULL.

Select c.firstName,c.lastName ,o.orderDate,o.orderAmount from Customer c left join orders o on c.customer\_id=o.cust\_id

1. Right outer join:

Retrieve all records from right side table and only matching records from left side table based on certain condition. The right side column for the left side table record are filled with NULL.

Select c.firstName,c.lastName ,o.orderDate,o.orderAmount from Customer c right join orders o on c.customer\_id=o.cust\_id

1. **Difference between component scan and Package Scan ?**
2. **How to define spring bean using the annotations ?**

@Configuration

public class SpringConfiguration{

@Bean

public TransferService transferService(){

return new TransferService();

}

}

Public Class TransferService{

Private String serviceMessage;

//getters

//setters

}

@Configuration :- indicates that particular class acts as configuration class and source of bean definitions . Which is similar to <beans/> in xml based configuration. This is the class level annotation.

@Bean :- This is the method level annotation used to define an bean. The method which is annotated with @bean return the bean of specific type and registered with beanfactory or application context. Which similar to <bean/> in xml based configuration. Therese are the properties can also be used with @bean .(intiMethodName(init-method in xml), destoryMethodName(destroy-method in xml), Scope and AutoWire)

And it should be retrieved as

ApplicationContext context=AnnotationConfigApplicationContext();

Context.register(SpringConfiguration.class);

TransferService transferService=context.getBean(TransferService.class);

transferService.setServiceMessage(“test”);

transferService.getServiceMessage();

1. **How to use dependency Injection using Spring Annotations?**

Dependency injection is done by passing the one bean method to other bean definition.

public class SpringConfiguration{

@Bean

public TransferService transferService(){

return new TransferService(localTransafer());

}

@Bean LocalTransfer localTransafer(){

return new LocalTransfer();

}

}

Here transfer service bean getting reference of localtransafer bean via constructor injection.

1. **What is Annotation based configuration in Spring?**

From spring 2.5 , it is possible to do the dependency injection via annotation instead of wiring the beans in XML. Annotation injection is performed before than the XML injection , so if we use both the XML injection override the annotation injection.

Annotation configuration is not enabled by default in spring container. It can be enabled by,

<beans >

<context:annotation-config />

</beans>

Once it is enabled , spring can do the annotation wiring in constructors , fields and methods.

Annotations Used:

**@Required:**

It is used for the setter method of the property. If the setter method is specified with @Required then it should be set along with the bean definition in bean configuration file , other wise it throws BeanInitilizationException (applicationContext.xml)

Ex: public class Employee{

Private String name;

Private int age;

//getter

//setter

@Required

Public void SetName(String name){

}

//getter and setter for age property

}

In bean configuration file,

<beans>

<context:annotation-config/>

//bean definition

<bean id=”emp” class=”Employee”>

<property name=”name” value=”Vasantha” />

</bean>

</beans>

@Required property value must be supplied at configuration time in bean configuration file . The other property “age” does not need to be set at the time of bean initialization.

**@Qualifier:**

It is an field level annotation. When we need to create two beans of the same type and want to refer only one at a time for the specific property , we can use @Qualifier along with @Autowired annotation.

EX:

public class Employee{

Private String name;

Private int age;

@Autowired

@Qualifier(“homeAdd”)

Private Address address;

//getter

//setter

@Required

Public void SetName(String name){

}

//getter and setter for age property

}

Public class Address{

Private int doorNo;

Private String streetName;

//getters and setters

}

In bean definitions xml file(applicationcontext.xml),

<beans>

<context:annotation-config/>

//bean definition

<bean id=”emp” class=”Employee”>

<property name=”name” value=”Vasantha” />

</bean>

<bean id=”homeAdd” class=”Address”>

</bean>

<bean id=”officeAdd” class=”Address”>

</bean>

</beans>

@Qualifier specifies the bean id from the bean definitions file, in the above Employee class the address field autowired with the bean “homeAdd” eventhough we have “officeAdd” bean of same type.

**@Autowired:**

It can be used with constructors , fields(properties) and methods (setter). It can be used for the dependency injection. It can be used for automatic dependency injection.

* **Autowired on Constructors: (Constructor based Autowiring)**

It can be used with constructors. If the constructors is annotated with @Autowired , it should be included in the bean definition when creating the bean, even if no <constructor-arg> element is specified in the bean definition XML file.

EX:

Public class Employee{

Private Address address

Public Employee(Address address){

this.address=address

}

}

Public class Address{

Private int doorNo;

Private String streetName;

//getters and setters

}

In bean definitions xml file,

<beans>

<context:annotation-config/>

<bean id=”emp” class=”Employee” />

<bean id=”address” class=”Address” />

</beans>

* **Autowired on properties:**

It can be applied to fields or properties. It can be get rid of defining the setter method in the bean class.

Ex:

Public class Employee{

@Autowired

Private Address address ;

//getters method setters

}

Refer Address class definition on the previous section.

In the bean definition configuration file,

<beans>

<context:annotation-config/>

<bean id=”emp” class=”Employee”/>

<bean id=”address” class=”Address” />

</beans>

Though the Employee class doesnt have the setter method for the address property , it will automatically injected at the time of bean creation based on “byType” injection.

* **Autowired on settermethod: (setter based injection)**

It can be applied to the setter method on the property. The setter method annotated with @Autowired ,then it doesn’t require the <property> element as part of bean definition in the bean configuration file. It auomatically the wire the bean on byType autowiring method. It basically get rid(be freed ) of <property> element in the bean configuration file.

Ex:

Public class Employee{

Private Address address ;

//getter method

//setters

@Autowired

Public void setAddress(Address address){

this.address=address;

}

}

Refer Address class definition on the previous section.

In the bean definition configuration file,

<beans>

<context:annotation-config/>

<bean id=”emp” class=”Employee”/>

<bean id=”address” class=”Address” />

</beans>

The “emp” bean doesn’t have the <property> element defined for the property “address” . But once the bean is instantiated , address property or filed is injected with respective Address type bean “address” since the setter method is @Autowired.

By default , @Autowired annotation requires the dependency, we can disable the default autowiring by setting the @Autowired(required=false).

1. **What is JSR Annotations supported by Spring?**

JSR is Java Specification Request. Which common for all J2EE technologies. Spring also supports JSR annotations.

**@PostConstruct:**

This is method level annotation. The method with @PostConstruct is called once the bean is created and the properties are set(after all dependency injection is set). This is an alternate for the initialization call back method “init-method”. Only one method should be annotated with this annotation. The return type should be void and should not take any parameters. It should not throw any checked exception. The method cannot be static may be final.

**Ex:**

Public class Employee{

@PostConstruct

Public void initilizeEmployee(){

}

}

**@PreDestroy:**

This is also method level annotation. The method with this annotation is called before the bean is removed from the container. This is an alternate for the destruction call back method “destroy-method”. Only one method should be annotated with this annotation. The return type should be void and should not take any parameters. It should not throw any checked exception. The method cannot be static may be final.

**Ex:**

Public class Employee{

@PreDestroy

Public void destroyEmployee(){

}

}

**@Resource:**

This is applicable for a method, fields and component class . It is used to inject the bean based on its name(enables byname injection).

**Ex:**

Public class Employee{

Private Adresss address;

@Resource(name=”homeAdd”)

Public void setAddress(Address address){

This.address=address

}

}

@Resource = @Autowired + @ Qualifier

If the name attribute is not used , it inject the bean with the same name on the fields or parameter of the setter method . If this annotation is appliedon component class, it makes the class is auto detectable at runtime.

1. **What are the important spring annotations used?**

The most frequently used annotations are,

**@Component:-** Indicates that the annotated class is a component. These type of component available for auto detection when the annotation based configuration or class –path scanning is used.

**@Repository:-** It is an special type of @Component. Used in persistent framework layer.

**@Controller:-** It is an special type of @Component. Used in Presentation layer

**@Service:-** It is an special type of @Component. Used in Service Layer

**@Configuration:-** The class annotated with @Configuration act as configuration class as like old XML. It contains bean definitions annotated with @Bean. It is similar to <beans/> in XML. It is responsible for having the logic for instantiating , initializing and configuring objects that will be managed by spring IOC Container.

Ex:

@Configuration

Public class SessionFactoryConfiguration{

@Bean

Public DataSource dataSource(){

DataSource dataSource=new DataSource();

dataSource.setURL(“”);

dataSource.setDriverClassName(“”);

dataSource.setUserName(“”);

dataSource.setPassWord(“”);

return dataSource;

}

@Bean

Public SessionFactory sessionFactory(){

SessionFactory sessionFactory=new SessionFactory(

//Setter injection

sessionFactory.setDataSource(dataSource()));

sessionFactory.setShowSQL(true);

return sessionFactory;

}

}

**@Bean:-** Method level annotation used to define the bean .It same as <bean> in XML.

**@Scope:-** By default all beans in spring IOC is Singletone. Scope can be defined for the bean using the @Scope.

Public class MyConfiguration{

@Bean

@scope(StandardScopes.PROTOTYPE)

Public TransferService transferService{

Return new TransferService();

}

}

In the above code , the bean default scope is overridden for the bean transferServcie. The new bean will be created for every client request.

**@AutoWired:-** This annotations is applicable for the constructors , fields , setters and config methods to provide the dependency injection provided by the spring. (constructors,fields,setters explained in previous question)

* Config methods: In java based Configuration ,One configuration class can directly refer the bean in another configuration class using @Autowired

Ex: @Configuration

Public Class ConfigOne{

@Bean

Public AccountRepository accountRepository(){

Return new AccountRepository();

}

}

@Configuration

Public class ConfigTwo{

@Autowired AccountRepository accountRepository;

@Bean

Public TransferService transaferService(){

return new TransferService();

}

}

Here, the AccountRepository bean is creatde in ConfigOne and will be Autowired in the AccountRepository field in ConfigTwo.

**@Import :-** Used at classlevel . Used to import one or more configuration classes. Which is equivalent to <import/> in XML.

For clarity and modularity , Bean definitions can be split into multiple @Configurations files. All these configuration files are aggregated using the @Import .

EX:

@Configuration

Public Class DataSourceConfig{

@Bean

Public DataSource dataSource(){

return new DriverManagerDataSource(…);

}

}

@Configuration

@Import(DataSourceConfig.class)

Public Class AppConfig(){

@Bean

Public transferService TransferService(){

return new TransaferService();

}

}

One or more classes can be impoerted by,

@Import(DatasourceConfig.class,DataSourceConfig1.class)

**@Lazy:-**

It is used along with @Configuration and @Bean. If The bean definition method annotated with @Lazy , spring container will not instantiate the bean at the time container loads. Instead it will be instantiated when it is used in the code at first time.

**Ex:**

@Configuration

Public class AppConfig{

@Bean

@Lazy(value=true)

Public TransferService transferService(){

Return new TransferService();

}

}

1. **What is the difference between init () and afterpropertiesSet() in spring ?**

Both are used to initializing the bean after the dependencies are injected. If the bean implements the InitilizingBean and overrides afterPropertiesSet then the order of execution will be @PostConstruct , afterPropertiesSet(), then init-method is called.

1. **What is the spring bean lifecycle ?**

BeanInstanitation->populate properties(dependency injection) - > BeanNameAware (setBeanName() ) -> BeanFactoryAware (setBeanFactory()) - > PreinitilizationBeanPostProcessor -> Initializing bean via afterProeprtiesSet() -> initializing bean via custom init-method -> PostInitilizationBeanPostProcessor -> **Bean is ready to Use** -> when the spring container shutdown , DisposableBeans(destroy()) -> custom-destroy method

Ex:

public class InitSequenceBean implements InitializingBean {

public InitSequenceBean() {

System.out.println("InitSequenceBean: constructor");

}

@PostConstruct

public void postConstruct() {

System.out.println("InitSequenceBean: postConstruct");

}

public void initMethod() {

System.out.println("InitSequenceBean: init-method");

}

@Override

public void afterPropertiesSet() throws Exception {

System.out.println("InitSequenceBean: afterPropertiesSet");

}

}

**Output :**

**InitSequenceBean: constructor**

**InitSequenceBean: postConstruct**

**InitSequenceBean: afterPropertiesSet**

**InitSequenceBean: init-method**

1. **What is websphere objectgrid ?**

**Ref ObjectGrid API:** [**https://www.ibm.com/support/knowledgecenter/SSTVLU\_8.6.1/com.ibm.websphere.extremescale.javadoc.doc/topics/com/ibm/websphere/objectgrid/package-summary.html#package\_description**](https://www.ibm.com/support/knowledgecenter/SSTVLU_8.6.1/com.ibm.websphere.extremescale.javadoc.doc/topics/com/ibm/websphere/objectgrid/package-summary.html#package_description)

It is an distributed cache product available on websphere application server deployment platform. The main interface used here is ObjectGrid. The JVM needs to create atlease one instance of this type.

It is designed to be as data cache ,collects data from multiple source and available to the client of the ObjectGrid. It can store large quantities of the data. The stored data are access by using the ObjectGrid API’s.

**Working on ObjectGrid:**

An application can have multiple ObjectGrid Instances and each has its own config file.

An ObjectGrid instance needs to be created first,

ObjectGrid obg=new ObjectGridImpl(); 🡪 ObjectGrid instance is created

BackingMap bm=obg.defineMap(“TABLE”); -> Map defined with the created ObjectGrid

BackingMap setter method is used to configure the map.

**Working on the ObjectGrid Session:**

Each thread has its own session to work on the ObjectGrid . The OBjectgrid getSession() method returns the session. Once the thread gets the session , it can access the ObjectMap to work on the manipulation of the data available on the ObjectGrid. Commit / Rollback /begin method from the session are used to handle the transation.

Session session=obg.getSession();

ObjectMap map=session.getMap(“TABLE”);

Session.begin();

MyData d1=(MyData)map.get(“key1”);

Session.commit();

1. **Name some distributed cache products?**

* **SwarmCache**
* **JBoss TreeCache**

1. **How to handle singleton in distributed environment or clustered environment?**

Singleton means once per JVM. In distributed environment , there will be multiple JVM and there will be one singleton instance per JVM. Singleton is used when we have some configuration details that needs to be shared across the application. So Singleton instance holds this config data that can be accessed throughout the application. Singleton pattern restrict to create more than one instance at single JVM.

There are so Many options are available to maintain the singleton across multiple JVM’s in distributed environment.

1. **Usage of RMI :**

We can register the singleton instance in the RMI registry in one node and provide the stub in the clustered JNDI tree to make it available across all the nodes. All other nodes should be a RMI clients. The disadvantage is , single point of failure. We should have a failure over node for central node which also has the singleton instance should be registered with the clustered JNDI tree when there is no singleton stub registered by the central node.

1. **Usage of JMS:**

There may be situation, singletons are available in multiple JVM’s as data cache.(map). Whenever data changed in data cache it needs to published into TOPIC using the Java Messaging API. Every container has listener which listens the topic and get the message data and update into their cache. Disadvantage here is , JMS messages tales time to process , so there may be some time the caches are out of sync.

1. **Usage of Distributed Cache:**

To overcome the problem with JMS. We can go for distributed singleton cache. SwarmCache and JBoss TreeCache are the good example of distributed caches.

1. **Usage of Vendor Products:**

There are so many vendor products are available to maintain the singleton in the clustered environment.

* **Terracotta / Oracle Coherence:**

Which use the concept of in-memory object replication to provide the singleton view across all JVM’s.

* **JGroups :**

It is a toolkit for reliable messaging. It creates the cluster whose node can send messages to each other. It allows to form the group and applications and JGorups send messages to everyone in the group so that they can be in sync.

* **JBoss - HASingtonService :**

Singletone service is deployed in all nodes in a cluster but only one runs. This is called master node. When the master fails , another master will selected from the cluster and the service is restarted.

* **Weblogic - SingletonService:**

Only one instance runs in the cluster and all clients are lookup the same instance.

* **Websphere - ObjectGrid**

Websphere supports singleton across cluster via ObjectGrid.

1. **What are the advantages of Clustering?**

* High availability
* High Performance
* High Fault Tolerance

There are two important steps involved in the clustering.

* **Load Balancing :**

When multiple request from multiple clients comes in ,there may a situation one server cannot handle these request. So there will be load balancer placed between the client and server and routes these request to different servers for doing the same function. The example is servlet. This makes the high availability and high performance.

* **Fail Over :**

when client makes lots of request and the server failed to handle this request in between , there will be a fail over systems which detedcts this failure and routes the request to different available server instance.

1. **How hibernate second level cache works?**

Caching is the facility provided by the ORM framework ,which speeds up the web applications by reducing the number of database hits.

**Hibernate supports 2 types of caching:**

* **First level :** Associated with session objects. Its enabled by default.
* **Second Level :** Associated with session factory object and its global to application. And its not enabled by default. It has to be manually enabled in hibernate properties.

There are 4 different second level cache providers available from different vendors,

^ EHCache (Easy Hibernate Cache) from hibernate framework

^ OSCache from Open Symphony

^ SwarmCache

^ JBoss TreeCache

There are 4 different cache usage available,

1. Read-only : cache will work only for read operation
2. Nonstrict-read-write : cache work for read and write but one at a time
3. Read-write : cache work for read and write simultaneously
4. Transaction : cache work for transaction

EHCache is the cache mostly used in the hibernate frame work which support first 3 usage and not act as transaction cache.

**Enable Second Level Cache: (4 steps)**

1. Add the cache provider in hibernate properties section in the config file.

<properties name=”hibernate.cache.provider\_class” > org.hibernate.cache.EhCacheProvider

</property>

1. Set the second level cache property as true

<property name=”hibernate.cache.use\_second\_level\_cache” > true </property>

1. Specify the cache element for the class in hbm file with appropriate value after the class element.

<cache usage =”read-only” />

1. Create ehcache.xml file

Ex:

<ehcache>

<defaultCache

maxElementsInMemory=”100”

eternal=”false”

timeToIdealSeconds=”120”

timeToLiveSeconds=”200”

>

</defaultCache>

<cache name=”com.Employee”

maxElementsInMemory=”100”

eternal=”false”

timeToIdealSeconds=”120”

timeToLiveSeconds=”200”

/>

</ehcache>

From the above file default cache is used for all persistent classes. We can also define the cache for the specific persistent class explicitly.

If we specify the eternal=”true” , we no need to specify the ideal and live seconds, hibernate itself handle this.

timeToIdealSeconds – defines how many seconds object can be ideal in the second level cache

timeToLiveSeconds – defined how many seconds object can be at second level cache whether its ideal or not.

**Second Level Cache Working Strategies:**

1. When the first session try to load an entity at very first time , it will not be available in first level(session) cache and second level cache. So it hit the database and load an entity to first level cache also to second level cache.
2. Second time session try to load the same entity , it look for the first level cache for the cached entity , if its available then load the entity
3. If the cached entity is not available on the first level cache then it look for the second level cache for the cached entity.
4. If the entity is available in the second level cache ,its copied to first level cache and loaded
5. Now, second session comes in try to load the entity , it lookfor the first levelcache (it will not be available) , then look for the second level cache(available in session factory) , so load the entity into first level cache and return the result.
6. Read and write operation on the cached entity is control by the caching mode specified in the class mapping of the each entity.
7. **How to use the hibernate second level ehcache in clustered environment ?**

**http://www.surekhatech.com/blog/hibernate-second-level-cache-using-ehcache-for-cluster-environment**

[**http://www.ehcache.org/documentation/2.8/replication/**](http://www.ehcache.org/documentation/2.8/replication/)

1. **What is AJAX ?**

AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. It is an extended javascript used for creating the interactive and most faster web applications. It just referes the component instead of loading the page every time. It is a technology helps to communicate the data with the server. It is used to make the Asynchronous call with the server.

1. **Can we make the synchronous call with AJAX ?**

When we use ajax with JQuery we jave an option to make the synchronous ajax call by setting the flag **async: false**. But synchronous ajax calls not prefred , since the browser will wait unitl the request gets completed.

**EX:**

function getRemote() {

//var remote;

return $.ajax({

type: "GET",

url: remote\_url,

async: false,

success : function(data) {

remote = data;

}

});

//return remote;

}

1. **Explain about Queue & Topic ?**

Queue is persistent . it is an point to point message communication. If the message comes into to the queue , once it is read by receiver , it gets deleted from the queue. If the receiver is offline , the message still exist in the queue.

Topics are non-persistent. It is an public-subscribe message communication. If the message comes into the topic, once its subscribed by all the subscriber it gets deleted. If the subscribers are offline , the messages gets deleted in general. IF we want to make the topic persistent and flexible like queue , we have to subscribe as durable subscriber.

Durable subscription use unique name to subscribe with the topic. So it the subscribers are offline the messages are stored in the topic with the subscriber subscription name. Once the subscriber the come to online , the messages with their name will be delivered.

1. **What is LDAP ?**

**Sanitize :-** To make cleanor hygienic , to remove unacceptable characters.

Lightweight Directory Access Protocol(LDAP) is an open source protocol used for querying and manipulating the information directories. It runs over transaport protocols such as TCP protocol.

1. **What is LDAP Injection ? how to prevent LDAP Injection Vulnerabilities ?**

**Arbitrary:-** based on Random choice without any reason or system.

Web application uses user input to create the LDAP statements. LDAP Injection is an technique used to create the custom LDAP statements based on user input in the web applications for dynamic web page requests. Which will create the security issue in web application. When an web application is not properly sanitize the user input , it will create some arbitrary commands such as granting permission to unauthorized queries , changing the data in LDAP tree. The attackers can use local proxy and they can change the data in the http request which constructing the LDAP statements.

Protecting the LDAP enabled web application is an responsible for the application developer and LDAP Aministrator.

**Incoming Data Validation:**

Prevent this the user input must need to be validated on server side. Most the of the code injection techniques uses special characters for the malicious attacks, so this special characters needs to be cleaned up from user input by the use of input based filters.

**Ex:**

To retrieve the employess who work on particular manager , here is the LDAP query

DirectorySearcher searcher=new DirectorySearcher(“(manager=’+**managerName.Text**+)”);

managerName.Text is an parameter value from httprequest. Under normal conditions,

manager=’smith’. The attacker can easily change the request parameter value as

(manager=’smith’)(|objectclass=\*)

It will return all the employees along with manager. To prevent this the user input which used as a LDAP search filter must be validated against list of valid values in application layer before sending it to the LDAP Server. This is called positive validation scheme.

However there is some situation special characters are required for the LDAP query .In this scenario the special characters are used with back slash character so that LDAP interpreter can assume this is part of LDAP query.

**LDAP Configuration :**

Implementing the tight access control on the LDAP may reduce the LDAP injection vulnerabilities . Restrict the LDAP access level used by the web application is also an prevention .

LDAP server should not be directly accessible on the internet.

1. **What is network threats ?**

There are 4 types of network threats,

1. Unstructured - virus,worm,Trojan horse
2. Structured - Identity theft , credit card information theft
3. Internal : created by the ex-employee of the organization
4. External: created by the person outside of the organization.
5. **When do we choose SOA Architecture ?**

**SOA –**Service Oriented Architecture.

It is an architecture used to integrate the different components exposed as service to build an application in a distributed environment. It also connects the systems in heterogeneous environment. It is also used for the reuse of the legacy application. It is also used when we need to abstract the implementation details to customer. Though we have the traditional distributed messaging techniques such as RMI and CORBA, it cannot connect the heterogeneous systems that are separated by firewalls. This is because firewalls allows only HTTP and SMTP traffic. This prevent the enterprise applications, to connect the distributed systems available over the LAN. The SOA architecture address this problem as exposing the systems as services instead of plain objects. The SOAP or Other XML messages are designed to communicate these systems.

1. **Why do we need web service security (WS-Security)?**

The component are exposed as service in SOA architecture and these services are available on over the internet to consume. The XML messages designed for the SOA Architecture to communicate with these services are firewall transparent and these messages can bypass the firewalls over HTTP and SMTP protocols. These kind of message transparency lead to security issues. So we need to use the WS-Security protocol to protect the messages from end to end(sender to receiver) in SOA system. But this WS-Security standards ensures the security between two trusted parties.

1. **What are the security vulnerabilities arising in Web applications?**

There are so many scenarios , the security vulnerability issues may happen in the web applications.

**Scnerio 1: Input Validation attacks**

If the input validation not happened in the web application , there are so many possible causes which attacks the web application.

1. **Buffer Overflow attack :**

By inserting the larger value than expected into program variable may execute some arbitrary or malicious code with the privilege of root user. Each process has its own memory , so in this scenario the memory allocated for the process is smaller than the data written to the memory. So the extra data goes some where undesirable.

**Possible Cause :**

Size of the input is not validated before written to memory.

**Solution:**

Input size validation should be done for the program variable.

1. **Cross Site Scripting : ( XSS)**

Client side script injection. Injecting the client side script to the web pages. The attackers explore the web application and find the page which doesn’t filter out the input for the authenticated users. They will simply insert the malicious code or script as part of the request which will be stored in the database , visitors log, comment field or message forum. Later when the actual user try to access the page , this malicious code or script send as part of the output , which causes the client browser. The client browser doesn’t know that these are sent from untrusted sources. This kind of script can easily access the users password, credit card details, session tokens and other sensitive informations from the web application. This way the attackers attack the web application. Most of the XSS attacks happens by using the <Script> tags in the request.

**Types:**

1. **Stored XSS :** The user input is stored in the data base ,message forum, visitors log and send to client browser as part of the output
2. **Reflected XSS :** The user input is immediately returned to the client as part the the response.

**Possible Cause:**

Input and output are not validated and not encoded properly.

**Solution:**

Input from user and output send to browser needs to be validated and encoded properly. <Script> tags needs to be filtered out in the input and output.

1. **SQL Injection:**

Explained earlier.

**Possible Cause:**

Unvalidated User Input

**Solution:**

User input needs to be validated. Special characters needs to be filtered out in appropriate places. Single and double quotes needs to be removed from input.

1. **IP-spoofing:**

Computers network communicate each other by exchanging the information in form of data packets. These packets contain various header information. One such header is “Source IP Address”. IP-Spoofing is also known as IP address forgery or host file hijack. This is the hijacking technique used by the attacker who create the fake IP packets and sent it to the clients. Once the attacker gains access to the network they can modify the data, reroute or delete the data. Sometimes they can steel the sensitive information from client system or they can install malicious code into the client systems.

**Possible Cause:**

Using the direct IP address authentication .

**Solution :**

Don’t use direct IP address based authentication.

Use some access control list in downstream interface to deny private IP address.

Filter the data packets on the router which comes from outside the network

Use public key based authentication.

Enable encryption based session , so that trusted sites outside of the network can communicate with actual hosts.

1. **Network eavesdropping:**

Eavesdrop :- secretly listens to a conversation.

This vulnerability occurs when the attackers gains access to the data path to the network. This can be done by IP- Spoofing or by Man-in the-middle attacks.They can easily capture the user name and password which send from client to server in form of plain text. By using these details they can easily capture the sensitive data by eavesdropping the connection. The victim(client) doesn’t know that he is monitored by the attacker or the presence of the attacker in the middle.

1. **Dictionary Attack:**

The attackers goal is to obtain the password. So they systematically test all the password until they found the correct one.

**Possible Cause:**

Passwords are sent as part of request which is not encrypted

**Solution :**

Use proper encryption method for the inputs and outputs.

1. **Data Tampering:**

**Tampering :-** interfere with something to cause damage or perform unauthorized alteration.

This is occurs when the attackers gain access to the data passes over the network . The attacker can manipulate the data sent from client to server. As a result the tampered data sent to the origination.

**Possible Cause:**

IP-Spoofing

**Solution:**

Prevent the ip-spoofing

**Scenerio 2: Denial Of Service attack**

The main goal of the attackers is to crash the web application by stealing secret information. With this attack they can easily crash the providers computer or network. They can even crash the router , firewall or proxy servers.

**Possible Cause:**

IP-Spoofing

**Solution:**

Prevent the ip-spoofing

**Scenerio 3: Session Management attacks**

Weak session management results in session management attacks in web application. This attacks occurs when the attackers gains access to the session ID such as cookie,url parameter , session id.

1. **Session Replay:**

This is the hijacking technique used by the attacker to perform the session replay. They steel the session id’s which is sent from client to server and replay the same request by bypassing the authentication. So that they get the system access.

1. **Man-in-the-Middle:**

This occurs when the attacker intercepts the message between client and server. They can easily modify the data and send to sender and receiver . The sender and receiver doesn’t aware of the Main in the middle.

**Scenerio 4: Parameter Tampering**

The goal is to attack he form parameters when it is send from client to server. The attacker can manipulate the header parameter and mode parameter. The parameter manipulation has the following types,

1. **Http header manipulation:**

The header always has the control information. Http request header is sent from client and Http Response header is send from server. Web browsers usually doesn’t allow to modify the headers. But the attackers can write their own program to manipulate the headers.

1. **Form Field manipulation:**

The attacker can change the form field values such as hidden filed, free form field ,check box, option box by page- . view source->save -> modify the content -> reload the page.

1. **QueryString manipulation**

When the client use Http Get method to send the request , the request parameters are appended with the url and displayed I the browser. The attacker can easily change this data.

1. **What are the security vulnerability arise in web services? How to prevent this?**

Apart from the above web application vulnerability, there are some more applicable for the XML based web services.

Because of the open standard and modularity nature of the web services still lead to security problems. SSL itself is not sufficient for the web service security, since SSL is not designed for the internet based web services. SOAP also doesn’t have its own security mechanism. Neither firewalls nor PKI(Public Key Infrastructure ) based security mechanism helpful for web service security. Because firewalls allows HTTP traffic via web service request. PKI security mechanisms is also not sufficient for the chains of applications connected by web services. WSDL also show the data elements very clear Attackers always interested on the XML message for their attack.

**XPath Injection:**

This is also an vulnerability arise in XML based web application. Similar to SQL injection. When we use the user input directly to the XPath queries , all the data can be fetched from the Native XML document in which application uses XML as data store.

Application stores employee details in XML store,

<?xml version="1.0" encoding="utf-8"?>

<employees>

<employee id="AS789" firstname="John" lastname="Doo" annualsalary="70000"/>

<employee id="AS719" firstname="Isabela" lastname="Dobora" annualsalary="90000"/>

<employee id="AS219" firstname="Eric" lastname="Lambert" annualsalary="65000"/>

</employees>

XPath Injection happens like this,

Document doc = builder.parse(new InputSource(new StringReader(DATASOURCE\_XML)));

// Retrieve employee ID from the input HTTP request

String eID = request.getParameter("employeeID");

if (eID == null) {

eID = "";

}

// Create XPATH expression

String xpathExpr = "/employees/employee[@id='" + eID + "']";

XPath expression = new DOMXPath(xpathExpr);

The attacker can change the request input and can fetch the irrelevant data from xml store.

By using the this XPath queries they can easily extract the complete XML document. XPath is also useful to identify the parts of the XML to which the XML signatures applies.

**SOAP Security:**

SOAP doesn’t define any security mechanism for the web services. SOAP message can easily pass through the firewalls using the HTTP protocol. SOAP messages are represented as SOAPEnvelope , and it has two element as SOAPHeader and SOAPBody. SOAP Header is an optional element and it contains the security and other soap message level details. It can also contains some details about SOAP server. The attacker use this server details and they guess the endpoint details and they can route the messages to some other destination by changing the header details. Head is an optional element. SOAP message can have one or more header element. Security element in header section specify the target of the role. If no security element present , it can be consumed by anyone. The attacker can insert extra header or they can delete the header or they can delete the body of the soap message.

To prevent this , SOAP messages needs to be encrypted and digital signature to be used.

**XML Security:**

XML encryption and XML digital signatures are two methods used to secure the XML messages. There are different algorithms are used in these techniques. Only using these techniques doesn’t make the web services secured. The other security attacks and solutions are described below,

**XML digital signatures:-** provides authentication, data integrity and non reputation. Encrypt the whole document using the digital certificate

**XML Encryption:-** provides secure XML document. It encrypt only part of the XML document.

**Classification on XML Web Service Security attacks and Solution :**

**Refer the table 1 from WebServiceSecurityPartII.pdf**

1. **How the webservice is digitally signed ?**

**Refer :** [**http://www.drdobbs.com/database/digitally-signing-and-verifying-messages/209400693?pgno=2**](http://www.drdobbs.com/database/digitally-signing-and-verifying-messages/209400693?pgno=2)

1. **What is scalability of the application ?**

Scalability is the capacity of the system or application.

1. **What is hibernate interceptor ?**

It is an hibernate feature which allows application to intercept on some hibernate events to add additional functionality or implement some generic functionality. Each method on the hibernate session has associated events. The interceptor interface provides call back methods from session to application. The interceptor are created by implementing the Interceptor Interface or extending the EmptyInterceptor class. The interceptor feature is very useful to implement the Audit log functionality.

There are two types of interceptor , (interceptors are enabled by using any of these ways)

1. Session Factory Interceptor – created by setting the interceptor instance as into Configuration object.

New Configuration().setInterceptor(new Interceptor());

1. Session Interceptor – created by passing the interceptor instance into the openSession().

Session=sessionFactory.openSession(new Interceptor());

**Methods available in Interceptor:**

onSave() – called when the object is saved, the object is not saved into database yet.

onFlushDirty() –called when the object is updated, the object is not updated into database yet

preFlush() – called before saved, updated or deleted object are committed into database

postFlush()- called after saved , updated or deleted objects are committed into database

onDelete() – called when object is deleted, objects are not delete from database yet.

**Implementation Details:**

http://www.mkyong.com/hibernate/hibernate-interceptor-example-audit-log/

1. **What are the Database locks are available?**
2. **What are the advantages of maven ?**
3. **Explain the below**

**Java.Data.LocalTime , LocalDate, LocalDateTime , LocalDateTimeFormatter , Period**

1. **Explain list.removeif() in java 1.8 ?**

In java1.8, there is a new default method introduced in Collection interface. Which will remove the elements from the list based on the predicate condition given.

Ex:

List<Employee> empList=new Arraylist<Employee>();

Emplist.add(new Employee(“Vasantha”,20)

empList.add(new Employee(“sumathi”,35)

// remove an employee who has age more than 30

empList.removeIf((Employee emp)->emp.age>30)

now the will have only employee details of “Vasantha”.

This is the replacement method for the Iterator.remove().

By using the iterator , we have to iterate over the list and check the condition then remove it. It requires more code but the removeIf() do the same functions in single line using the Lambda expression as predicate.

1. **How it will behave ?**

**ArrayList a=new ArrayList();**

**a.add(“fg”)**

**a.add(“Sd”);**

**String strArry[]=new String[3];**

**String str2[]=a.toArray(strArry); -How it will behave ?**

**What will the value in strArry[] and str2[] ?**

**Ans:**

Both strArry and str will have the value of list.

**a.toArray(strArry) :-**  this line convert the list to an array and poy the values to strArray

**String str2[]=a.toArray(strArry); :-** the converted array also assigned to str2 .

So both of the array will have the same values.

1. **What is the difference between toArray() and toArray(T[] t) in ArrayList?**

The main difference is with their signature.

Public Object[] toArray() :- It returns an Object array . it convert the list to array with all element from the list.

Public T[] toArray(T[] t) :- it returns the specified type of array which is passed as an argument. It converts the list to array with all the element from the list . It also copy the returned array to its argument variable of specified array type.