5marks Questions and Answers

1.Write in detail about the use of send Redirect

method.

sendRedirect() method redirects the response to another resource, inside or outside the server. It makes the client/browser to create a new request to get to the

resource. It sends a temporary redirect response to the client using the specified redirect location URL.

Syntax:

void sendRedirect(java.lang.String location) throws java.io.IOException

sendRedirect() accepts the respective URL to which the request is to be redirected.

Can redirect the request to another resource like Servlet, HTML page, or JSP page that are inside or outside the server.

It works on the HTTP response object and always sends a new request for the object.

A new URL that is being redirected can be seen in the browser.

Example:

As we discussed above, we will create a simple servlet that will redirect the page to another website.

Index.html:-

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Home</title>

</head>

<body>

<form action="redirect" method="get">

<h2>sendRedirect() method in Java Servlets</h2>

sendRedirect() method, redirects the client request from the one servlet to another servlet.
 It creates a new request from

the client browser for the resource.

For More information <input type="submit" value="Click Here">

</form>

</body>

</html>

We have a form with action ="redirect" method = "get", so Index.html maps the servlet having '/redirect' URL and executes the 'doGet' method in it.

ServletRedirect.java:-

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/redirect")

public class ServletRedirect extends HttpServlet {

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

resp.sendRedirect("https://www.geeksforgeeks.org/url-rewriting-using-java-servlet/amp/");

}

} We need to import all the required packages from javax.servlet. Instead of mapping the page with servlet in web.xml, we are specifying it using annotation – @WebServlet("/redirect"). When the 'doGet' method executes, the response will redirect to the specified URL

2.Discuss on the steps used in creating a new Bean?

Java bean: It is a development component for performing a task in an application. Any ordinary java class with a set of variables and methods can be called as javabean. The methods present in a java bean must follow a naming convention. A java bean can be used in a web application to store the state of the users.

Creation process of a java bean:-

Create a class in a package as the bean class.

Provide the required variables as the properties.

Provide setter and getter methods for each of the variables.

Store the package folder inside a classes folder.

Compile the file as ordinary java file.

Example:

```
package pack1;
public class mybean
{
 private String name,pass;
 public void setName(String n)
  {
    name=n;
  }
   public void setPass(String p)
  {
    pass=p;
  }
 public String getName()
  {
  return name;
   }
  public String getPass()
  {
   return pass;
   }
}
```

3. What are the features of EJB technology?

Enterprise Java Beans (EJB) is one of the several Java APIs for standard manufacture of enterprise software. EJB is a server-side software element that summarizes business logic of

an application. Enterprise Java Beans web repository yields a runtime domain for web related software elements including computer reliability, Java Servlet Lifecycle (JSL) management, transaction procedure and other web services. The EJB enumeration is a subset of the Java EE enumeration.

To run EJB application we need an application server (EJB Container) such as Jboss, Glassfish, Weblogic, Websphere etc. It performs:

1. Life cycle management

2. Security

3. Transaction management

4. Object pooling

Types of Enterprise Java Beans:-

There are three types of EJB:

1. Session Bean: Session bean contains business logic that can be invoked by local, remote or webservice client. There are two types of session beans: (i) Stateful session bean and (ii) Stateless session bean.

(i) Stateful Session bean :

Stateful session bean performs business task with the help of a state. Stateful session bean can be used to access various method calls by storing the information in an instance variable. Some of the applications require information to be stored across separate method calls. In a shopping site, the items chosen by a customer must be stored as data is an example of stateful session bean.

(ii) Stateless Session bean :

Stateless session bean implement business logic without having a persistent storage mechanism, such as a state or database and can used shared data. Stateless session bean can be used in situations where information is not required to used across call methods.

2. Message Driven Bean: Like Session Bean, it contains the business logic but it is invoked by passing message.

3. Entity Bean: It summarizes the state that can be remained in the database. It is deprecated. Now, it is replaced with JPA (Java Persistent API). There are two types of entity bean:

(i) Bean Managed Persistence :

In a bean managed persistence type of entity bean, the programmer has to write the code for database calls. It persists across multiple sessions and multiple clients.

(ii) Container Managed Persistence :

Container managed persistence are enterprise bean that persists across database. In container managed persistence the container take care of database calls.

use Enterprise Java Beans:-

1.Application needs Remote Access. In other words, it is distributed.

2.Application needs to be scalable. EJB applications supports load balancing, clustering and fail-over.

3.Application needs encapsulated business logic. EJB application is differentiated from demonstration and persistent layer.

4. Advantages of Java Beans?

A Bean obtains all the benefits of Java's "write-once, run-anywhere" paradigm.

The properties, events, and methods of a Bean that are exposed to an application

builder tool can be controlled.

A Bean may be designed to operate correctly in different locales, which makes it

useful in global markets.

Auxiliary software can be provided to help a person configure a Bean. This software is

only needed when the design-time parameters for that component are being set. It

does not need to be included in the run-time environment.

The configuration settings of a Bean can be saved in persistent storage and restored

at a later time.

A Bean may register to receive events from other objects and can generate events that

are sent to other objects.

4. Write short notes on lifecycle methods of a servlet?

The servlet is initialized by calling the init() method.

The servlet calls service() method to process a client's request.

The servlet is terminated by calling the destroy() method.

Finally, servlet is garbage collected by the garbage collector of the JVM.

Now let us discuss the life cycle methods in detail.

The init() Method:-

The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards. So, it is used for one-time initializations, just as with the init method of applets.

The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started.

When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate. The init() method simply creates or loads some data that will be used throughout the life of the servlet.

The init method definition looks like this

```
*public void init() throws ServletException {
  // Initialization code...
}*
```

The service() Method:-

The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.

Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

Here is the signature of this method –

```
*public void service(ServletRequest request, ServletResponse response)
  throws ServletException, IOException {
}*
```

The service () method is called by the container and service method invokes doGet, doPost, doPut, doDelete, etc. methods as appropriate. So you have nothing to do with service() method but you override either doGet() or doPost() depending on what type of request you receive from the client.

The doGet() and doPost() are most frequently used methods with in each service request. Here is the signature of these two methods.

The doGet() Method:-

A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

```
*public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
  // Servlet code
}*
```

The destroy() Method:-

The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

After the destroy() method is called, the servlet object is marked for garbage collection. The destroy method definition looks like this –

```
*public void destroy() {
  // Finalization code...
}*
```

5. Explain the use of logical operators in Perl?

These operators are used to combine two or more conditions/constraints or to complement the evaluation of the original condition in consideration.

Logical AND: The 'and' operator returns true when both the conditions in consideration are satisfied. For example, $a and $b is true when both a and b both are true (i.e. non-zero). You can use also &&.

Logical OR: The 'or' operator returns true when one (or both) of the conditions in consideration is satisfied. For example, $a or $b is true if one of a or b is true (i.e. non-zero). Of course, it gives result "true" when both a and b are true. You can use also ||

Logical NOT: The 'not' operator will give 1 if the condition in consideration is satisfied. For example, not($d) is true if $d is 0.

Program: To demonstrate the working of Logical Operators:

```
# Perl Program to illustrate the Logical Operators
 # Operands
$a = true;
$b = false;
 # AND operator
$result = $a && $b;
print "AND Operator: ", $result,"\n";
# OR operator
$result = $a || $b;
print "OR Operator: ", $result,"\n";
# NOT operator
$d = 0;
$result = not($d);
print "NOT Operator: ", $result;
```

Output:

AND Operator: false

OR Operator: true

NOT Operator: 1

7. Explain about JSP elements?

In this section we will learn about the JSP elements that we can use on a JSP page. This section will describe you all types of JSP elements that we can use on JSP page.

JSP elements are called the JSP tags on JSP page. On the JSP page mainly three groups of JSP elements are used :

1.JSP Scripting Elements

2.JSP Directive Elements

3.JSP Standard Action Elements

JSP Scripting Elements:-

JSP Scripting Elements are used for writing the Java Code inside the JSP page. There are different types of scripting elements these elements are used for various purposes. Following are the scripting elements :

JSP Scriptlet element/tag : A scriptlet tag is denoted by the special characters <% %>. The special characters '<%' specifies the starting of the scriptlet tag and '%>' specifies end of scriptlet. On the JSP page Java Code is written inside it.

JSP Declaration element/tag : The declaration tag is denoted by the special characters <%! %>. The special character '<%!' specifies the start of the declaration and '%>' specifies the end of the declaration. Declaration element is used to declare the fields, methods to use inside the JSP page. These declaration element helps developer to declare fields, methods like in Servlet how they can declare.

JSP Expression element/tag : Then expression tag is denoted by the special characters <%= %>. The special character '<%=' specifies the starting of the expression and '%>' specifies the end of the expression. The expression tag is used to set the output. The expression is used like the 'out' implicit object in JSP.

JSP Directive Elements:-

Directive elements in JSP provides the special information to the JSP engine. It gives the information about the JSP page. Each JSP page goes through the two phases i.e. translation phase and request time phase. At the translation phase JSP page is translated into a Servlet. The JSP engine translates the JSP page into a Servlet, directive elements are handled at the translation time. These directives are translated to the Servlet only for once, until there is no changes made to the directive elements in the Servlet.

JSP Directive Elements can be declared within the following special characters '<%@ %>'. Syntax for using directive in the JSP page is as follows :

<%@ directiveName attr1="value1" attr2="

JSP Standard Action Elements:-

JSP Action Elements are used to take action on the basis of some information. These action elements can create, modify or use the other objects. Action elements are coded with the strict syntax they are used to apply the built-in functionality. Action elements are represented as <jsp:tagName> </jsp:tagName>. Code is written inside these tags, the <jsp:tagName> specifies the starting of the action tag where as the action tag is ends with </jsp:tagName>. When you are not required to defined the body of the tag you may end up the action tag by <jsp:tagName/>

Miscellaneous:-

JSP Comment Elements:-

JSP comment elements are used to hide your code from the "view page source". However, you can use HTML comment tags in JSP page but, when user of your website will choose the "view page source" then they can see your commented code. The JSP comment is denoted by the special character <%-- --%>. The special character '<%--' specifies the opening comment element and the special character '--%>' specifies the end of the comment tag.

8.Life Cycle of an Entity Bean?

The life cycle of an entity bean is controlled by the EJB container, not by your application. However, you may find it helpful to learn about the life cycle when deciding in which method your entity bean will connect to a database. See Resource Connections for more information.

the stages that an entity bean passes through during its lifetime. After the EJB container creates the instance, it calls the setEntityContext method of the entity bean class. The setEntityContext method passes the entity context to the bean.

After instantiation, the entity bean moves to a pool of available instances. While in the pooled stage, the instance is not associated with any particular EJB object identity. All instances in the pool are identical. The EJB container assigns an identity to an instance when moving it to the ready stage.

There are two paths from the pooled stage to the ready stage. On the first path, the client invokes the create method, causing the EJB container to call the ejbCreate and ejbPostCreate methods. On the second path, the EJB container invokes the ejbActivate method. While in the ready stage, an entity bean's business methods may be invoked.

There are also two paths from the ready stage to the pooled stage. First, a client may invoke the remove method, which causes the EJB container to call the ejbRemove method. Second, the EJB container may invoke the ejbPassivate method.

At the end of the life cycle, the EJB container removes the instance from the pool and invokes the unsetEntityContext method.

In the pooled state, an instance is not associated with any particular EJB object identity. With bean-managed persistence, when the EJB container moves an instance from the pooled state to the ready state, it does not automatically set the primary key. Therefore, the ejbCreate and ejbActivate methods must set the primary key. If the primary key is incorrect, the ejbLoad and ejbStore methods cannot synchronize the instance variables with the database. In the AccountEJB example, the ejbCreate method assigns the primary key from one of the input parameters. The ejbActivate method sets the primary key (id) as follows:

id = (String)context.getPrimaryKey();

In the pooled state, the values of the instance variables are not needed. You can make these instance variables eligible for garbage collection by setting them to null in the ejbPasssivate method.

9. Explain the features of facelets?

It is a light weight page declaration language which is used to build JavaServer Faces views using HTML style.

It includes the following features:

It uses XHTML for creating web pages.

It supports Facelets tag libraries in addition to JavaServer Faces and JSTL tag libraries.

It supports the Expression Language (EL).

It uses templating for components and pages.