Practical No 6

Aim: Define a RESTful web service that accepts the details to be stored in a database and performs CRUD operation

Theory:

RESTful Web Services are basically REST Architecture based Web Services. In REST Architecture everything is a resource. RESTful web services are light weight, highly scalable and maintainable and are very commonly used to create APIs for web-based applications.

Entity Class:

An entity is a collection of fields and associated database operations. Entity classes have a stereotype of entity. An entity class is essentially an object wrapper for a database table. The attributes of an entity are transformed to columns on the database table.

Java Server Faces (JSF) :

It is a Java-based web application framework intended to simplify development integration of web-based user interfaces. JavaServer Faces is a standardized display technology, which was formalized in a specification through the Java Community Process.

CRUD is an acronym for CREATE, READ, UPDATE and DELETE which are basic functions of persistent storage. CRUD operations can use forms or an interface view to retrieve and return data from a database

Difference between SOAP and REST

No.	SOAP	REST
1)	SOAP is a **protocol**.	REST is an **architectural style**.
2)	SOAP stands for **Simple Object Access Protocol**.	REST stands for **REpresentational State Transfer**.
3)	SOAP **can't use REST** because it is a protocol.	REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP.
4)	SOAP **uses services interfaces to expose the business logic**.	REST **uses URI to expose business logic**.
5)	**JAX-WS** is the java API for SOAP web services.	**JAX-RS** is the java API for RESTful web services.
6)	SOAP **defines standards** to be strictly followed.	REST does not define too much standards like SOAP.
7)	SOAP **requires more bandwidth** and resource than REST.	REST **requires less bandwidth** and resource than SOAP.

8)	SOAP **defines its own security**.	RESTful web services **inherits security measures** from the underlying transport.
9)	SOAP **permits XML** data format only.	REST **permits different** data format such as Plain text, HTML, XML, JSON etc.
10)	SOAP is **less preferred** than REST.	REST **more preferred** than SOAP.

Steps:

1. Click on Window menu and click on **Projects, Files** & **Services** to open it.
2. **Right click on Java DB** and then **click on Start Server** to start the server .
3. Now expand Java DB and **right click on sample** and then **click on connect** to connect the sample database with server.
4. Now create a web application with the name **CRUD_Operation**.
5. Create an entity class. **Right click on project name -> New -> Entity Class**.
6. A window will appear like bellow pic. Enter following data and click on Next ….
   Class Name -
   > seller Package
   Name -> com.kk
7. **Click on Finish**.
8. Right click on project name and create JSF Pages from Entity Classes.
   Right click on project name -> New -> JSF Pages from Entity Classes
9. **Select com.kk.seller** and **click on Add button and then Next button** on below.
10. A window like below will appear on the screen. **Enter the data into that window as entered in below pic** and **click on Next button**.
11. Now **click on Finish**.
12. Right click on project name and create RESTful Web Services from Entity Classes.
    Right click on project name -> New -> RESTful Web Services from Entity Classes
13. **Repeat step 9** and then it will go on next page. Then **enter the com.kk.service**
    in Resource Package and then **click on Finish button**

14. Now **open seller.java file under com.kk package**.

15. Now **right click on web application name and Deploy** it.

16. Now **right click on project name** and **run** it.

Code:Seller.java

```
package com.kk;

import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.xml.bind.annotation.XmlRootElement;

@Entity
@XmlRootElement
public class seller implements Serializable {
   private static final long serialVersionUID = 1L;
   @Id
   @GeneratedValue(strategy = GenerationType.AUTO)
   private Long id;

   public Long getId() {
      return id;
   }

   public void setId(Long id) {
      this.id = id;
   }

   @Override
   public int hashCode() {
      int hash = 0;
      hash += (id != null ? id.hashCode() : 0);
      return hash;
   }

   @Override
   public boolean equals(Object object) {
      // TODO: Warning - this method won't work in the case the id fields are not set
```

```
    if (!(object instanceof seller)) {
        return false;
    }
    seller other = (seller) object;
    if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {
        return false;
    }
    return true;
}

@Override
public String toString() {
    return "com.kk.seller[ id=" + id + " ]";
}

}
```

Output: