**Hibernate**

**HIbetnate by XMl:**

1. Create the Persistent class

**Student.java:**

**package** Mypack1;

**public** **class** Student {

**private** **int** stdId;

**private** String firstName;

**private** String lastName;

**public** **int** getStdId() {

**return** stdId;

}

**public** **void** setStdId(**int** stdId) {

**this**.stdId = stdId;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

}

2) Create the mapping file for Persistent class

Student.hbm.xml:

<?xml version=*'1.0'* encoding=*'UTF-8'*?>

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 5.3//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd">

<hibernate-mapping>

<class name=*"Mypack1.Student"* table=*"studentd"*>

<id name=*"stdId"* type=*"integer"* column=*"stid"*>

<generator class=*"assigned"*></generator>

</id>

<property name=*"firstName"* column=*"stname"*></property>

<property name=*"lastName"* column=*"stlastname"*></property>

</class>

</hibernate-mapping>

</hibernate-mapping>

3) Create the Configuration file

hibernate.cfg.xml:

<?xml version=*'1.0'* encoding=*'UTF-8'*?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 5.3//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">

<hibernate-configuration>

<session-factory>

<property name=*"hbm2ddl.auto"*>update</property>

<property name=*"dialect"*>org.hibernate.dialect.Oracle9Dialect</property>

<property name=*"connection.url"*>jdbc:oracle:thin:@localhost:1521:xe</property>

<property name=*"connection.username"*>system</property>

<property name=*"connection.password"*>system</property>

<property name=*"connection.driver\_class"*>oracle.jdbc.driver.OracleDriver</property>

<property name=*"hibernate.show\_sql"*>true</property>

<mapping resource=*"Student.hbm.xml"*/>

</session-factory>

</hibernate-configuration>

4) Create the class that retrieves or stores the object

Insert

**package** Mypack1;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**public** **class** StoreData {

**public** **static** **void** main( String[] args )

{

StandardServiceRegistry ssr = **new** StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta = **new** MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory = meta.getSessionFactoryBuilder().build();

Session session = factory.openSession();

Transaction t = session.beginTransaction();

Student e1=**new** Student();

e1.setStdId(1);

e1.setFirstName("Tim");

e1.setLastName("Cook");

session.save(e1);

t.commit();

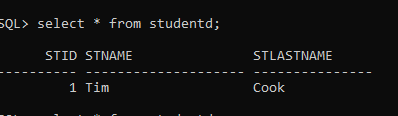
System.out.println("successfully saved details");

factory.close();

session.close();

}

}



Update

**package** Mypack1;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**public** **class** StoreData {

**public** **static** **void** main( String[] args )

{

StandardServiceRegistry ssr = **new** StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta = **new** MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory = meta.getSessionFactoryBuilder().build();

Session session = factory.openSession();

Transaction t = session.beginTransaction();

Student e1=**new** Student();

e1.setStdId(1);

e1.setFirstName("Ajay");

e1.setLastName("Kumar");

session.update(e1);

t.commit()

// printing the updated value in console

Student e2=session.get(Student.class,1);

System.out.println(e2.getFirstName());

System.out.println(e2.getLastName());

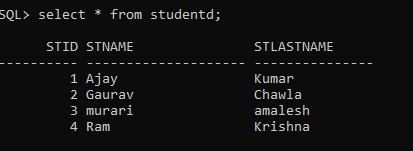
System.out.println("successfully saved details");

factory.close();

session.close();

}

}



Delete

**package** Mypack1;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**public** **class** StoreData {

**public** **static** **void** main( String[] args )

{

StandardServiceRegistry ssr = **new** StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta = **new** MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory = meta.getSessionFactoryBuilder().build();

Session session = factory.openSession();

Transaction t = session.beginTransaction();

Student e1=**new** Student();

e1.setStdId(4);

session.delete(e1);

t.commit();

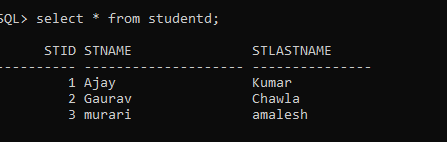
System.out.println("successfully saved details");

factory.close();

session.close();

}

}



Select

**package** Mypack1;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.Scanner;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**import** org.hibernate.query.Query;

**public** **class** Select {

**public** **static** **void** main( String[] args )

{

StandardServiceRegistry ssr = **new** StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta = **new** MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory = meta.getSessionFactoryBuilder().build();

Session session = factory.openSession();

Transaction t = session.beginTransaction();

Scanner sc=**new** Scanner(System.***in***);

**int** id=sc.nextInt();

Query q=session.createQuery("from Student where id=:Idno");

q.setParameter("Idno", id);

List l=q.list();

Iterator it=l.iterator();

**while**(it.hasNext()){

Student m=(Student)it.next();

System.***out***.println("Name : "+m.getFirstName());

System.***out***.println("Name : "+m.getLastName());

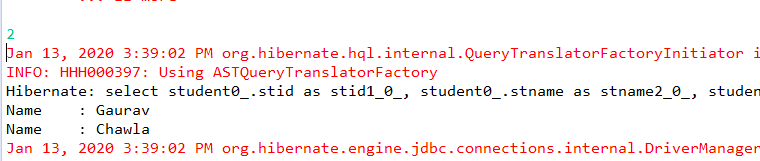
factory.close();

session.close();

}

}

}

****

**Select All**

**package** Mypack1;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.Scanner;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**import** org.hibernate.query.Query;

**public** **class** Select {

**public** **static** **void** main(String[] args) {

StandardServiceRegistry ssr = **new** StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta = **new** MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory = meta.getSessionFactoryBuilder().build();

Session session = factory.openSession();

Transaction t = session.beginTransaction();

Query q = session.createQuery("from Student");

// q.setParameter("Idno", id);

List l = q.list();

Iterator it = l.iterator();

**while** (it.hasNext()) {

Student m = (Student) it.next();

System.***out***.println("Name : " + m.getFirstName());

System.***out***.println("LName : " + m.getLastName());

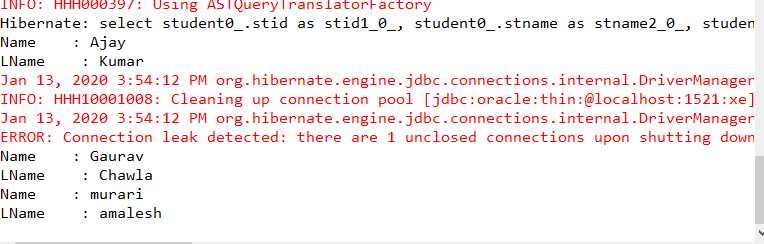
factory.close();

session.close();

}

}

}

****

**Hibernate by Annotations:**

Create the Persistent class

**Student.java:**

**package** Mypack1;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name="studentd")

**public** **class** Student {

@Id

@Column(name="stid")

**private** **int** stdId;

@Column(name="stname")

**private** String firstName;

@Column(name="stlastname")

**private** String lastName;

**public** **int** getStdId() {

**return** stdId;

}

**public** **void** setStdId(**int** stdId) {

**this**.stdId = stdId;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() {

**return** lastName;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

}

**Configration file:**

hibernate.cfg.xml:

<?xml version=*'1.0'* encoding=*'UTF-8'*?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 5.3//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">

<hibernate-configuration>

<session-factory>

<property name=*"hbm2ddl.auto"*>update</property>

<property name=*"dialect"*>org.hibernate.dialect.Oracle9Dialect</property>

<property name=*"connection.url"*>jdbc:oracle:thin:@localhost:1521:xe</property>

<property name=*"connection.username"*>system</property>

<property name=*"connection.password"*>system</property>

<property name=*"connection.driver\_class"*>oracle.jdbc.driver.OracleDriver</property>

<property name=*"hibernate.show\_sql"*>true</property>

<mapping class=*"Mypack1.Student"*/>

</session-factory>

</hibernate-configuration>

Create the class that retrieves or stores the object

package Mypack1;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

public class StoreData {

public static void main( String[] args )

{

StandardServiceRegistry ssr = new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory = meta.getSessionFactoryBuilder().build();

Session session = factory.openSession();

Transaction t = session.beginTransaction();

Student e1=new Student();

e1.setStdId(4);

e1.setFirstName("Ram");

e1.setLastName("Krishna");

// e1.setStdId(1);

// e1.setFirstName("Tim");

// e1.setLastName("Cook");

//

session.save(e1);

// session.update(e1);

// session.delete(e1);

t.commit();

// Student e2=session.get(Student.class,1);

// System.out.println(e2.getFirstName());

// System.out.println(e2.getLastName());

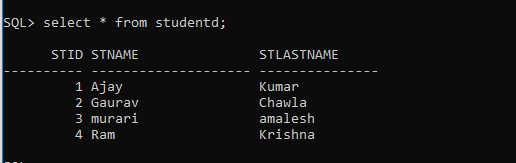
// System.out.println("successfully saved details");

factory.close();

session.close();

}

}



ONE TO ONE ANNOTATION:

### 1) Persistent classes for one to one mapping

There are two persistent classes Employee.java and Address.java. Employee class contains Address class reference and vice versa.

Employee.java

package Mypack2;

import javax.persistence.CascadeType;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToOne;

import javax.persistence.PrimaryKeyJoinColumn;

import javax.persistence.Table;

@Entity

@Table(name="emp220")

public class Employee {

@Id

@GeneratedValue(strategy=GenerationType.AUTO)

@PrimaryKeyJoinColumn

private int employeeId;

private String name,email;

@OneToOne(targetEntity=Address.class,cascade=CascadeType.ALL)

private Address address;

public int getEmployeeId() {

return employeeId;

}

public void setEmployeeId(int employeeId) {

this.employeeId = employeeId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public Address getAddress() {

return address;

}

public void setAddress(Address address) {

this.address = address;

}

}

Address.java

**package** Mypack2;

**import** javax.persistence.\*;

@Entity

@Table(name="address220")

**public** **class** Address {

@Id

@GeneratedValue(strategy=GenerationType.***AUTO***)

**private** **int** addressId;

**private** String addressLine1,city,state,country;

**private** **int** pincode;

@OneToOne(targetEntity=Employee.**class**)

**private** Employee employee;

**public** **int** getAddressId() {

**return** addressId;

}

**public** **void** setAddressId(**int** addressId) {

**this**.addressId = addressId;

}

**public** String getAddressLine1() {

**return** addressLine1;

}

**public** **void** setAddressLine1(String addressLine1) {

**this**.addressLine1 = addressLine1;

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

}

**public** String getState() {

**return** state;

}

**public** **void** setState(String state) {

**this**.state = state;

}

**public** String getCountry() {

**return** country;

}

**public** **void** setCountry(String country) {

**this**.country = country;

}

**public** **int** getPincode() {

**return** pincode;

}

**public** **void** setPincode(**int** pincode) {

**this**.pincode = pincode;

}

**public** Employee getEmployee() {

**return** employee;

}

**public** **void** setEmployee(Employee employee) {

**this**.employee = employee;

}

}

### 3) Configuration file

This file contains information about the database and mapping file.

**hibernate.cfg.xml**

<?xml version=*'1.0'* encoding=*'UTF-8'*?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 5.3//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">

<hibernate-configuration>

<session-factory>

<property name=*"hbm2ddl.auto"*>update</property>

<property name=*"dialect"*>org.hibernate.dialect.Oracle9Dialect</property>

<property name=*"connection.url"*>jdbc:oracle:thin:@localhost:1521:xe</property>

<property name=*"connection.username"*>system</property>

<property name=*"connection.password"*>system</property>

<property name=*"connection.driver\_class"*>oracle.jdbc.driver.OracleDriver</property>

<mapping class=*"Mypack2.Address"*/>

<mapping class=*"Mypack2.Employee"*/>

</session-factory>

</hibernate-configuration>

### 4) User classes to store and fetch the data

**Store.java**

package Mypack2;

import org.hibernate.\*;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

public class Store {

public static void main(String[] args) {

StandardServiceRegistry ssr=new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta=new MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory=meta.getSessionFactoryBuilder().build();

Session session=factory.openSession();

Transaction t=session.beginTransaction();

Employee e1=new Employee();

e1.setName("Ravi Malik");

e1.setEmail("ravi@gmail.com");

Address address1=new Address();

address1.setAddressLine1("G-21,Lohia nagar");

address1.setCity("Vijayawada");

address1.setState("AP");

address1.setCountry("India");

address1.setPincode(201301);

e1.setAddress(address1);

address1.setEmployee(e1);

session.persist(e1);

t.commit();

session.close();

System.out.println("success");

}

}

Fetch.java

package Mypack2;

import java.util.Iterator;

import java.util.List;

import javax.persistence.TypedQuery;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.boot.Metadata;

import org.hibernate.boot.MetadataSources;

import org.hibernate.boot.registry.StandardServiceRegistry;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

public class Fetch {

public static void main(String[] args) {

StandardServiceRegistry ssr=new StandardServiceRegistryBuilder().configure("hibernate.cfg.xml").build();

Metadata meta=new MetadataSources(ssr).getMetadataBuilder().build();

SessionFactory factory=meta.getSessionFactoryBuilder().build();

Session session=factory.openSession();

TypedQuery query=session.createQuery("from Employee");

List<Employee> list=query.getResultList();

Iterator<Employee> itr=list.iterator();

while(itr.hasNext()){

Employee emp=itr.next();

System.out.println(emp.getEmployeeId()+" "+emp.getName()+" "+emp.getEmail());

Address address=emp.getAddress();

System.out.println(address.getAddressLine1()+" "+address.getCity()+" "+

address.getState()+" "+address.getCountry()+" "+address.getPincode());

}

session.close();

System.out.println("success");

}

}

SPRING CORE

1) Create the Java Project

Go to **File** menu - **New** - **project** - **Java Project**. Write the project name e.g. firstspring - **Finish**. Now the java project is created.

### 1) Create the Java Project

Go to **File** menu - **New** - **project** - **Java Project**. Write the project name e.g. firstspring - **Finish**. Now the java project is created.

### 2) Add spring jar files

There are mainly three jar files required to run this application.

* **org.springframework.core-3.0.1.RELEASE-A**
* **com.springsource.org.apache.commons.logging-1.1.1**
* **org.springframework.beans-3.0.1.RELEASE-A**

For the future use, You can download the required jar files for spring core application.

[download the core jar files for spring](https://static.javatpoint.com/src/sp/spcorejars.zip)

[download the all jar files for spring including aop, mvc, j2ee, remoting, oxm, etc.](https://static.javatpoint.com/src/sp/springjars.zip)

To run this example, you need to load only spring core jar files.

To load the jar files in eclipse IDE, **Right click on your project** - **Build Path** - **Add external archives** - **select all the required jar files** - **finish.**.

### 3) Create Java class

**package** Springdemo;

**public** **class** Student {

**private** **int** id;

**private** String firstname;

**private** String lastname;

@Override

**public** String toString() {

**return** "Student [id=" + id + ", firstname=" + firstname + ", lastname=" + lastname + "]";

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getFirstname() {

**return** firstname;

}

**public** **void** setFirstname(String firstname) {

**this**.firstname = firstname;

}

**public** String getLastname() {

**return** lastname;

}

**public** **void** setLastname(String lastname) {

**this**.lastname = lastname;

}

}

### 4) Create the xml file

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id=*"studentbean"* class=*"Springdemo.Student"*>

<property name=*"id"* value=*"245"*></property>

<property name=*"firstname"* value=*"santhosh"*></property>

<property name=*"lastname"* value=*"munta"*></property>

</bean>

</beans>

### 5) Create the test class

**package** Springdemo;

**import** org.springframework.beans.factory.BeanFactory;

**import** org.springframework.beans.factory.xml.XmlBeanFactory;

**import** org.springframework.core.io.ClassPathResource;

**import** org.springframework.core.io.Resource;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Resource resource=**new** ClassPathResource("emp.xml");

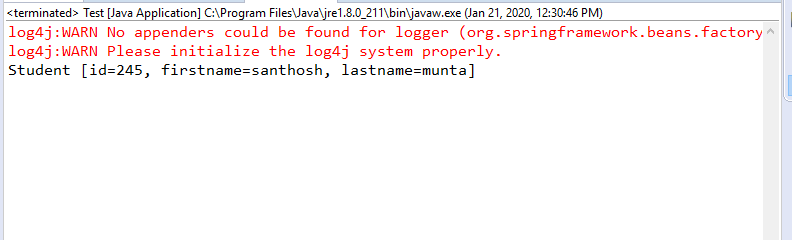
BeanFactory factory=**new** XmlBeanFactory(resource);

Student s=(Student)factory.getBean("studentbean");

System.***out***.println(s);

}

}



Constructor:

1. primitive and String-based values
2. Dependent object (contained object)
3. Collection values etc.

Injecting primitive and string-based values

Let's see the simple example to inject primitive and string-based values. We have created three files here:

* Employee.java
* app.xml
* Test.java

**package** Mypack3;

**public** **class** Employee {

**private** **int** id;

**private** String name;

**private** String location;

**public** Employee(**int** id, String name, String location) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.location = location;

}

@Override

**public** String toString() {

**return** "Employee [id=" + id + ", name=" + name + ", location=" + location + "]";

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id=*"e"* class=*"Mypack3.Employee"*>

<constructor-arg value=*"10"* type=*"int"*></constructor-arg>

<constructor-arg value=*"sreekanth"*></constructor-arg>

<constructor-arg value=*"mumbai"*></constructor-arg>

</bean>

</beans>

**package** Mypack3;

**import** org.springframework.beans.factory.BeanFactory;

**import** org.springframework.beans.factory.xml.XmlBeanFactory;

**import** org.springframework.core.io.\*;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Resource r=**new** ClassPathResource("app.xml");

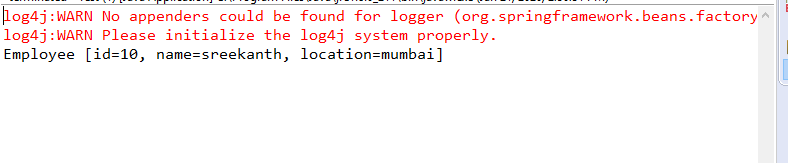
BeanFactory factory=**new** XmlBeanFactory(r);

Employee s=(Employee)factory.getBean("e");

System.***out***.println(s.toString());

}

}



**Using REFERENCE:**

**Address.java**

**package** Demo;

**public** **class** Address {

**private** String city;

**private** String state;

**private** String country;

**public** Address(String city, String state, String country) {

**super**();

**this**.city = city;

**this**.state = state;

**this**.country = country;

}

**public** String toString(){

**return** city+" "+state+" "+country;

}

}

**Employee.java**

**package** Demo;

**public** **class** Employee {

**private** **int** id;

**private** String name;

**private** Address address;//Aggregation

**public** Employee() {System.***out***.println("def cons");}

**public** Employee(**int** id, String name, Address address) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.address = address;

}

**void** show(){

System.***out***.println(id+" "+name);

System.***out***.println(address.toString());

}

}

apcon.xml

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id=*"add"* class=*"Demo.Address"*>

<constructor-arg value=*"vijayawada"*></constructor-arg>

<constructor-arg value=*"AP"*></constructor-arg>

<constructor-arg value=*"India"*></constructor-arg>

</bean>

<bean id=*"e"* class=*"Demo.Employee"*>

<constructor-arg value=*"123"* type=*"int"*></constructor-arg>

<constructor-arg value=*"saikumar"*></constructor-arg>

<constructor-arg ref=*"add"*></constructor-arg>

</bean>

</beans>

Test.java

**package** Demo;

**import** org.springframework.beans.factory.BeanFactory;

**import** org.springframework.beans.factory.xml.XmlBeanFactory;

**import** org.springframework.core.io.\*;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Resource r=**new** ClassPathResource("apcon.xml");

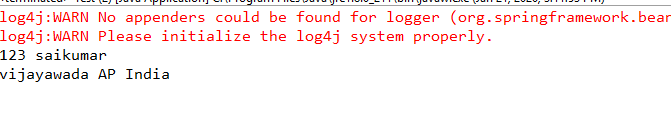
BeanFactory factory=**new** XmlBeanFactory(r);

Employee s=(Employee)factory.getBean("e");

s.show();

}

}

****

**Using List and reference:**

1. **Office.java**
2. **offcon.xml**
3. **Test.java**

package Demo1;

import java.util.Iterator;

import java.util.List;

public class Office {

private int id;

private List<String> dept;

private String company;

private List<Address> location;

public Office(int id, List<String> dept, String company, List<Address> location) {

super();

this.id = id;

this.dept = dept;

this.company = company;

this.location = location;

}

public void displayInfo(){

System.out.println(id+" "+dept+" "+" "+company+" "+location.toString());

}

}

**package** Demo1;

**public** **class** Address {

**private** String city;

**public** Address(String city) {

**super**();

**this**.city = city;

}

@Override

**public** String toString() {

**return** "Address [city=" + city + "]";

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"*>

<bean id=*"qwe"* class=*"Demo1.Address"*>

<constructor-arg value=*"vijayawada"*> </constructor-arg>

</bean>

<bean id=*"qw"* class=*"Demo1.Address"*>

<constructor-arg value=*"Gannavaram"*> </constructor-arg>

</bean>

<bean id=*"q"* class=*"Demo1.Office"*>

<constructor-arg value=*"111"*></constructor-arg>

<constructor-arg>

<list>

<value>Production</value>

<value>support</value>

</list>

</constructor-arg>

<constructor-arg value=*"google"*></constructor-arg>

<constructor-arg>

<list>

<ref bean=*"qwe"*/>

<ref bean=*"qw"*/>

</list>

</constructor-arg>

</bean>

</beans>

**package** Demo1;

**import** org.springframework.beans.factory.BeanFactory;

**import** org.springframework.beans.factory.xml.XmlBeanFactory;

**import** org.springframework.core.io.ClassPathResource;

**import** org.springframework.core.io.Resource;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Resource r=**new** ClassPathResource("offcon.xml");

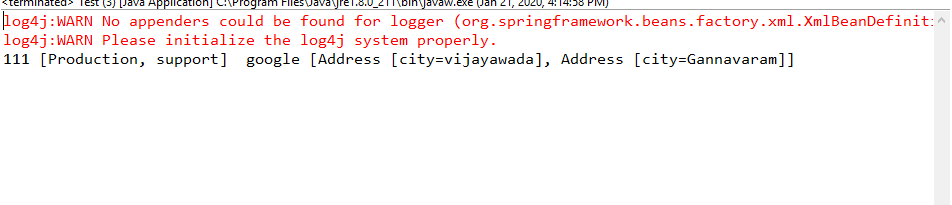
BeanFactory factory=**new** XmlBeanFactory(r);

Office of=(Office)factory.getBean("q");

of.displayInfo();

}

}

****

**SPRING MVC:**

**Index.jsp**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head>

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

<title>Insert title here</title>

</head>

<body>

<a href=*"hello"*>Click here...</a>

</body>

</html>

Hellocontroller.java

**package** qwertykey;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.web.bind.annotation.RequestMapping;

@Controller

**public** **class** HelloController {

@RequestMapping("/hello")

**public** String redirect()

{

**return** "viewpage";

}

@RequestMapping("/helloagain")

**public** String display()

{

**return** "index";

}

}

Web.xml:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://java.sun.com/xml/ns/javaee"* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"* id=*"WebApp\_ID"* version=*"2.5"*>

<servlet>

<servlet-name>spring</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>spring</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>

### **spring-servlet.xml(** bean in the xml file**)**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

xsi:schemaLocation=*"*

*http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context.xsd*

*http://www.springframework.org/schema/mvc*

*http://www.springframework.org/schema/mvc/spring-mvc.xsd"*>

<!-- Provide support for component scanning -->

<context:component-scan base-package=*"qwertykey"* />

<!--Provide support for conversion, formatting and validation -->

<mvc:annotation-driven/>

<!-- Define Spring MVC view resolver -->

<bean id=*"viewResolver"* class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<property name=*"prefix"* value=*""*></property>

<property name=*"suffix"* value=*".jsp"*></property>

</bean>

</beans>

### view components

### view.jsp

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head>

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

<title>Insert title here</title>

</head>

<body>

<p>Welcome to Spring MVC Tutorial</p>

</body>

### </html>

Spring MVC Signup/Login Form

Step1: Create HTML File

<!DOCTYPE html>

<html>

<head>

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

<title>Insert title here</title>

</head>

<form action=*"Set.cs"*method=*"post"*>

<body>

<table align=*"center"*>

<tr><td>NAME</td><td><input type=*"text"* name=*"name"*></td></tr>

<tr><td>LOCATION</td><td><textarea rows=*"5"*cols=*"10"*name=*"address"*></textarea></td></tr>

<tr><td>Mail</td><td><input type=*"text"* name=*"email"*></td></tr>

<tr><td>TECH</td><td><input type=*"text"* name=*"course"*></td></tr>

<tr><td>UserName : </td><td><input type=*"text"* name=*"uname"*/></td></tr>

<tr><td>Password : </td><td><input type=*"text"* name=*"pass"*/></td></tr>

<tr><td><input type=*"submit"* value=*"login"*></td><td><input type=*"Reset"* value=*"reset"*></td></tr>

</table>

</form>

</body>

</html>

Step2: Design XML File( url pattern is checked, and spring servlet name is noticed)

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://java.sun.com/xml/ns/javaee"* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"* id=*"WebApp\_ID"* version=*"2.5"*>

<display-name>spmvclogin</display-name>

<servlet>

<servlet-name>spring</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>spring</servlet-name>

<url-pattern>\*.cs</url-pattern>

</servlet-mapping>

</web-app>

Step3: Design spring-servlet File(Package name is checked)

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

xsi:schemaLocation=*"*

*http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context.xsd*

*http://www.springframework.org/schema/mvc*

*http://www.springframework.org/schema/mvc/spring-mvc.xsd"*>

<!-- Provide support for component scanning -->

<context:component-scan base-package=*"spmvclogin"* />

<!--Provide support for conversion, formatting and validation -->

<mvc:annotation-driven/>

<bean id=*"viewResolver"* class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<property name=*"prefix"* value=*""*></property>

<property name=*"suffix"* value=*".jsp"*></property>

</bean>

</beans>

Step4: Controller class (url @RequestMapping is identified)

**package** spmvclogin;

**import** javax.servlet.http.HttpServletRequest;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestParam;

@Controller

**public** **class** Datacontr {

**static** String *fullName*;

**static** String *userName*;

**static** String *password*;

@RequestMapping("/Set")

**public** String data(@RequestParam("name") String name,@RequestParam("uname") String uname,@RequestParam("pass") String pass,Model m)

{

*fullName*=name;

*userName*=uname;

*password*=pass;

String msg="Hello "+ name+" your account is created";

//add a message to the model

m.addAttribute("message", msg);

**return** "viewpage";

}

@RequestMapping("/Find")

**public** String display(@RequestParam("name") String name,@RequestParam("pass") String pass,Model m)

{

**if**(!(name.equals(*userName*)))

{

String msg="Incorrect username";

//add a message to the model

m.addAttribute("message", msg);

**return** "page";

}

**else** **if**(!(pass.equals(*password*)))

{

String msg="Incorrect password";

//add a message to the model

m.addAttribute("message", msg);

**return** "page";

}

**else**

{

String msg="Welcome "+*fullName*;

m.addAttribute("message", msg);

**return** "page";

}

}

}

Step5: Viewpage to display(login page)

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head>

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

<title>Insert title here</title>

</head>

<body>

${message}

<jsp:include page=*"login.jsp"*></jsp:include>

</body>

</html>

Step6: login to display

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head>

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

<title>Insert title here</title>

</head>

<body>

<form action=*"Find.cs"* method=*"post"*>

UserName : <input type=*"text"* name=*"name"*/> <br><br>

Password : <input type=*"text"* name=*"pass"*/> <br><br>

<tr><td><input type=*"submit"* value=*"login"*></td><td><input type=*"Reset"* value=*"Reset"*></td></tr>

</form>

</body>

</html>

Step7: Find is detected rom controller class and logic is executed

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head>

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

<title>Insert title here</title>

</head>

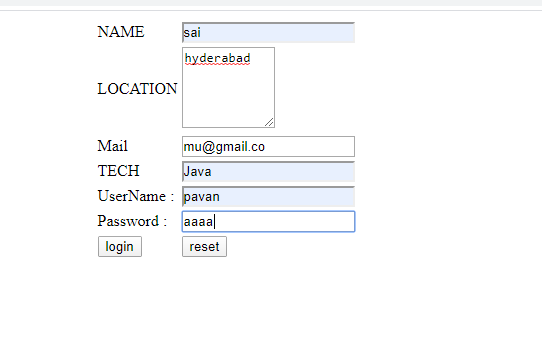
<body>

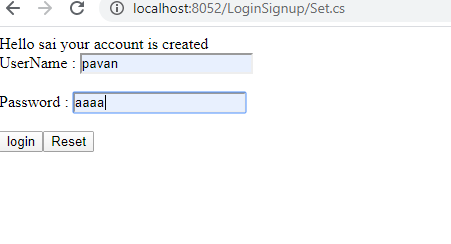
${message}

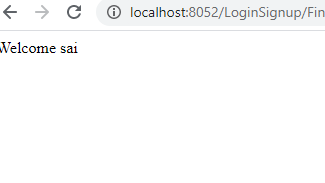
</body>

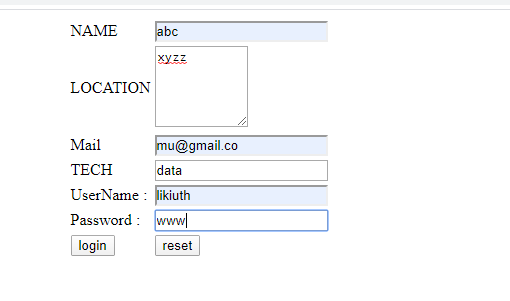
</html>

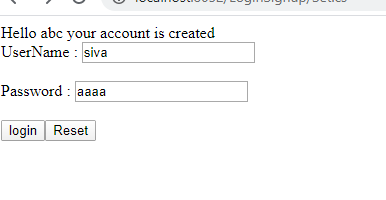
OUTPUT:











If username and password are correct

