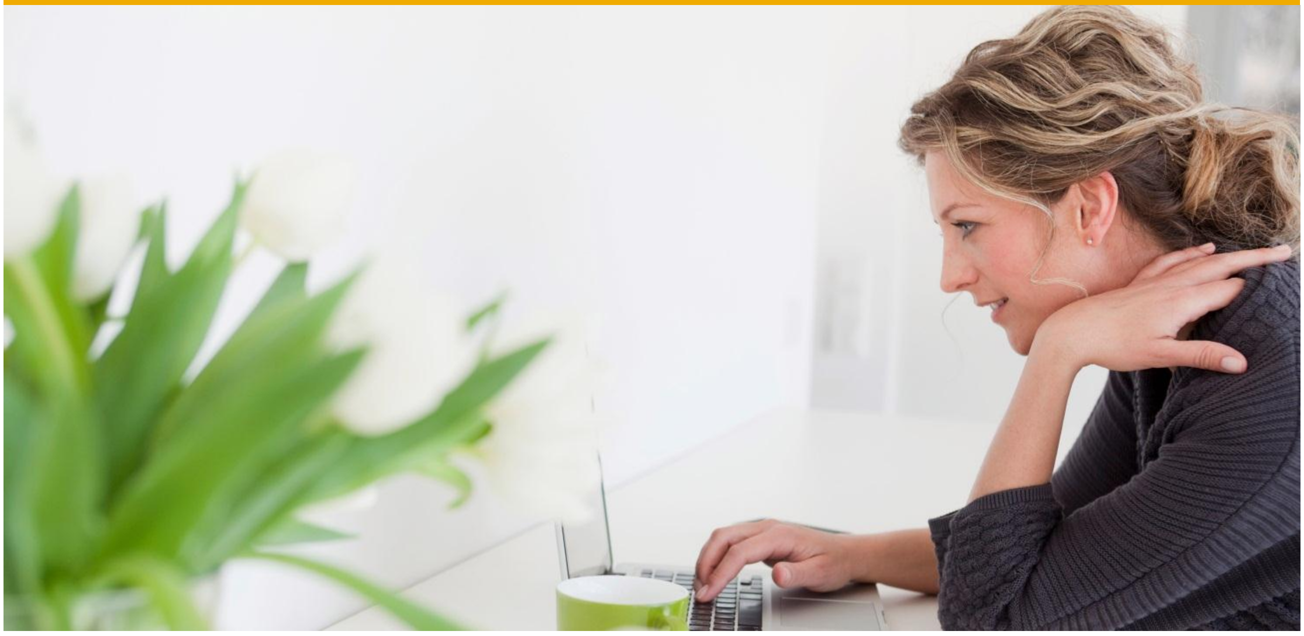


Developing Web Applications with Sybase ASE, Spring MVC and JPA

Step by step tutorial on developing Java web applications with mission critical Sybase ASE database.

TABLE OF CONTENTS

Goal of this tutorial is to explain how web application based on Sybase ASE database can be developed in Java using most popular Spring MVC framework.

This tutorial explains how to build simple Contacts management web application for managing list of contacts using Sybase ASE Developer Edition. Tomcat container, Spring MVC web framework, Hibernate implementation of JPA and Eclipse IDE will be used to develop end to end application.

APPLICATION ARCHITECTURE

Contact Management will be 3 tier application- web tiers on JSP/Servlet Tomcat Web Server, Sybase ASE database, and client browser can be on any machine. Users can create, read and delete contacts a web browser once application is deployed.

For simplicity sake in development environment, Sybase ASE, and Tomcat server are both installed on same machine.
Again, to simplify our example and keep focus on configuration and setup issues, entire business logic of contact management is modeled after one Contacts Table in the database

Contacts Table is mapped to a JPA entity and its management done by a Service class. Spring MVC Controller will be invoking Service class methods depending on operation requested. This web application deployed as self-containing WAR file.
Here is SQL DDL for our Contacts table. Table name could be anything. For our purpose, table is created in pubs3 database bundled with Sybase ASE.

```sql
CREATE TABLE pubs3.dbo.asespring_contact
(id numeric (7,0) identity,
firstname  varchar(25) NULL,
lastname varchar(25) NULL,
telephone varchar(15) NULL,
email varchar(30) null,
primary key (id),created datetime default getDate());
```

PRIOR KNOWLEDGE

Before starting this tutorial, it is assumed that reader is aware of following technologies

- Java 5 Programming including annotations, web tier with JSP / Servlet using Tomcat.
- Familiarity with Spring MVC framework – Dependency Injection, Model View Controller,
- Basic database operations with SQL – create, query table using simple SQL
- Java Persistence API (JPA) Object to Relationship Management – What is Entity Manager, Managed/Non Managed entities, Persistence context

GETTING STARTED

Following components would need to be downloaded on local machine before configuring Eclipse IDE.

1. Download Eclipse EE IDE for Java EE Developers ,Indigo release
   a. http://www.eclipse.org/downloads/packages/release/indigo/sr2
2. Java 6 SDK
   a. http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html

3. Spring MVC 3.2.0 Release
    a. http://www.springsource.org/download/community
    b. http://repo.springsource.org/libs-release-local/org/springframework/spring/3.2.0.RELEASE/spring-framework-3.2.0.RELEASE-dist.zip
    c. Spring aop alliance 1.0 jar http://ebr.springsource.com/repository/app/bundle/version/detail?name=com.springsource.org.aopalliance&version=1.0.0
4. Hibernate JPA
    a. http://sourceforge.net/projects/hibernate/files/hibernate4/4.1.8.Final/
5. Sybase ASE Developer Edition 64 bit
    a. http://www.sybase.com/ase_1500devel
    b. Install at c:\sybase (for Windows platform)
    c. Note Sybase TDS JDBC Driver jconn4.jar from c:\Sybase\jConnect-7_0\classes

Configure Sybase ASE

Use Sybase Control Center, isql , or favorite SQL editor to create a user and Contacts Table for Sybase database. In this case, Eclipse bundled Database management tool was used by creating a new connection Profile , registering Sybase JDBC driver with Eclipse.

```
CREATE TABLE pubs3.dbo.asespring_contact
(id numeric (7,0) identity,
firstname  varchar(25) NULL,
lastname varchar(25) NULL,
telephone varchar(15) NULL,
email varchar(30) null,
primary key (id),created datetime default getDate());
```

a. Verify that new table exists by browsing Sybase dbo schema
b. JDBC URL for Sybase jConnect jCon4.jar driver is *jdbc:sybase:Tds:<my_machine>:5000/pubs3*
c. JDBC Driver Class *com.sybase.jdbc4.jdbc.SybDriver*

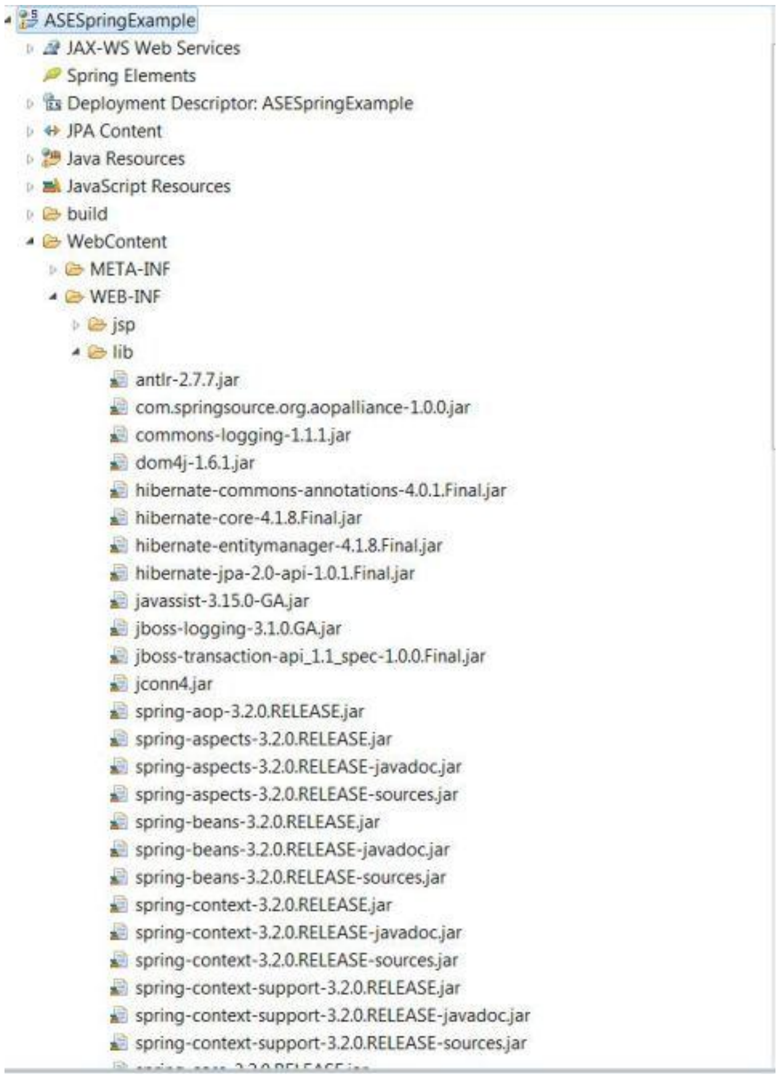
Configuring Eclipse IDE

In Eclipse, select Create new Dynamic Web Project, naming it ASESpringExample and store it in default location

Copy following jar files to WebContent/WEB-INF/lib directory in Eclipse

- Entire Spring 3.2.x jars
- hibernate-entitymanager-4.1.8.final under hibernate/jpa
- All jar files under hibernate/lib/required
- Sybase jConn4.jar
- Spring aop alliance jar

Screenshot of how WEB-INF/lib should look after copying.



```
ASESpringExample
  JAX-WS Web Services
  Spring Elements
  Deployment Descriptor: ASESpringExample
  JPA Content
  Java Resources
  JavaScript Resources
  build
  WebContent
    META-INF
    WEB-INF
      jsp
      lib
        antlr-2.7.7.jar
        com.springsource.org.aopalliance-1.0.0.jar
        commons-logging-1.1.1.jar
        dom4j-1.6.1.jar
        hibernate-commons-annotations-4.0.1.Final.jar
        hibernate-core-4.1.8.Final.jar
        hibernate-entitymanager-4.1.8.Final.jar
        hibernate-jpa-2.0-api-1.0.1.Final.jar
        javassist-3.15.0-GA.jar
        jboss-logging-3.1.0.GA.jar
        jboss-transaction-api_1.1_spec-1.0.0.Final.jar
        jconn4.jar
        spring-aop-3.2.0.RELEASE.jar
        spring-aspects-3.2.0.RELEASE.jar
        spring-aspects-3.2.0.RELEASE-javadoc.jar
        spring-aspects-3.2.0.RELEASE-sources.jar
        spring-beans-3.2.0.RELEASE.jar
        spring-beans-3.2.0.RELEASE-javadoc.jar
        spring-beans-3.2.0.RELEASE-sources.jar
        spring-context-3.2.0.RELEASE.jar
        spring-context-3.2.0.RELEASE-javadoc.jar
        spring-context-3.2.0.RELEASE-sources.jar
        spring-context-support-3.2.0.RELEASE.jar
        spring-context-support-3.2.0.RELEASE-javadoc.jar
        spring-context-support-3.2.0.RELEASE-sources.jar
```

- Create directory jsp under WebContent/WEB-INF/jsp
  - This directory will contain all our application specific JSP's – contact.jsp
- Add new new JPA Entity under Project→ New JPA Entity
  - Add empty com.asesample.entity.Contact Class with @Entity annotation. Details of class below.

XML CONFIGURATION

Contact Management application is mixture of annotation and framework specific XML files- though recent most release

Configure Web application xml (web.xml)

Spring Dispatcher servlet mapping which specifies which URL will be handled by Spring MVC framework. In this example, root "/" is specified

- Dispatcher servlet name corresponds to Spring Context configuration file + "servlet" suffix. As servlet name here is "spring" , corresponding Spring configuration file is spring-servlet.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"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
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>ASESpringExample</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>/</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

Configure JPA Configuration files

- Persistence.xml Persistence.xml is located at JavaResource/META-INF/
- This file gets automatically created when creating JPA Entity by right clicking on Project.
- As we are using Hibernate as JPA Provider, add provider to be org.hibernate.ejb.HibernatePersistence in persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
      <persistence-unit name="contactsybase" transaction-
type="RESOURCE_LOCAL">
      <provider>org.hibernate.ejb.HibernatePersistence</provider>
      <class>com.asesample.entity.Contact</class>
      <properties>
      <property name="hibernate.show_sql" value="true" />
        <property name="hibernate.format_sql" value="false" />
        <property name="hibernate.connection.driver_class"
                      value="com.sybase.jdbc4.jdbc.SybDriver" />
        <property name="hibernate.connection.url"
```

```
                              value="jdbc:sybase:Tds:<my_machine>/pubs3" />
                <property name="hibernate.connection.username" value="sa" />
                <property name="hibernate.connection.password" value="" />

                <property name="javax.persistence.jdbc.driver"
        value="com.sybase.jdbc4.jdbc.SybDriver"/>
                <property name="javax.persistence.jdbc.url"
        value="jdbc:sybase:Tds:<my_machine>:5000/pubs3"/>
                <property name="javax.persistence.jdbc.user" value="sa"/>
                <property name="javax.persistence.jdbc.password" value=""/>
              </properties>

              </persistence-unit>
              </persistence>
```

Replace <my_machine> with localhost name of your machine.

Configure Spring configuration XML

This file is located at

- WebContent/WEB-INF/spring-servlet.xml

It will contain following sections

- Enable Spring beans : ContactService Bean manages create, delete and delete of Contact Bean
- Enable JPA Entity Manager auto-injection :
- Spring Transaction Management : Add JPA Transaction Manager Bean,
- View resolvers

Here is complete Spring Context configuration XML located at WebContent/WEB-INF/spring-servlet.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"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:context="http://www.springframework.org/schema/context"
    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-3.0.xsd
                           http://www.springframework.org/schema/tx
                           http://www.springframework.org/schema/tx/spring-
tx.xsd">



    <context:component-scan base-package="com.asesample" />

    <bean id="viewResolver"
        class="org.springframework.web.servlet.view.UrlBasedViewResolver">
        <property name="viewClass"
            value="org.springframework.web.servlet.view.JstlView" />
        <property name="prefix" value="/WEB-INF/jsp/" />
        <property name="suffix" value=".jsp" />
    </bean>

    <tx:annotation-driven/>


   <bean id="dataSource"
      class="org.springframework.jdbc.datasource.DriverManagerDataSource">
      <property name="driverClassName" value="com.sybase.jdbc4.jdbc.SybDriver"/>
      <property name="url" value="jdbc:sybase:Tds:<machine_name>:5000/pubs3"/>
      <property name="username" value="sa"/>
      <property name="password" value=""/>
      </bean>

      <bean id="entityManagerFactory"
class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
      <property name="dataSource" ref="dataSource" />
        <property name="jpaVendorAdapter">
            <bean
class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter" >
            <property name="database" value="SYBASE"/>
            <property name="showSql" value="true"/>
            </bean>
        </property>
    </bean>


    <bean id="contactService"
class="com.asesample.entity.ContactServiceImpl"></bean>
    <bean id="transactionManager"
class="org.springframework.orm.jpa.JpaTransactionManager"
     p:entityManagerFactory-ref="entityManagerFactory"/>

  <bean

class="org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcesso
r" />
</beans>
```

Replace <my_machine> with localhost name of your machine.

SOURCE CODE

Web Resources

We will use Spring Form tag library to bind data from JSP to JPA Model. Here is complete contact.jsp listing.

This gets invoked /index.html is redirected to contact.jsp by Spring MVC Controller.

```
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
<head>
    <title>Spring 3 MVC Series - Contact Manager</title>
</head>
<body>
<h2>Contact Manager</h2>
<form:form method="post" action="add.html" commandName="contact">

    <table>
    <tr>
        <td><form:label path="firstname">First Name</form:label></td>
        <td><form:input path="firstname" /></td>
    </tr>
    <tr>
        <td><form:label path="lastname">Last Name</form:label></td>
        <td><form:input path="lastname" /></td>
    </tr>
    <tr>
        <td><form:label path="email">Email</form:label></td>
        <td><form:input path="email" /></td>
    </tr>
    <tr>
        <td><form:label path="telephone">Telephone</form:label></td>
        <td><form:input path="telephone" /></td>
    </tr>
    <tr>
        <td colspan="2">
            <input type="submit" value="Add Contact"/>
        </td>
    </tr>
</table>

</form:form>

<h3>Contacts</h3>
<c:if  test="${!empty contactList}">
<table class="data">
<tr>
    <th>Name</th>
    <th>Email</th>
    <th>Telephone</th>
    <th>&nbsp;</th>
</tr>
<c:forEach items="${contactList}" var="contact">
    <tr>
        <td>${contact.lastname}, ${contact.firstname} </td>
        <td>${contact.email}</td>
        <td>${contact.telephone}</td>
        <td><a href="delete/${contact.id}">delete</a></td>
    </tr>
</c:forEach>
</table>
</c:if>
</body>
      </html>
```

Java Resources

- Contact.java – a JPA Entity with @Entity annotation and containing Table, Column mapping for database columns to Java fields- firstname, lastname, email, telephone. Primary key is auto-generated by Sybase ASE with IDENTITY type column and hence annotated @GeneratedValue

POJO (Model Object) –JPA Entity for Contact

```
package com.asesample.entity;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;


@Entity
@Table(name="deepa_contact1")
public class Contact {

      @Id
      @Column(name="id")
      @GeneratedValue
      private Integer id;

      @Column(name="firstname")
      private String firstname;

      @Column(name="lastname")
       private String lastname;

      @Column(name="email")
       private String email;

      @Column(name="telephone")
       private String telephone;

      … //getters and setters for above field.
```

- ContactService.java and ContactServiceImpl.java - Service Interface and Implementation class which has JPA Persistence Context injected into it by Spring and manages create, update, delete for Contact object through Entity Manager Factory.

```
package com.asesample.entity;

@Transactional(readOnly=false,propagation=Propagation.REQUIRES_NEW)
public class ContactServiceImpl implements ContactService {

  private EntityManager em;
```

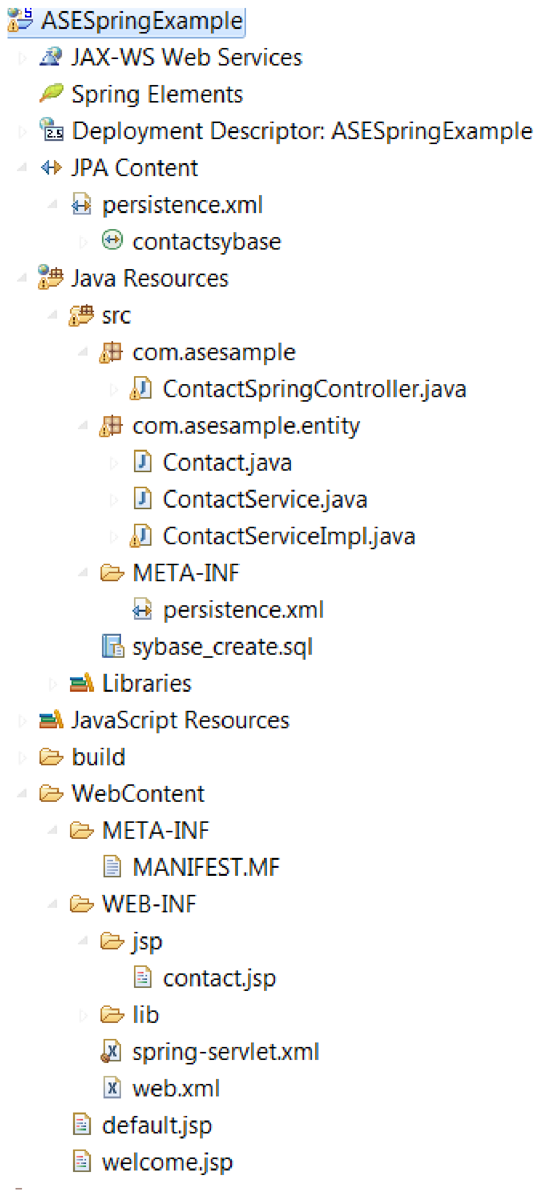
```
 @PersistenceContext(unitName="contactsybase")
    public void setEntityManager(EntityManager entityManager) {
            this.em = entityManager;
    }

 @Override
 @Transactional(readOnly=false,propagation=Propagation.REQUIRES_NEW)
 public void addContact(Contact contact) {
     // TODO Auto-generated method stub
     System.out.println("Adding new contact..");
     Contact c=new Contact();
     c.setEmail(contact.getEmail());
     c.setFirstname(contact.getFirstname());
     c.setLastname(contact.getLastname());
     c.setTelephone(contact.getTelephone());
     em.persist(contact);

 }

 @Transactional(readOnly=false,propagation=Propagation.REQUIRED)
 public List<Contact> listContact() {
     Query q=em.createQuery("select c from Contact c");
                 return q.getResultList();
 }

 @Transactional(readOnly=false,propagation=Propagation.REQUIRED)
 public void removeContact(Integer id) {
     Contact c=em.find(Contact.class, id);
     em.remove(c);


 }

}
```

- ContactSpringController.java – Spring MVC Controller has HTTP URL mapping for /add, /delete, /index for listing list of Contacts in database. This controller calls ContactService methods to perform database operation and fetches data to and fro from JSP front end

Note @Transactional annotation on methods-without these annotation, transaction context won't get propagated to JPA entities in persistence context

```
package com.asesample;

@Controller
@SessionAttributes
public class ContactSpringController {

 @Autowired
 ContactService contactService;

 @Transactional
 @RequestMapping("/index")
 public String listContacts(Map <String,Object>map){
     System.out.println("ContactSpringController-listContacts invoked");
     map.put("contact", new Contact());
     map.put("contactList",contactService.listContact());
     return "contact";
}

@Transactional
 @RequestMapping(value = "/add", method = RequestMethod.POST)
   public String addContact(@ModelAttribute("contact")
                             Contact contact, BindingResult result) {


  try{
       contactService.addContact(contact);
       }catch(Exception e){
           e.printStackTrace();
       }

       return "redirect:index.html";
   }

@Transactional
@RequestMapping(value = "delete/{contactId}")
   public String removeContact(@PathVariable("contactId")
   Integer contactId) {


       try{
       contactService.removeContact(contactId);
       }catch(Exception e){ e.printStackTrace();  }

       return "redirect:/index.html";
   }

}
```

Complete Eclipse Source Code Tree

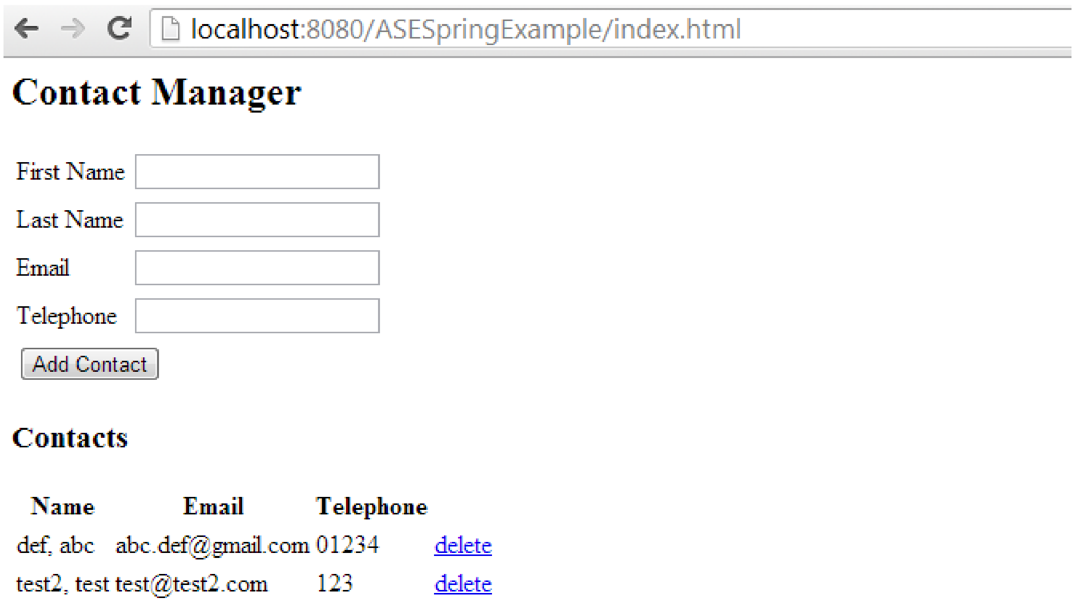


```
ASESpringExample
  JAX-WS Web Services
  Spring Elements
  Deployment Descriptor: ASESpringExample
  JPA Content
    persistence.xml
      contactsybase
  Java Resources
    src
      com.asesample
        ContactSpringController.java
      com.asesample.entity
        Contact.java
        ContactService.java
        ContactServiceImpl.java
      META-INF
        persistence.xml
      sybase_create.sql
    Libraries
  JavaScript Resources
  build
  WebContent
    META-INF
      MANIFEST.MF
    WEB-INF
      jsp
        contact.jsp
      lib
      spring-servlet.xml
      web.xml
    default.jsp
    welcome.jsp
```

DEPLOY AND RUN APPLICATION IN BROWSER

Right click on application in Eclipse→Run As → Run on Server, Select Tomcat 6.0 and click Finish. Monitor console logs below to make sure there are no deployment errors. Application will now be built, deployed as WAR file to server with context root – “ASESpringExample”

Go to following URL to run application in browser – http://localhost:8080/ASESpringExample/index.html



Name	Email	Telephone	
def, abc	abc.def@gmail.com	01234	delete
test2, test	test@test2.com	123	delete

As "/index" URL is mapped to Controller Method "listContacts" in ContactSpringController.java with return view as "contact", contact.jsp will be rendered with entry form and list of existing contacts in table.

As primary key is auto-generated, put any data, press Add Contact and see it appear below. View data stored in table directly to verify new data has been added.

www.sap.com