**Table of Content**

|  |
| --- |
| JNDI Configuration in Server |
| Creating JNDI configuration in Server |
| JNDI Configuration in Application |
| Root Context Configuration |
| Log Property |
| Error Page Path |
| Relocating Property Files |
| Web Configuration Files |
| Server Configuration Files |
| Transaction Management Change |
| LOB Handler Change |
| JAR file included |
| TLD Configuration |
| Virtual Host |
| Server Configuration |
| Domain Name System |
| Port Configuration |

# JNDI Configuration in Server:

For database connectivity we are using JNDI (Java Naming and Directory Interface). So we are configuring JNDI in Tomcat’s *server.xml.* It contains database connection properties like Resource URL, Port, Username, and Password.

## Creating JNDI configuration in Server:

File: apache-tomcat-8.0.27\conf\server.xml

<GlobalNamingResources>

…

<Resource name="jdbc/CrmData" auth="Container"

type="javax.sql.DataSource"

driverClassName="oracle.jdbc.OracleDriver"

url="jdbc:oracle:thin:@10.8.18.15:1521:bsro"

username="rtms\_webdb" password="rtms\_webdb" maxActive="20"

maxIdle="10" maxWait="-1" />

</GlobalNamingResources>

File: apache-tomcat-8.0.27\conf\context.xml

<ResourceLink name="jdbc/CrmData" global="jdbc/CrmData"   
 type="javax.sql.DataSource" />

## JNDI configuration in Application:

We are referring the JNDI resource in our application.

File: FCAC\src\main\webapp\WEB-INF\spring\daoContext.xml

<bean id="bfrcDataSource" class="org.springframework.jndi.JndiObjectFactoryBean">

<property name="jndiName" value="java:/comp/env/jdbc/CrmData" />  
<property name="resourceRef" value="false" />

</bean>

# Root Context Configuration:

To run our application in Tomcat root context, we have to mention our application name into the **docBase** attribute.

File: apache-tomcat-8.0.27\conf\server.xml

<Host name="localhost"  appBase="webapps" unpackWARs="true" autoDeploy="true">

<Context path="" **docBase**="FCAC">

<WatchedResource>WEB-INF/web.xml</WatchedResource>

</Context>  
 </Host>

# Log Property:

Update log property file configurations according to write our application logs in to a separate file.

File: FCAC\src\main\resources\log4j.properties

log4j.rootLogger=INFO, file

#Redirect to Tomcat logs folder

log4j.appender.file.File=${catalina.home}/logs/**Fcac.log**

log4j.appender.file.MaxFileSize=10MB

log4j.appender.file.MaxBackupIndex=10

log4j.appender.file.layout=org.apache.log4j.PatternLayout

log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n

# Error Page Path:

Error page path have been updated to make it as identifiable by Tomcat. Added front-slash before the error page path.

File: FCAC\src\main\webapp\WEB-INF\web.xml

/WEB-INF/views/error/404new.jsp

/WEB-INF/views/error/500new.jsp

# Relocating Property Files

Spring reads the resource only if the properties files available inside the **resource** folder. So the property files have been moved accordingly.

* *FCAC\src\main\webapp\WEB-INF\serverconfig* to *FCAC\src\main\resources\serverconfig*
* *FCAC\src\main\webapp\WEB-INF\views\completely-firestone\xml* to *FCAC\src\main\resources\xml  
  (For this, we updated the file path in* ***completely.firestone.default.resource.directory*** *under* ***web-config.properties file****)*

## Server Configuration Files:

We have to **modify** the below bean creation tag to read property files in Java code:

File: FCAC\src\main\webapp\WEB-INF\applicationContext.xml

<bean id="resourceLoader" class="org.springframework.core.io.DefaultResourceLoader" />

<bean id="propertyConfigurer" class="com.bfrc.framework.util.PropertyConfigUtil">

<constructor-arg type="org.springframework.core.io.ResourceLoader" ref="resourceLoader"/>

<constructor-arg type="java.lang.String" value="web-config"/>

<property name="defaultLocation" value="/serverconfig/" />

</bean>

# Transaction Management:

We have to remove JTA Transaction (which had been integrated with WebSphere), instead we are using Hibernate Transaction Manager. So we have removed JTA Transaction configurations and used Hibernate Transaction Manager (which is already exist) in daoContext.xml

File: FCAC\src\main\webapp\WEB-INF\spring\daoContext.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"

xsi:schemaLocation="

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

<!-- WebSphere/Oracle LOB Handling -->

<!-- <bean id="nativeJdbcExtractor" class="org.springframework.jdbc.support.nativejdbc.WebSphereNativeJdbcExtractor" /> -->

<bean id="nativeJdbcExtractor" class="org.springframework.jdbc.support.nativejdbc.SimpleNativeJdbcExtractor" />

<bean id="oracleLobHandler" class="org.springframework.jdbc.support.lob.OracleLobHandler" lazy-init="true">

<property name="nativeJdbcExtractor">

<ref local="nativeJdbcExtractor" />

</property>

</bean>

<!-- JTA Transactions -->

**<!— Removed the following JTA Transaction Manager -->**

**<!—**

**<bean id="jtaTransactionManager" class="org.springframework.transaction.jta.WebSphereTransactionManagerFactoryBean" />**

**<bean id="jtaTransactionManagerTransactionManager" class="org.springframework.beans.factory.config.MethodInvokingFactoryBean">**

**<property name="targetObject" ref="jtaTransactionManager"/>**

**<property name="targetMethod" value="getTransactionManager"/>**

**</bean>**

**<bean id="txManager" class="org.springframework.transaction.jta.JtaTransactionManager" /> -->**

<bean id="bfrcDataSource" class="org.springframework.jndi.JndiObjectFactoryBean">

<property name="jndiName" value="java:/comp/env/jdbc/CrmData" />

<property name="resourceRef" value="false" />

</bean>

<!-- <bean id="bfrcDataSource" class="org.apache.tomcat.jdbc.pool.DataSource"> -->

<!-- <property name="driverClassName" value="oracle.jdbc.OracleDriver" /> -->

<!-- <property name="url" value="${fcac.db.url}" /> -->

<!-- <property name="username" value="${fcac.db.username}" /> -->

<!-- <property name="password" value="${fcac.db.password}" /> -->

<!-- <property name="maxActive" value="5" /> -->

<!-- </bean> -->

<bean id="bfrcSessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

<property name="lobHandler">

<ref local="oracleLobHandler" />

</property>

<property name="dataSource">

<ref local="bfrcDataSource" />

</property>

<property name="configLocation" value="classpath:/com/bfrc/framework/dao/hibernate3/hibernate.cfg.xml" />

<property name="hibernateProperties">

<value>

hibernate.show\_sql=${hibernate.show\_sql}

hibernate.dialect=org.hibernate.dialect.Oracle10gDialect

</value>

</property>

</bean>

<bean id="apptDataSessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

<!-- <property name="lobHandler"> -->

<!-- <ref local="oracleLobHandler" /> -->

<!-- </property> -->

<property name="dataSource">

<ref local="bfrcDataSource" />

</property>

<property name="configLocation" value="classpath:/com/bfrc/apptdata/dao/hibernate.cfg.xml" />

</bean>

<bean id="tireSaleDataSessionFactory" class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

<!-- <property name="lobHandler"> -->

<!-- <ref local="oracleLobHandler" /> -->

<!-- </property> -->

<property name="dataSource">

<ref local="bfrcDataSource" />

</property>

<property name="configLocation" value="classpath:/com/bsro/tiresaledata/dao/hibernate.cfg.xml" />

</bean>

<!—Modified the Transaction Manager Bean ID hence txManager is used all over the application -->

<!-- <bean id="**bfrcTransactionManager**" class="org.springframework.orm.hibernate3.HibernateTransactionManager"> -->

<bean id="**txManager**" class="org.springframework.orm.hibernate3.HibernateTransactionManager">

<property name="sessionFactory">

<ref local="bfrcSessionFactory" />

</property>

</bean>

</beans>

# LOB Handler Change:

Updated LOB handler from WebSphereNativeJdbcExtractor to CommonsDbcpNativeJdbcExtractor . Hence we are not using WebSphere.

File: FCAC\src\main\webapp\WEB-INF\spring\daoContext.xml

<!--<bean id="nativeJdbcExtractor" class="org.springframework.jdbc.support.nativejdbc.WebSphereNativeJdbcExtractor"/>-->  
<bean id="nativeJdbcExtractor" class="org.springframework.jdbc.support.nativejdbc.CommonsDbcpNativeJdbcExtractor"/>

# JAR file included:

We need to add the below list of JAR files into Tomcat library (apache-tomcat-8.0.27\lib).

1. javamail-mail-1.4
2. jstl-1.2
3. ojdbc6
4. jbossall-client-3.2.3

# TLD Configuration:

To read TLD files, we have to update the body-content from **JSP** to **scriptless**, **all over the file**.

File:FCAC\src\main\webapp\WEB-INF\bsro-widgets.tld

<!-- <body-content>JSP</body-content> -->  
<body-content>scriptless</body-content>

# Virtual Host:

## Server Configuration:

Add virtual host, mapped with the application path in server.xml configuration.

File: apache-tomcat-8.0.27\conf\server.xml

<Host name="**firestonecompleteautocare.com**" appBase="webapps" unpackWARs="true" autoDeploy="true">  
 <Alias>**www. firestonecompleteautocare.com**</Alias>  
 <Context path="" docBase="**fcac**" debug="0" reloadable="true"/>  
</Host>

## Domain Name System:

To simulate Domain Name System, we need to add the **domain with server machine IP address** in hosts file which will root the URL request to our local server itself. So that, the domain will trigger local application.

File: /etc/hosts

10.142.10.121 firestonecompleteautocare.com  
10.142.10.121 www.firestonecompleteautocare.com

## Port Configuration:

We can call the application in browser with [*www.firestonecompleteautocare.com:8080*](http://www.firestonecompleteautocare.com:8080)hence Tomcat runs in 8080 port. To make the URL more common we can change to **Default Port** **80.** So that we can simple call by [www.firestonecompleteautocare.com](http://www.firestonecompleteautocare.com)

File: apache-tomcat-8.0.27\conf\server.xml

<Connector port="**80**" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />