Hibernate Tomcat JNDI DataSource Example Tutorial

APRIL 2, 2018 BY PANKAJ — 12 COMMENTS

Welcome to the Hibernate Tomcat JNDI DataSource example tutorial. We have already seen how to use Hibernate ORM tool in standalone java application, today we will learn how to use **Hibernate with DataSource** in Tomcat servlet container.

Using hibernate in web application is very easy, all we need is to configure `DataSource` properties in hibernate configuration file. First of all we need to setup test database and JNDI DataSource in tomcat container.

Table of Contents [hide]

Hibernate DataSource JNDI Example Database Setup

I am using MySQL for my example, below script is executed to create a simple table and insert some values into it.

`employee.sql`

```
CREATE TABLE `Employee` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(20) DEFAULT NULL,
  `role` varchar(20) DEFAULT NULL,
  `insert_time` datetime DEFAULT NULL,
```

```
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=26 DEFAULT CHARSET=utf8;

INSERT INTO `Employee` (`id`, `name`, `role`, `insert_time`)
VALUES
        (3, 'Pankaj', 'CEO', now());
INSERT INTO `Employee` (`id`, `name`, `role`, `insert_time`)
VALUES
        (14, 'David', 'Developer', now());
```

The Database schema name is **TestDB**.

Tomcat JNDI DataSource Configuration

For configuring tomcat container to initialize DataSource, we need to make some changes in tomcat server.xml and context.xml files.

server.xml

```
<Resource name="jdbc/MyLocalDB"
      global="jdbc/MyLocalDB"
      auth="Container"
      type="javax.sql.DataSource"
      driverClassName="com.mysql.jdbc.Driver"
      url="jdbc:mysql://localhost:3306/TestDB"
      username="pankaj"
      password="pankaj123"

      maxActive="100"
      maxIdle="20"
      minIdle="5"
      maxWait="10000"/>
```

Add above resource in the server.xml `GlobalNamingResources` element.

context.xml

```
<ResourceLink name="jdbc/MyLocalDB"
              global="jdbc/MyLocalDB"
              auth="Container"
              type="javax.sql.DataSource" />
```

Add above `ResourceLink` in the context.xml file, it's required so that applications can access the JNDI resource with name `jdbc/MyLocalDB`.

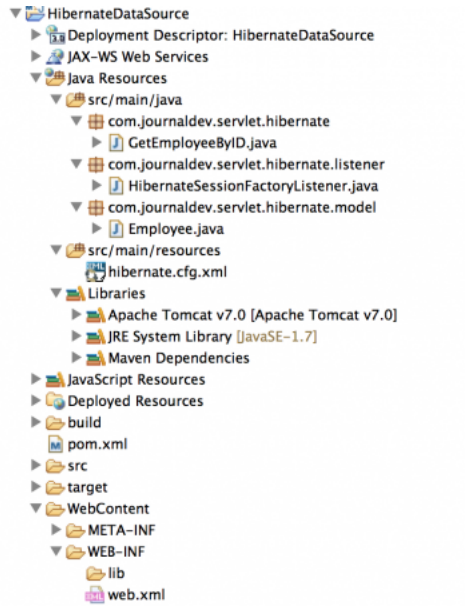
Just restart the server, you should not see any errors in the tomcat server logs. If there are any wrong configurations, such as password is wrong, you will get the corresponding exception in the server log.

You also need to make sure that MySQL driver jar file is inside the tomcat lib directory, otherwise tomcat will not be able to create database connection and you will get `ClassNotFoundException` in logs.

Now our database and tomcat server JNDI setup is ready, let's move to create our web application using hibernate.

Hibernate DataSource Example Dynamic Web Project

Create a dynamic web project in Eclipse and then configure it as Maven project. Our final project structure will look like below image.



Note that I am using **Tomcat-7** for my project deployment and I have added it to the build path, so that we don't need to separately add Servlet API dependencies in our project.

Tomcat-7 supports **Servlet 3 specs** and we will be using annotations to create our servlets. If you are not familiar with Servlet 3 annotations, you should check out Servlet Tutorial for Beginners.

Let's look into each of the components one by one.

Hibernate Maven Dependencies

Our final pom.xml file looks like below.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>HibernateDataSource</groupId>
        <artifactId>HibernateDataSource</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>war</packaging>

        <dependencies>
                <dependency>
                        <groupId>org.hibernate</groupId>
                        <artifactId>hibernate-core</artifactId>
                        <version>4.3.5.Final</version>
                </dependency>
                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-java</artifactId>
                        <version>5.0.5</version>
```

```
                        <scope>provided</scope>
```

I am using Hibernate latest version **4.3.5.Final**, `hibernate-core` dependency is added for Hibernate. `mysql-connector-java` dependency is added because we are using MySQL database, although scope is provided because it's already part of the tomcat container libraries.

Even if we don't add MySQL driver dependencies, our project will compile and run fine. However it's better to include it so that if someone will look into the project dependencies, it will be clear that we are using MySQL database.

Hibernate DataSource Configuration

Our hibernate configuration file with datasource looks like below.

`hibernate.cfg.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
                "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
                "http://hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <property
name="hibernate.connection.datasource">java:comp/env/jdbc/MyLocalDB</property>
        <property name="hibernate.current_session_context_class">thread</property>

        <!-- Mapping with model class containing annotations -->
        <mapping class="com.journaldev.servlet.hibernate.model.Employee"/>
    </session-factory>
</hibernate-configuration>
```

`hibernate.connection.datasource` property is used to provide the DataSource name that will be used by Hibernate for database operations.

Hibernate DataSource Example Model Class

As you can see in hibernate configuration file, we are using annotations in our model class Employee. Our model bean looks like below.

`Employee.java`

```
package com.journaldev.servlet.hibernate.model;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
```

```
import javax.persistence.UniqueConstraint;

@Entity
@Table(name="Employee",
          uniqueConstraints={@UniqueConstraint(columnNames={"ID"})})
public class Employee {

        @Id
        @GeneratedValue(strategy=GenerationType.IDENTITY)
        @Column(name="ID", nullable=false, unique=true, length=11)
        private int id;
```

Model bean is same as we used in Hibernate Beginners Tutorial, you should check it out if you have any confusion related to any of the annotations used.

Hibernate DataSource Tomcat JNDI Servlet Listener

Since we have to initialize Hibernate `SessionFactory` because we can use it in the application and also when web application is destroyed, we need to destroy SessionFactory. So the best place to do this in a `ServletContextListener` implementation.

`HibernateSessionFactoryListener.java`

```
package com.journaldev.servlet.hibernate.listener;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.servlet.annotation.WebListener;

import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;
import org.hibernate.service.ServiceRegistry;
import org.jboss.logging.Logger;

@WebListener
public class HibernateSessionFactoryListener implements ServletContextListener {

        public final Logger logger =
Logger.getLogger(HibernateSessionFactoryListener.class);

    public void contextDestroyed(ServletContextEvent servletContextEvent) {
        SessionFactory sessionFactory = (SessionFactory)
servletContextEvent.getServletContext().getAttribute("SessionFactory");
```

If you are not familiar with servlet listeners, please read Servlet Listener Tutorial.

Hibernate Tomcat JNDI Example Servlet Implementation

Let's write a simple servlet where we will pass employee id as request parameter and it will print out the employee information from database, obviously we will use Hibernate to query the database and get employee information.

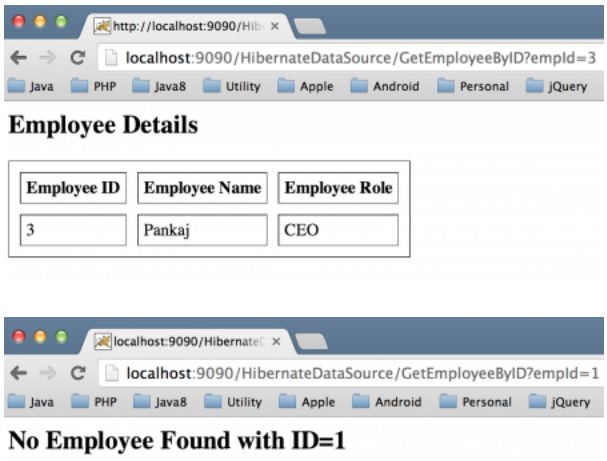
`GetEmployeeByID.java`

```
package com.journaldev.servlet.hibernate;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.jboss.logging.Logger;

import com.journaldev.servlet.hibernate.model.Employee;

@WebServlet("/GetEmployeeByID")
public class GetEmployeeByID extends HttpServlet {
        private static final long serialVersionUID = 1L;
```

It's a very simple servlet class, I am using `@WebServlet` annotation to provide the URI pattern for it.

Testing Hibernate DataSource Tomcat JNDI Example Application

Our application is ready now, just export as war file and deploy it in the tomcat container. Below are some of the screenshots when we invoke our application servlet.



Notice that I am passing **empId** request parameter in the request URL query string. You will also see our application generated logs in the server logs.

```
May 08, 2014 8:14:16 PM org.hibernate.cfg.Configuration configure
INFO: HHH000043: Configuring from resource: hibernate.cfg.xml
May 08, 2014 8:14:16 PM org.hibernate.cfg.Configuration getConfigurationInputStream
INFO: HHH000040: Configuration resource: hibernate.cfg.xml
May 08, 2014 8:14:16 PM org.hibernate.cfg.Configuration doConfigure
INFO: HHH000041: Configured SessionFactory: null
May 08, 2014 8:14:16 PM
```

```
com.journaldev.servlet.hibernate.listener.HibernateSessionFactoryListener
contextInitialized
INFO: Hibernate Configuration created successfully
May 08, 2014 8:14:16 PM
com.journaldev.servlet.hibernate.listener.HibernateSessionFactoryListener
contextInitialized
INFO: ServiceRegistry created successfully
May 08, 2014 8:14:16 PM org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
May 08, 2014 8:14:17 PM org.hibernate.engine.jdbc.internal.LobCreatorBuilder
useContextualLobCreation
INFO: HHH000423: Disabling contextual LOB creation as JDBC driver reported JDBC
version [3] less than 4
May 08, 2014 8:14:17 PM
org.hibernate.engine.transaction.internal.TransactionFactoryInitiator initiateService
```

If you will undeploy the application or stop the server, you will see server logs for destroying the SessionFactory.

```
May 08, 2014 11:31:16 PM
com.journaldev.servlet.hibernate.listener.HibernateSessionFactoryListener
contextDestroyed
INFO: Closing sessionFactory
May 08, 2014 11:31:16 PM
com.journaldev.servlet.hibernate.listener.HibernateSessionFactoryListener
contextDestroyed
INFO: Released Hibernate sessionFactory resource
```

That's all for **Hibernate DataSource example** for tomcat container, I hope it's easy to understand and implement. Download the sample project from below link and play around with it to learn more.

Download Hibernate DataSource Project

« PREVIOUS
[Solved] org.hibernate.HibernateException: No

NEXT »
Hibernate One to One Mapping Example

CurrentSessionContext configured Annotation

About Pankaj

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on **Google Plus**, **Facebook** or **Twitter**. I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **Youtube**.

FILED UNDER: HIBERNATE

Comments

Dhiraj Singh says

FEBRUARY 6, 2017 AT 9:56 PM

Hi Pankaj,
I've been following your tutorials from long time. Well, here in this tutorial I'm getting NPE at line Session session = sessionFactory.getCurrentSession(); in GetCustomerByID class. I'm not able to figure out what's the root cause. Here is the logs:
Log:
——
Feb 07, 2017 11:20:29 AM org.apache.catalina.core.AprLifecycleListener init
INFO: The APR based Apache Tomcat Native library which allows optimal performance in production environments was not found on the java.library.path: /usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib
Feb 07, 2017 11:20:29 AM org.apache.tomcat.util.digester.SetPropertiesRule begin
WARNING: [SetPropertiesRule]{Server/Service/Engine/Host/Context} Setting property 'source' to 'org.eclipse.jst.j2ee.server:HibernateConnectionsusingDataSourceAndJNDITest' did not find a matching property.
Feb 07, 2017 11:20:29 AM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["http-bio-8080"]
Feb 07, 2017 11:20:29 AM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["ajp-bio-8009"]
Feb 07, 2017 11:20:29 AM org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 986 ms
Feb 07, 2017 11:20:29 AM org.apache.catalina.core.StandardService startInternal
INFO: Starting service Catalina
Feb 07, 2017 11:20:29 AM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet Engine: Apache Tomcat/7.0.54
Feb 07, 2017 11:20:30 AM org.apache.catalina.loader.WebappClassLoader validateJarFile
INFO: validateJarFile(/home/dhiraj/MyWorkSpace/workspace/.metadata/.plugins/org.eclipse.wst.server.core/tmp0/wtpwebapps/HibernateCor
INF/lib/javax.servlet-api-3.0.1.jar) – jar not loaded. See Servlet Spec 3.0, section 10.7.2. Offending class: javax/servlet/Servlet.class
Feb 07, 2017 11:20:30 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-8080"]
Feb 07, 2017 11:20:30 AM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
Feb 07, 2017 11:20:30 AM org.apache.catalina.startup.Catalina start

INFO: Server startup in 1348 ms
Feb 07, 2017 11:20:36 AM com.journaldev.servlet.hibernate.GetEmployeeByID doGet
INFO: Request Param empId=3
Feb 07, 2017 11:20:36 AM com.journaldev.servlet.hibernate.GetEmployeeByID doGet
INFO: Here is the session factory object: null
Feb 07, 2017 11:20:36 AM org.apache.catalina.core.StandardWrapperValve invoke
SEVERE: Servlet.service() for servlet [com.journaldev.servlet.hibernate.GetEmployeeByID] in context with path [/HibernateConnectionsusingDataSourceAndJNDITest] threw exception
java.lang.NullPointerException
at com.journaldev.servlet.hibernate.GetEmployeeByID.doGet(GetEmployeeByID.java:35)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:620)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:727)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:303)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:208)
at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:241)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:208)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:220)
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:122)
at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:501)
at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:171)
at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:102)
at org.apache.catalina.valves.AccessLogValve.invoke(AccessLogValve.java:950)
at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:116)
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:408)
at org.apache.coyote.http11.AbstractHttp11Processor.process(AbstractHttp11Processor.java:1040)
at org.apache.coyote.AbstractProtocol$AbstractConnectionHandler.process(AbstractProtocol.java:607)
at org.apache.tomcat.util.net.JIoEndpoint$SocketProcessor.run(JIoEndpoint.java:316)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
at java.lang.Thread.run(Thread.java:745)
Reply

Oleg says
JULY 30, 2016 AT 4:44 AM
You forgot to do
configuration.addAnnotatedClass(Employee.class);
at the contextInitialized method.
Without it we have got exception:
Exception in thread “main” org.hibernate.UnknownEntityTypeException: Unable to locate persister: servlet.hibernate.Employee
Reply

Pankaj says
JULY 30, 2016 AT 7:54 AM
We are loading hibernate configuration file using `configuration.configure("hibernate.cfg.xml");` and mapping class is specified there.
Reply

Oleg says
AUGUST 1, 2016 AT 8:39 AM

it doesn't work
configuration.configure can't read
maybe for hibernate.cfg.xml file must be a special place?
Reply

Pankaj says
AUGUST 2, 2016 AT 12:26 AM
It has to be in the classpath, simple IO concept.
Reply

srini says
JUNE 27, 2016 AT 9:18 AM
Hi Pankaj,
https://www.journaldev.com/3531/spring-mvc-hibernate-mysql-integration-crud-example-tutorial
now i want to use jndi datasource for the above refered link.
the below is the hibernate config file data.
i am confusing how do i change the below configuration if i want to use jndi datasource.
com.journaldev.spring.model.Person
org.hibernate.dialect.MySQLDialect
true
each dao bean id is refering to session factory.
could you please suggest me how do i change this config file.
Reply

Mike says
OCTOBER 29, 2015 AT 2:29 PM
Thank you very much for your tutorials – they are very clearly written.
This is the first one I've had trouble with:
– I couldn't get the java classes to compile – Eclipse could not import javax.servlet classes until I configured the build path and added the apache tomcat 7 library. This made the javac errors go away.
– Then I exported the .war file … but you don't tell us where to put it. By googling around, I read that it should to into the apache-tomcat-7.0/webapps directory, and that by stopping/restarting tomcat, it would uncompress this into a folder.
– However, when I stopped/restarted tomcat *within* Eclipse, nothing happened.
– So I stopped tomcat in Eclipse and then started it from the command line: bin/startup.sh – and this opened the folder into the webapps/ directory.
– Then I loaded a browser but I got a connection error – incorrect database user+password. I had set these in the tomcat conf/server.xml file as you suggested:
<Resource name="jdbc/MyLocalDB"
global="jdbc/MyLocalDB"
username = "…"
– The fix was to edit the hibernate.cfg.xml file inside of webapps/HibernateDataSource/WEB-INF/classes and add in:
root
passwd
And now it works(!) But … I don't know if I am using Hibernate inside of Tomcat the way you designed (e.g., I'm not sure where exactly it is getting the jdbc mysql driver from).

Can you suggest what is right/wrong here ?
Thanks!
-Mike
Reply

Luca says
JANUARY 15, 2015 AT 8:26 AM
Hi, thank you very much for this great tutorial!
I've a couple of questions:
1) Is this setup a valid solution for enterprise web app with big number of users?
2) Is this using connection pooling?
3) You're declaring the jdbc/MyLocalDB resource at server level. Can this be done at web app level (dynamic web project)?
thanks!
Reply

Pankaj says
JANUARY 15, 2015 AT 10:34 AM
Yes to all your questions.
Reply

Luca says
JANUARY 15, 2015 AT 11:07 PM
Hi, thanks for your answer! About point 3: can you show me how? Thanks again!
Reply

Farhan says
MAY 22, 2014 AT 3:53 AM
Hi Pankaj,
I did not understand why have you specified driver class name in hibernate.cfg.xml file? because you already specified driver class name in in Context.xml file for tomcat.
Reply

Josemar Flausino says
MAY 13, 2014 AT 8:17 AM
Hello! Good afternoon!
First of all, sorry for my english! So, My name is Josemar and I'm trying to make a project using Hibernate 4.3.5 / JPA 2.0 plus JNDI DataSource! Well I have already configured persistence.xml and server.xml / context.xml, but I don't know how I can get EntityManagerFactory from jndi. When I try this: "Persistence.createEntityManagerFactory("my_UP")" it doesn't work. Do you have some example about it?
Thank you!
Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Search for tutorials...

DOWNLOAD ANDROID APP

HIBERNATE FRAMEWORK

Hibernate Tutorial

- Hibernate Example
- Hibernate SessionFactory
- Hibernate Session get load
- Hibernate Session save
- HQL Example
- Hibernate Criteria