**Tomcat**

1. **How to store the database credentials in tomcat ? (not in plain text)**

- Tomcat is one of the famous web server used to deploy the webapplication .

- it provides web application manager where we can deploy the application.

- to making use of the this feature , we need to create the account with the role of manager-gui.

- this is done by adding the below lines in conftomcat-users.xml

**<role rolename="manager-gui"/>**

**<user username="admin" password="admin" roles="manager-gui"/>**

Password is stored in plain text. But In real time , the passwords are not to be stored as plain text and it should be encrypted. Tomcat provides an build in script to encrypt the password which is digest.bat available in bin directory. With this script we can specify the encryption algorithm and the text we want to encrypt

|  |  |
| --- | --- |
| **1**  **2**  **3** | **C:tomcatbin>digest.bat -a sha-256 admin**  **admin:8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918**  **C:tomcatbin>** |

In the above code , we are using the sha-256 algorithm for password encryption . This generated encrypted password can be copied to confgtomcat-users.xml file (replace the old password)

Once this is done, we have to tell the tomcat that the password is encrypted. To achieve this , locate the below line in configserver.xml file.

**<Realm className="org.apache.catalina.realm.UserDatabaseRealm"**

**resourceName="UserDatabase" />**

and add the digest attribute as like,

|  |  |
| --- | --- |
| **<Realm className="org.apache.catalina.realm.UserDatabaseRealm"**  **resourceName="UserDatabase" digest=”sha-256”/>** |  |

**Encrypting the database passwords & JNDI Configuration in Tomcat: (Password Encryption and Decryption):**

**Refer :** **http://www.jdev.it/encrypting-passwords-in-tomcat/**

**Step:1**

* JNDI is treated as resource and this is configured in tomcats server.xml file with in the <GlobalNamingResourcesElement>

**<Resource name="jdbc/MyDB" global="jdbc/MyDB" factory="oracle.jdbc.pool.OracleDataSourceFactory"** **auth="Container" type="javax.sql.DataSource" driverClassName="oracle.jdbc.driver.OracleDriver" url="jdbc:oracle:thin:@localhost:1521:orcl" username="pankaj" password="pankaj123" maxActive="100" maxIdle="20" minIdle="5" maxWait="10000"/>**

The JNDI name is “jdbc/MyDB” and its globally refered as with the specified global name “jdbc/MyDB”.

The JDBC driver jar file should be placed in the tomcats lib directory , so that it can be refred with the above configuration.

**Step:2**

* Create the resource link in context.xml file

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

The created JNDI named “jdbc/MyDB” is linked with the name “jdbc/MyLocalDB” and it can be later used in java code to lookup the JNDI Resource.

**Context ctx = new InitialContext();**

**DataSource ds = (DataSource) ctx.lookup("java:/comp/env/jdbc/MyLocalDB");**

**Step 3:**

* Add the below line in web.xml

**<resource-ref>**

**<description>MyDB Connection</description>**

**<res-ref-name> jdbc/MyLocalDB </res-ref-name>**

**<res-type>javax.sql.DataSource</res-type>**

**<res-auth>Container</res-auth>**

**</resource-ref>**

Various encryption algorithms are AES, MD5,SHA-1

* Database password should not be stored as plaintext in tomcat. It should be encrypted while storing in the server.xml file.
* We have to write the utility to encrypt and decrypt the passwords.

**Sample:**

**public** **class** CipherEncryption {

**public** **static** **void** main(String[] args) **throws** NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, IllegalBlockSizeException, BadPaddingException, UnsupportedEncodingException {

// **TODO** Auto-generated method stub

KeyPairGenerator keyPairGenerator=KeyPairGenerator.*getInstance*("RSA");

keyPairGenerator.initialize(2048);

KeyPair keyPair=keyPairGenerator.genKeyPair();

PublicKey publicKey=keyPair.getPublic();

//create the Cipher object for encryption

Cipher cipher=Cipher.*getInstance*("RSA/ECB/PKCS1Padding");

cipher.init(Cipher.***ENCRYPT\_MODE***, publicKey);

String data="Welcome to Tutorialspoint";

cipher.update(data.getBytes());

**byte**[] encrytedData=cipher.doFinal();

System.***out***.println(" data ......."+data);

System.***out***.println("Encrypted data using AES algorithm ......."+**new** String(encrytedData,"UTF-8"));

//initilize the cipher object for decryption

PrivateKey privateKey=keyPair.getPrivate();

cipher.init(Cipher.***DECRYPT\_MODE***, privateKey);

**byte**[] decryptedData=cipher.doFinal(encrytedData);

System.***out***.println(" Decrypted data .........."+**new** String(decryptedData));

}

}

* By using this we can encrypt the passwords that should be stored in server.xml

**<Resource name="jdbc/MyDB" global="jdbc/MyDB" factory="oracle.jdbc.pool.OracleDataSourceFactory"** **auth="Container" type="javax.sql.DataSource" driverClassName="oracle.jdbc.driver.OracleDriver" url="jdbc:oracle:thin:@localhost:1521:orcl" username="pankaj" password="dd30b6e87a972be1999fda04f657eca0" maxActive="100" maxIdle="20" minIdle="5" maxWait="10000"/>**

* Extract this utility as jar and place it in the tomcat lib directory
* Create the custom DatasourceFactory which extends **org.apache.tomcat.jdbc.pool.DataSourceFactory and** customize the datasource creation by overriding the createDataSource() Method . This method create the datasource connection by decrypting the database password.

Ex:

**import** java.io.UnsupportedEncodingException;

**import** java.security.InvalidKeyException;

**import** java.security.NoSuchAlgorithmException;

**import** java.sql.SQLException;

**import** java.util.Properties;

**import** javax.crypto.BadPaddingException;

**import** javax.crypto.IllegalBlockSizeException;

**import** javax.crypto.NoSuchPaddingException;

**import** javax.naming.Context;

**import** javax.sql.DataSource;

**import** org.apache.juli.logging.Log;

**import** org.apache.juli.logging.LogFactory;

**import** org.apache.tomcat.jdbc.pool.DataSourceFactory;

**import** org.apache.tomcat.jdbc.pool.PoolConfiguration;

**import** org.apache.tomcat.jdbc.pool.XADataSource;

**public** **class** EncryptedDataSourceFactory **extends** DataSourceFactory {

**private** **static** **final** Log ***log*** = LogFactory.getLog(EncryptedDataSourceFactory.**class**);

**private** Encryptor encryptor = **null**;

**public** EncryptedDataSourceFactory() {

**try** {

encryptor = **new** Encryptor(); // If you've used your own secret key, pass it in...

} **catch** (InvalidKeyException | NoSuchAlgorithmException | NoSuchPaddingException | UnsupportedEncodingException e) {

***log***.fatal("Error instantiating decryption class.", e);

**throw** **new** RuntimeException(e);

}

}

@Override

**public** DataSource createDataSource(Properties properties, Context context, **boolean** XA) **throws** InvalidKeyException,

IllegalBlockSizeException, BadPaddingException, SQLException, NoSuchAlgorithmException,

NoSuchPaddingException {

// Here we decrypt our password.

PoolConfiguration poolProperties = EncryptedDataSourceFactory.parsePoolProperties(properties);

poolProperties.setPassword(encryptor.decrypt(poolProperties.getPassword()));

// The rest of the code is copied from Tomcat's DataSourceFactory.

**if** (poolProperties.getDataSourceJNDI() != **null** && poolProperties.getDataSource() == **null**) {

performJNDILookup(context, poolProperties);

}

org.apache.tomcat.jdbc.pool.DataSource dataSource = XA ? **new** XADataSource(poolProperties)

: **new** org.apache.tomcat.jdbc.pool.DataSource(poolProperties);

dataSource.createPool();

**return** dataSource;

}

}

* Once we created the custom datasource to decrypt the password, we should intimate the tomcat to use the this class for datasource creation. To do so , add the below line in server.xml file.

**<Resource name="jdbc/MyDB" global="jdbc/MyDB" factory="nl.wimvanhaaren.tomcat.secured.EncryptedDataSourceFactory" auth="Container" type="javax.sql.DataSource" driverClassName="oracle.jdbc.driver.OracleDriver" url="jdbc:oracle:thin:@localhost:1521:orcl" username="pankaj" password="dd30b6e87a972be1999fda04f657eca0" maxActive="100" maxIdle="20" minIdle="5" maxWait="10000"/>**