**JASYPT HOW TO**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| f you use a database for your ticket registries or an LDAP for user authentication, etc, it's easy to end up with clear text credentials in your configuration or properties files. This tutorial shows how to use Jasypt to replace those values with encrypted strings that get decrypted at run time.  A case in point is the username and password in this deployerConfigContext.xml bean definition for a database:   |  | | --- | | <bean id="dataSource"    class="org.apache.commons.dbcp.BasicDataSource"    p:driverClassName="org.postgresql.Driver"    p:url="jdbc:postgresql://localhost:5432/cas\_sso?autoReconnect=true"    p:password="password"    p:username="postgres"  /> |   An open source product called [Java Simplified Encryption (JASYPT)](http://jasypt.org/index.html) allows you to replace clear text passwords in files with encrypted strings that are decrypted at run time. The following shows how this can be done. Thank you to Ulrick Sandberg whose blog, [Encrypting Properties with Jasypt](http://blog.jayway.com/2008/12/09/encrypting-properties-with-jasypt/), helped me a lot.  Jasypt can be integrated into the Spring configuration framework so that property values are decrypted as the configuration file is loaded. Spring uses the PropertyPlaceholder framework to replace tokens with values from a properties file, and Jasypt's approach replaces the PropertyPlaceholderConfigurer class with one that recognizes encrypted strings and decrypts them.  The approach described here uses password-based encryption, which means that the system still needs a secret password in order to decrypt our credentials. We don't want to simply move the secret from one file to another, and Jasypt avoids that by passing the key as an environment variable or even directly to the application through a web interface each time it is deployed. See [this Jasypt page](http://jasypt.org/webconfiguration.html) for more information on the latter.  **Add a dependency on Jasypt**  In your overlay pom, add a dependency on Jasypt. Version 1.7 is available at the time of this writing. If your site uses Java 5 or earlier, you'll also need a dependency on [ICU4j](http://site.icu-project.org/). See [this Jasypt page](http://jasypt.org/maven.html) for details.  <dependency>  <groupId>org.jasypt</groupId>  <artifactId>jasypt</artifactId>  <version>{version}</version>  <scope>compile</scope>  </dependency>  **Move your credentials into a properties file**  Modify the configuration file to reference the database credentials in your properties file using standard token replacement syntax:   |  | | --- | | <bean id="dataSource"    class="org.apache.commons.dbcp.BasicDataSource"    p:driverClassName="org.postgresql.Driver"    p:url="jdbc:postgresql://localhost:5432/cas\_sso?autoReconnect=true"    p:password="${dataSource.password}"    p:username="${dataSource.username}"  /> |   Create the corresponding entries in the properties file.   |  | | --- | | dataSource.password=password  dataSource.username=postgres |   These values will, of course, be encrypted later - postponing that step allows you to test your Jasypt integration without dealing with encrypting and decrypting yet.  **Declare the Jasypt beans**  The Jasypt token processor needs to know which properties file to work on, as well as which encryptor and encryption algorithm to use. Declare the following beans in your configuration file:   |  | | --- | | <bean id="propertyPlaceholderConfigurer"        class="org.jasypt.spring.properties.EncryptablePropertyPlaceholderConfigurer">    <constructor-arg ref="configurationEncryptor" />    <property name="location" value="/WEB-INF/cas.properties" />  </bean>    <bean id="configurationEncryptor" class="org.jasypt.encryption.pbe.StandardPBEStringEncryptor">    <property name="config" ref="environmentVariablesConfiguration" />  </bean>    <bean id="environmentVariablesConfiguration"        class="org.jasypt.encryption.pbe.config.EnvironmentStringPBEConfig">    <property name="algorithm" value="PBEWithMD5AndDES" />    <property name="passwordEnvName" value="CAS\_PBE\_PASSWORD" />    <!--    <property name="password" value="actual password here for testing ONLY" />    -->  </bean> |   The example above specifies that our tokens can be found in cas.properties, and that the PBE password will be passed to the application in an environment variable named CAS\_PBE\_PASSWORD. Note that you could set a clear text password value, and that might be handy for testing.  If you deploy and run your CAS server with only the changes outlined above it should run normally. This will verify that your configuration is consistent, and Jasypt is getting invoked and working with your properties file correctly.  **Encrypt your credentials**  Jasypt provides a command line utility that can be used to encrypt the values of your properties. Download the [Jasypt distribution](http://sourceforge.net/projects/jasypt/files/) and unpack it. The utilities reside in the **bin** directory.  Encrypting the database user's name is as simple as entering it and the PBE password:  C:\CAS SSO\jasypt-1.7\bin> encrypt input=postgres password=secret  ----ENVIRONMENT-----------------  Runtime: Sun Microsystems Inc. Java HotSpot(TM) Client VM 17.1-b03  ----ARGUMENTS-------------------  input: postgres  password: secret  ----OUTPUT----------------------  jd5ZREpBqxuN9ok0IhnXabgw7V3EoG2p  Cut-n-paste the output value into your properties file.  **There are two important points to remember:**  1. Tell Jasypt that the value is encrypted by wrapping it in the **ENC** "function" -   |  | | --- | | dataSource.password=ENC(1oqwimyLNS+hRJ240Rhkgm9FYZudI5qs)  dataSource.username=ENC(jd5ZREpBqxuN9ok0IhnXabgw7V3EoG2p) |   2. Use the same PBE password for all encrypted values.  **Pass the PBE Password to the Application**  There are several ways to do this. You could modify your CAS startup script to prompt the user for the password and export it as an environment variable. Or you could use [the Jasypt web configuration method](http://jasypt.org/webconfiguration.html) mentioned above.  This can be used to collect user input in a Windows batch file:   |  | | --- | | @echo off  SET /P CAS\_PBE\_PASSWORD=[Enter the CAS configuration password] | |

* No labels