WHITESTEIN
Technologies

Living Systems® Process Suite

Server Installation and Upgrade

Living Systems Process Suite Documentation

3.2
Tue Jan 12 2021

Whitestein Technologies AG | Hinterbergstrasse 20 | CH-6330 Cham
Tel +41 44-256-5000 | Fax +41 44-256-5001 | http://www.whitestein.com

Contents

Chapter 1

Server Installation and Upgrade

Before you perform any of the actions, please, make sure to meet the requirements.

This guide contains information on how to set up a new LSPS environment and how to upgrade an existing environment.

Chapter 2

Requirements

Before you set up an environment for LSPS, make sure it meets the following requirements:

- *Hardware requirements for 50 users, and about 10 concurrent sessions*:
  - **the server hosting an application server with the LSPS, your Application User Interface and models**:
    - *Minimum*: i7 x86, 64bit, 8 core CPU. 12 GB RAM
    - *Recommended*: i7 x86, 64bit, 8 core CPU. 24 GB RAM
- a **dedicated database server**:
  - *Minimum*: i7 x86, 64bit, 8 core CPU. 16 GB RAM
  - *Recommended*: i7 x86, 64bit, 16 core CPU. 32 GB RAM
- *LSPS Application environment*:
  - supported Java SDK: JVM should be run with at least `-Xmx1024m`
    - Oracle JDK 1.7.x
    - Oracle JDK 1.8.x if supported by your application server

      Important: Make sure to purchase the appropriate licenses for your JDKs on servers.
  - supported databases:
    - MySQL 5.6
    - MS SQL 2008 R2
    - MS SQL 2014
    - DB2 10.5
    - Oracle 11g R2
  - supported application servers:
    - JBoss EAP 6.4.4
    - JBoss EAP 7.0.0
    - WildFly 9.0.2
    - WildFly 10.0.0
    - WildFly 11.0.0
    - WebSphere 8.5.0.2
    - Weblogic 12.1.1

      Important: Make sure that your server has all the necessary security vulnerabilities fixed.
  - supported web browsers (for Application User Interface and Management Console):
    - Google Chrome 23

    * Mozilla Firefox 17
    * Internet Explorer 9, 10, 11
    * Safari 10

Consider setting up the following environments:

- development environment: environment for deployment of the latest application; This is intended only for development purposes;
- pre-production environment: clone of the production environment for testing;
- production environment: production environment with application that passed testing;

The resources and tools for setup and deployment of the LSPS Application on an application server are delivered as part of the *Process Runtime Suite* in the `lsps-deliverable-<VERSION>-lsps-runtime.zip` file. Apart from the deployable LSPS Application ear, the suite contains the command-line tool for management of the server as well as database migration tools.

Chapter 3

Installation

Before you start setting up the environment, make sure it meets the requirements, you have selected a compatible combination of java, database, application server, and LSPS Application, and purchased any applicable licenses, such as, license for Oracle JDK.

To set up a new LSPS server with your LSPS Application, do the following:

1. Create and initialize the system database.
2. Set up an application server.

3.1 Setting up the Database

LSPS requires an lsps database with a dedicated user to store its system data. To set up such a database, do the following:

1. Make sure the database is supported by your application server.
2. Create a database with the character encoding to UTF-8.
   Example MySQL database setup with UTF-8

```
 CREATE DATABASE lsps DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8_general_ci;
GRANT ALL PRIVILEGES ON lsps.* TO lsps@'' IDENTIFIED BY 'lsps';
FLUSH PRIVILEGES;
```

3. Initialize the database with the db-migration tool from the runtime suite: `./migrate.sh databaseUrl:<DATABASE_URL> user:<DB_USER> password:<ESCAPED_DB_PASSWD>` (for details, refer to migratescriptparams).
4. Configure your database:
   - Microsoft SQL server:
     - Enable the XA transaction support. Refer to the relevant Microsoft documentation for information on your version of Microsoft SQL server and JDBC driver; for example, for Windows Server 2003 and sqljdbc4.jar, refer to http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774 and http://msdn.microsoft.com/en-US/library/aa342335(v=sql.90).
     - It might be necessary to install MSDTC in Windows (Add/remove Windows components -> Application Server -> Details -> Enable network DTC access).
   - On MySQL, set the allowed packet size to 512 MBs and the default time zone in the configuration file of your mysql server, typically, the `my.cnf` or `mysqld.conf` files:

```
[mysqld]
 max_allowed_packet=512M
 default_time_zone='+0:00'
```

3.2 Setting up the Application Server

You can deploy LSPS on the following application servers:

- JBoss
- WildFly
- Oracle WebLogic
- IBM WebSphere

Make sure the selected application server supports your database.

3.2.1 JBoss

You can use JBoss Enterprise Application Server 7 or JBoss Enterprise Application Server 6.4.

3.2.1.1 Setting up JBoss Enterprise Application Platform 7.0.0 with LSPS Application

> **Important:** It is strongly discouraged to run other applications on JBoss Enterprise Application Platform (EAP) with LSPS since LSPS is using a customized Hibernate.

To set up JBoss EAP 7.0.0 with LSPS, do the following:

1. Install JBoss EAP.
2. Create JBoss module with the LSPS login module:
   (a) Copy lsps-security-jboss-${lsps.version}.jar to $EAP_HOME/modules/com/whitestein/lsps/security/main.
   (b) Create the file $EAP_HOME/modules/com/whitestein/lsps/security/main/module.xml with the following content:

```
<module xmlns="urn:jboss:module:1.0" name="com.whitestein.lsps.security">
 <resources>
   <resource-root path="lsps-security-jboss-${lsps.version}.jar"/>
 </resources>
 <dependencies>
   <module name="javax.api"/>
   <module name="javax.transaction.api"/>
   <module name="org.picketbox" />
 </dependencies>
</module>
```

   Take a note of the module name since it is used later for security realm configuration.
3. Create a JBoss module with your JDBC driver. Type IV JDBC drivers are recommended.
   - For MySQL you need to copy MySQL Connector/J 5.5 or later, that is, *mysql-connector-java-<VERSION>-bin.jar*, to $WF_HOME/modules/com/mysql/main. Then create $WF_HOME/modules/com/mysql/main/module.xml with following content:

```
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
 <resources>
   <resource-root path="mysql-connector-java-<VERSION>-bin.jar"/>
 </resources>
 <dependencies>
   <module name="javax.api"/>
   <module name="javax.transaction.api"/>
 </dependencies>
</module>
```

- For Microsoft SQL server, get the Microsoft SQL driver *sqljdbc4.jar* and copy it to $EAP_HO↩ ME/modules/com/microsoft/sqlserver/mainr.
  Also consider adding the `sendStringParametersAsUnicode=false` property.

```
<module xmlns="urn:jboss:module:1.0" name="com.microsoft.sqlserver">
 <resources>
   <resource-root path="sqljdbc4.jar"/>
 </resources>
 <dependencies>
   <module name="javax.api"/>
   <module name="javax.transaction.api"/>
 </dependencies>
</module>
```

- For Oracle 11g, copy ojdbc6.jar to $EAP_HOME/modules/com/oracle/jdbc/main

```
<module xmlns="urn:jboss:module:1.0" name="com.oracle.jdbc">
 <resources>
   <resource-root path="ojdbc6.jar"/>
 </resources>
 <dependencies>
   <module name="javax.api"/>
   <module name="javax.transaction.api"/>
 </dependencies>
</module>
```

The module name will be used later for creating data source.

4. Configure JMS in the server profile file (typically, $EAP_HOME/standalone/configuration/standalone-full.xml):

   (a) Set JMS to be persistent: add `<persistence-enabled>true</persistence-enabled>`.

   (b) Add queue and topic under messaging-activemq -> server:

```
<jms-queue name="LSPS_QUEUE">
 <entry name="java:jboss/jms/LSPS_QUEUE"/>
</jms-queue>
<jms-topic name="LSPS_TOPIC">
  <entry name="java:jboss/jms/LSPS_TOPIC"/>
</jms-topic>
```

   LSPS will use the default JMS connection factory.

5. Configure your data source in the server profile file (typically, $EAP_HOME/standalone/configuration/standalone-full.xml).

   The data source must be an XA-datasource with the transaction isolation TRANSACTION_READ_COMM↩ ITTED and the JNDI name must be *jdbc/LSPS_DS*.

   - For MySql, add the following under the data sources subsystem:

```
<xa-datasource jndi-name="java:/jdbc/LSPS_DS" pool-name="LSPS_DS" enabled="true" use-jav
 <driver>mysqlxa</driver>
 <xa-datasource-property name="URL">jdbc:mysql://localhost:3306/lsps?useUnicode=true&amp
 <security>
   <user-name>lsps</user-name>
   <password>lsps</password>
 </security>
```

```
    <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
    <xa-pool>
      <min-pool-size>10</min-pool-size>
      <max-pool-size>20</max-pool-size>
      <prefill>true</prefill>
    </xa-pool>
  </xa-datasource>
```

- For Microsoft SQL server, add the following under the data sources subsystem:

```
<xa-datasource jndi-name="java:/jdbc/LSPS_DS" pool-name="LSPS_DS" enabled="true" use-jav
   <driver>mssqlxa</driver>
   <xa-datasource-property name="URL">jdbc:sqlserver://localhost;databaseName=lsps</xa-d
   <security>
     <user-name>lsps</user-name>
     <password>lsps</password>
   </security>
   <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
   <xa-pool>
     <min-pool-size>10</min-pool-size>
     <max-pool-size>20</max-pool-size>
     <prefill>true</prefill>
   </xa-pool>
   <new-connection-sql>select 1</new-connection-sql>
   <validation>
     <check-valid-connection-sql>select 1</check-valid-connection-sql>
   </validation>
 </xa-datasource>
```

- For for Oracle 11g XE server, add the following under data sources subsystem:

```
<xa-datasource jndi-name="java:/jdbc/LSPS_DS" pool-name="LSPS_DS" enabled="true" use-jav
   <driver>oraclexa</driver>
   <xa-datasource-property name="URL">jdbc:oracle:thin:@localhost:1521:XE</xa-datasource
   <security>
     <user-name>lsps</user-name>
     <password>lsps</password>
   </security>
   <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
   <xa-pool>
     <min-pool-size>10</min-pool-size>
     <max-pool-size>20</max-pool-size>
     <prefill>true</prefill>
   </xa-pool>
   <validation>
     <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle
     <stale-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle
     <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleE
   </validation>
 </xa-datasource>
```

6. In the `<drivers>` element, make sure to delete the h2 driver (driver with the `name="h2"` attribute) and add `<driver>` referred to from the data source definition:

   - for MySql:

```
<driver name="mysqlxa" module="com.mysql">
   <xa-datasource-class>com.mysql.cj.jdbc.MysqlXADataSource</xa-datasource-class>
 </driver>
```

   - for MSSql:

```
<driver name="mssqlxa" module="com.microsoft.sqlserver">
   <xa-datasource-class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-datasourc
 </driver>
```

   - for Oracle:

```
    <driver name="oraclexa" module="com.oracle.jdbc">
       <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
     </driver>
```

- Delete driver with name="h2".

7. Set Infinispan acquire-timeout setting to at least 5 minutes in your profile XML, such as standalone-full.xml. For details refer to the respective faq entry

```
 <local-cache name="passivation">
        <locking isolation="REPEATABLE_READ" acquire-timeout="600000"/>
        <transaction mode="BATCH"/>
        <file-store passivation="true" purge="false"/>
   </local-cache>
</cache-container>
```

8. Consider adding `logging.properties` for better log legibility:

```
 # Specify the handlers to create in the root logger
# (all loggers are children of the root logger)
# The following creates two handlers
handlers = java.util.logging.ConsoleHandler, java.util.logging.FileHandler
~
# Set the default logging level for the root logger
.level = WARN
~
# Set the default logging level for new ConsoleHandler instances
java.util.logging.ConsoleHandler.level = ALL
~
# Set the default logging level for new FileHandler instances
java.util.logging.FileHandler.level = ALL
~
# Set the default formatter for new ConsoleHandler instances
java.util.logging.ConsoleHandler.formatter = com.whitestein.lsps.log.LSPSFormatter
~
# Set the default formatter for new FileHandler instances
java.util.logging.FileHandler.formatter = com.whitestein.lsps.log.LSPSFormatter
~
# Set the logging level for the LSPS
com.whitestein.lsps.level = FINER
```

9. Create a mail session with the JNDI name mail/LSPS_MAIL defined in $EAP_HOME/standalone/configuration/standalone-full.xml or the respective JBoss config under the mail subsystem tag.

```
 <mail-session name="lspsmail" jndi-name="java:jboss/mail/LSPS_MAIL">
    <smtp-server outbound-socket-binding-ref="mail-smtp"/>
  </mail-session>
```

   Important: If you are using Microsoft Exchange Server as your mail server, make sure you have enabled the smtp feature.

10. Set up the login module for the LSPS realm: add the following to the "security-domains" of the security subsystem tag in $EAP_HOME/standalone/configuration/standalone-full.xml or the respective JBoss config referencing the security module from step 2:

```
 <security-domain name="lspsRealm" cache-type="default">
  <authentication>
    <login-module code="com.whitestein.lsps.security.jboss.LSPSRealm" flag="required" module=
    <module-option name="dsJndiName" value="/jdbc/LSPS_DS"/>
    </login-module>
  </authentication>
</security-domain>
```

The attribute cache-type of the security-domain element turns on caching of user login data. As a side effect, changes to user credentials will not be used until after the cache is flushed; for example, if a user changes user's password through the web console but the user is already using PDS, they will be able to connect PDS to the server with the old credentials. You can turn the cache off by omitting the cache-type attribute of the security-domain element. The security domain definition will look as follows:

```
 <security-domain name="lspsRealm">
  <authentication>
    <login-module code="com.whitestein.lsps.security.jboss.LSPSRealm" flag="required" module=
    <module-option name="dsJndiName" value="/jdbc/LSPS_DS"/>
    </login-module>
  </authentication>
</security-domain>
```

For additional module options, refer to the JavaDoc of the implementing class and super classes. You can also plug in your own authentication module.

11. Configure bean pools.

    The application requires configuration of bean and data source connection pools. The configuration depends on the expected load as well as other performance parameters. You can change the configuration in the $EAP_HOME/standalone/configuration/standalone-full.xml or the respective JBoss config.

    - Limit the MDB pool and the default pool used for timers, async, and remoting.
    - Extend the SLSB pool to 10 $*$ (MDB pool size + default pool size) at least .
    - Decrease the acquisition timeouts to reasonable values so that in case of lack of resources long waiting deadlocks do not occur. Set data-source connection pool maximum to MDB pool size + default pool size at least. For example:

      ```
       <bean-instance-pools>
        <!-- A sample strict max pool configuration -->
        <strict-max-pool name="slsb-strict-max-pool" max-pool-size="200" instance-acquisition-t
        <strict-max-pool name="mdb-strict-max-pool" max-pool-size="20" instance-acquisition-tim
      ~
        <thread-pools>
          <thread-pool name="default">
            <max-threads count="10"/>
            ...
          </thread-pool>
        </thread-pools>
      ```

12. Configure networking (optional).

    By default JBoss binds to the local host. You can change the configuration in $EAP_HO↩ ME/standalone/configuration/standalone-full.xml (or the JBoss config you use). For example, to bind JBoss to any IPv4 address use the following setting:

    ```
     <interface name="management">
      <any-address/>
    </interface>
    <interface name="public">
      <any-address/>
    </interface>
    <interface name="unsecure">
      <any-address/>
    </interface>
    ```

13. Set up the JBoss option file in $EAP_HOME/bin/standalone.conf (or the JBoss config you use).

    (a) Set up Java memory options:

    - minimum setting: JAVA_OPTS="-Xms128m -Xmx512m -XX:MetaspaceSize=96M -XX:Max↩ MetaspaceSize=256m"
    - recommended setting: JAVA_OPTS="-Xms256m -Xmx1024m -XX:MetaspaceSize=96M -XX:↩ MaxMetaspaceSize=256m"

JVM might require larger amount of memory depending on the uploaded and used LSPS modules.

(b) Set up the server startup configuration file to use (pointing to the configuration file you modified), e.g. for standalone-full.xml:

```
JAVA_OPTS="$JAVA_OPTS -Djboss.server.default.config=standalone-full.xml"
```

(c) Add the following Java system properties to the config file, so it is added to the Java options passed to the JVM:

i. Configure EMF framework to work properly on JBoss:

```
JAVA_OPTS="$JAVA_OPTS -Dorg.eclipse.emf.ecore.EPackage.Registry.INSTANCE=org.eclipse
```

ii. Configure the conversion of empty numeric inputs:

```
JAVA_OPTS="-Dorg.apache.el.parser.COERCE_TO_ZERO=false $JAVA_OPTS"
```

LSPS needs the system property org.apache.el.parser.COERCE_TO_ZERO to be set to "false" (see https://jsp-spec-public.dev.java.net/issues/show_bug.cgi?id=183). Without the property, empty input in numeric fields in the web application is interpreted as 0. This interpretation causes some functionalities not to work.

14. Copy the LSPS Application EAR to $JB_HOME/standalone/deployments.

15. Optional: Hide useless vaadin warning from the log:

```
<logger category="com.vaadin.ui.ConnectorTracker" use-parent-handlers="true">
  <level name="ERROR"/>
</logger>
```

3.2.1.2 Setting up JBoss Enterprise Application Platform 6.4.0 with LSPS Application

Important: It is strongly discouraged to run other applications on JBoss Enterprise Application Platform (EAP) with LSPS since LSPS is using a customized Hibernate.

To set up JBoss EAP 6.4.0 with LSPS, do the following:

1. Install JBoss EAP.

2. Create JBoss module with the LSPS login module:

(a) Copy lsps-security-jboss-${lsps.version}.jar to $EAP_HOME/modules/com/whitestein/lsps/security/main.

(b) Create the file $EAP_HOME/modules/com/whitestein/lsps/security/main/module.xml with the following content:

```
<module xmlns="urn:jboss:module:1.0" name="com.whitestein.lsps.security">
 <resources>
   <resource-root path="lsps-security-jboss-${lsps.version}.jar"/>
 </resources>
 <dependencies>
   <module name="javax.api"/>
   <module name="javax.transaction.api"/>
   <module name="org.picketbox" />
 </dependencies>
</module>
```

Take a note of the module name since it is used later for security realm configuration.

3. Create a JBoss module with your JDBC driver. Type IV JDBC drivers are recommended.

- For MySQL you need to copy MySQL Connector/J 5.5 or later, that is, *mysql-connector-java-<VERSION>-bin.jar*, to $WF_HOME/modules/com/mysql/main. Then create $WF_HOME/modules/com/mysql/main/module.xml with following content:

```
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
 <resources>
   <resource-root path="mysql-connector-java-<VERSION>-bin.jar"/>
 </resources>
 <dependencies>
   <module name="javax.api"/>
   <module name="javax.transaction.api"/>
 </dependencies>
</module>
```

- For Microsoft SQL server, get the Microsoft SQL driver *sqljdbc4.jar* and copy it to $EAP_HO↩ ME/modules/com/microsoft/sqlserver/mainr.
  Also consider adding the `sendStringParametersAsUnicode=false` property.

```
<module xmlns="urn:jboss:module:1.0" name="com.microsoft.sqlserver">
 <resources>
   <resource-root path="sqljdbc4.jar"/>
 </resources>
 <dependencies>
   <module name="javax.api"/>
   <module name="javax.transaction.api"/>
 </dependencies>
</module>
```

- For Oracle 11g, copy ojdbc6.jar to $EAP_HOME/modules/com/oracle/jdbc/main

```
<module xmlns="urn:jboss:module:1.0" name="com.oracle.jdbc">
   <resources>
     <resource-root path="ojdbc6.jar"/>
   </resources>
   <dependencies>
     <module name="javax.api"/>
     <module name="javax.transaction.api"/>
   </dependencies>
 </module>
```

The module name will be used later for creating data source.

4. Configure JMS in the server profile file (typically, $EAP_HOME/standalone/configuration/standalone-full.xml):

   (a) Set JMS to be persistent: `<persistence-enabled>true</persistence-enabled>`

   (b) Add queue and topic under jms-destinations:

```
<jms-queue name="LSPS_QUEUE">
 <entry name="java:jboss/jms/LSPS_QUEUE"/>
</jms-queue>
<jms-topic name="LSPS_TOPIC">
  <entry name="java:jboss/jms/LSPS_TOPIC"/>
</jms-topic>
```

   LSPS will use the default JMS connection factory.

5. Configure your data source in the server profile file (typically, $EAP_HOME/standalone/configuration/standalone-full.xml).

   The data source must be an XA-datasource with the transaction isolation TRANSACTION_READ_COMM↩ ITTED and the JNDI name must be *jdbc/LSPS_DS*.

   - For MySql, add the following under the data sources subsystem:

```
<xa-datasource jndi-name="java:/jdbc/LSPS_DS" pool-name="LSPS_DS" enabled="true" use-jav
 <driver>mysqlxa</driver>
 <xa-datasource-property name="URL">jdbc:mysql://localhost:3306/lsps?useUnicode=true&amp
 <security>
   <user-name>lsps</user-name>
   <password>lsps</password>
```

```
    </security>
    <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
    <xa-pool>
      <min-pool-size>10</min-pool-size>
      <max-pool-size>20</max-pool-size>
      <prefill>true</prefill>
    </xa-pool>
  </xa-datasource>
```

- For Microsoft SQL server, add the following under the data sources subsystem:

```
<xa-datasource jndi-name="java:/jdbc/LSPS_DS" pool-name="LSPS_DS" enabled="true" use-jav
   <driver>mssqlxa</driver>
   <xa-datasource-property name="URL">jdbc:sqlserver://localhost;databaseName=lsps</xa-d
   <security>
     <user-name>lsps</user-name>
     <password>lsps</password>
   </security>
   <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
   <xa-pool>
     <min-pool-size>10</min-pool-size>
     <max-pool-size>20</max-pool-size>
     <prefill>true</prefill>
   </xa-pool>
   <new-connection-sql>select 1</new-connection-sql>
   <validation>
     <check-valid-connection-sql>select 1</check-valid-connection-sql>
   </validation>
 </xa-datasource>
```

- For for Oracle 11g XE server, add the following under data sources subsystem:

```
<xa-datasource jndi-name="java:/jdbc/LSPS_DS" pool-name="LSPS_DS" enabled="true" use-jav
   <driver>oraclexa</driver>
   <xa-datasource-property name="URL">jdbc:oracle:thin:@localhost:1521:XE</xa-datasource
   <security>
     <user-name>lsps</user-name>
     <password>lsps</password>
   </security>
   <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
   <xa-pool>
     <min-pool-size>10</min-pool-size>
     <max-pool-size>20</max-pool-size>
     <prefill>true</prefill>
   </xa-pool>
   <validation>
     <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle
     <stale-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle
     <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleE
   </validation>
 </xa-datasource>
```

6. In the `<drivers>` element, make sure to delete the h2 driver (driver with the `name="h2"` attribute) and add `<driver>` referred to from the datasource definition:

   - for MySql:

```
<driver name="mysqlxa" module="com.mysql">
   <xa-datasource-class>com.mysql.cj.jdbc.MysqlXADataSource</xa-datasource-class>
 </driver>
```

   - for MSSql:

```
<driver name="mssqlxa" module="com.microsoft.sqlserver">
   <xa-datasource-class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-datasourc
 </driver>
```

   - for Oracle:

```
      <driver name="oraclexa" module="com.oracle.jdbc">
         <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
       </driver>
```

- Delete driver with name="h2".

7. Set Infinispan acquire-timeout setting to at least 5 minutes in your profile XML, such as standalone-full.xml. For details refer to the respective faq entry

```
 <local-cache name="passivation">
        <locking isolation="REPEATABLE_READ" acquire-timeout="600000"/>
        <transaction mode="BATCH"/>
        <file-store passivation="true" purge="false"/>
    </local-cache>
</cache-container>
```

8. Consider adding `logging.properties` for better log legibility:

```
 # Specify the handlers to create in the root logger
# (all loggers are children of the root logger)
# The following creates two handlers
handlers = java.util.logging.ConsoleHandler, java.util.logging.FileHandler
~
# Set the default logging level for the root logger
.level = WARN
~
# Set the default logging level for new ConsoleHandler instances
java.util.logging.ConsoleHandler.level = ALL
~
# Set the default logging level for new FileHandler instances
java.util.logging.FileHandler.level = ALL
~
# Set the default formatter for new ConsoleHandler instances
java.util.logging.ConsoleHandler.formatter = com.whitestein.lsps.log.LSPSFormatter
~
# Set the default formatter for new FileHandler instances
java.util.logging.FileHandler.formatter = com.whitestein.lsps.log.LSPSFormatter
~
# Set the logging level for the LSPS
com.whitestein.lsps.level = FINER
```

9. Create a mail session with the JNDI name mail/LSPS_MAIL defined in $EAP_HOME/standalone/configuration/standalone-full.xml or the respective JBoss config under the mail subsystem tag.

```
 <mail-session name="lspsmail" jndi-name="java:jboss/mail/LSPS_MAIL">
  <smtp-server outbound-socket-binding-ref="mail-smtp"/>
</mail-session>
```

   Important: If you are using Microsoft Exchange Server as your mail server, make sure you have enabled the smtp feature.

10. Set up the login module for the LSPS realm: add the following to the "security-domains" of the security subsystem tag in $EAP_HOME/standalone/configuration/standalone-full.xml or the respective JBoss config referencing the security module from step 2:

```
 <security-domain name="lspsRealm" cache-type="default">
  <authentication>
    <login-module code="com.whitestein.lsps.security.jboss.LSPSRealm" flag="required" module=
    <module-option name="dsJndiName" value="/jdbc/LSPS_DS"/>
    </login-module>
  </authentication>
</security-domain>
```

The attribute cache-type of the security-domain element turns on caching of user login data. As a side effect, changes to user credentials will not be used until after the cache is flushed; for example, if a user changes user's password through the web console but the user is already using PDS, they will be able to connect PDS to the server with the old credentials. You can turn the cache off by omitting the cache-type attribute of the security-domain element. The security domain definition will look as follows:

```
 <security-domain name="lspsRealm">
  <authentication>
    <login-module code="com.whitestein.lsps.security.jboss.LSPSRealm" flag="required" module=
    <module-option name="dsJndiName" value="/jdbc/LSPS_DS"/>
    </login-module>
  </authentication>
</security-domain>
```

For additional module options, refer to the JavaDoc of the implementing class and super classes. You can also plug in your own authentication module.

11. Configure bean pools.

    The application requires configuration of bean and data source connection pools. The configuration depends on the expected load as well as other performance parameters. You can change the configuration in the $EAP_HOME/standalone/configuration/standalone-full.xml or the respective JBoss config.

    - Limit the MDB pool and the default pool used for timers, async, and remoting.
    - Extend the SLSB pool to 10 $*$ (MDB pool size + default pool size) at least .
    - Decrease the acquisition timeouts to reasonable values so that in case of lack of resources long waiting deadlocks do not occur. Set data-source connection pool maximum to MDB pool size + default pool size at least. For example:

      ```
       <bean-instance-pools>
        <!-- A sample strict max pool configuration -->
        <strict-max-pool name="slsb-strict-max-pool" max-pool-size="200" instance-acquisition-t
        <strict-max-pool name="mdb-strict-max-pool" max-pool-size="20" instance-acquisition-tim
      </bean-instance-pools>
      ~
      <thread-pools>
        <thread-pool name="default">
          <max-threads count="10"/>
            ...
        </thread-pool>
      </thread-pools>
      ```

12. Configure networking (optional).

    By default JBoss binds to the local host. You can change the configuration in $EAP_HO↩ ME/standalone/configuration/standalone-full.xml (or the JBoss config you use). For example, to bind JBoss to any IPv4 address use the following setting:

```
 <interface name="management">
  <any-address/>
</interface>
<interface name="public">
  <any-address/>
</interface>
<interface name="unsecure">
  <any-address/>
</interface>
```

13. Set up the JBoss option file in $EAP_HOME/bin/standalone.conf (or the JBoss config you use).

    (a) Set up Java memory options:

    - minimum setting: JAVA_OPTS="-Xms128m -Xmx512m -XX:MaxPermSize=128m .....
    - recommended setting: JAVA_OPTS="-Xms256m -Xmx1024m -XX:MaxPermSize=256m .....

    JVM might require larger amount of memory depending on the uploaded and used LSPS modules.

(b) Set up the server startup configuration file to use (pointing to the configuration file you modified), e.g. for standalone-full.xml:

```
JAVA_OPTS="$JAVA_OPTS -Djboss.server.default.config=standalone-full.xml"
```

(c) Add the following Java system properties to the config file, so it is added to the Java options passed to the JVM:

i. Configure EMF framework to work properly on JBoss:

```
JAVA_OPTS="$JAVA_OPTS -Dorg.eclipse.emf.ecore.EPackage.Registry.INSTANCE=org.eclipse
```

ii. Configure the conversion of empty numeric inputs:

```
JAVA_OPTS="-Dorg.apache.el.parser.COERCE_TO_ZERO=false $JAVA_OPTS"
```

LSPS needs the system property org.apache.el.parser.COERCE_TO_ZERO to be set to "false" (see https://jsp-spec-public.dev.java.net/issues/show_bug.cgi?id=183). Without the property, empty input in numeric fields in the web application is interpreted as 0. This interpretation causes some functionalities not to work.

14. Copy the LSPS Application EAR to $JB_HOME/standalone/deployments.
15. Optional: Hide useless vaadin warning from the log:

```
<logger category="com.vaadin.ui.ConnectorTracker" use-parent-handlers="true">
  <level name="ERROR"/>
</logger>
```

3.2.2 WildFly

You can use WildFly 11, WildFly 10, or WildFly 9.4.

You can integrate the application with a CAS authentication server.

Troubleshooting Data Source

3.2.2.1 Setting up WildFly 11.0.0 with LSPS Application

Important: It is strongly discouraged to run other applications on WildFly Enterprise Application Platform (WF) with LSPS since LSPS is using a customized Hibernate.

To set up WildFly 11.0.0 with LSPS, do the following:

1. Install WildFly.
2. Create WildFly module with the LSPS login module:

(a) Copy lsps-security-jboss-${lsps.version}.jar to $WF_HOME/modules/com/whitestein/lsps/security/main.

(b) Create the file $WF_HOME/modules/com/whitestein/lsps/security/main/module.xml with the following content:

```
<module xmlns="urn:jboss:module:1.0" name="com.whitestein.lsps.security">
 <resources>
   <resource-root path="lsps-security-jboss-${lsps.version}.jar"/>
 </resources>
 <dependencies>
   <module name="javax.api"/>
   <module name="javax.transaction.api"/>
   <module name="org.picketbox" />
 </dependencies>
</module>
```

Take a note of the module name since it is used later for security realm configuration.

3. Create a WildFly module with your JDBC driver. Type IV JDBC drivers are recommended.

   - For MySQL you need to copy MySQL Connector/J 5.5 or later, that is, *mysql-connector-java-<VERSION>-bin.jar*, to $WF_HOME/modules/com/mysql/main. Then create $WF_HOME/modules/com/mysql/main/module.xml with following content:

     ```
     <module xmlns="urn:jboss:module:1.0" name="com.mysql">
      <resources>
        <resource-root path="mysql-connector-java-<VERSION>-bin.jar"/>
      </resources>
      <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
      </dependencies>
     </module>
     ```

   - For Microsoft SQL server, get the Microsoft SQL driver *sqljdbc4.jar* and copy it to $WF_HOME/modules/com/microsoft/sqlserver/mainr.
     Also consider adding the `sendStringParametersAsUnicode=false` property.

     ```
     <module xmlns="urn:jboss:module:1.0" name="com.microsoft.sqlserver">
      <resources>
        <resource-root path="sqljdbc4.jar"/>
      </resources>
      <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
      </dependencies>
     </module>
     ```

   - For Oracle 11g, copy ojdbc6.jar to $WF_HOME/modules/com/oracle/jdbc/main

     ```
     <module xmlns="urn:jboss:module:1.0" name="com.oracle.jdbc">
        <resources>
          <resource-root path="ojdbc6.jar"/>
        </resources>
        <dependencies>
          <module name="javax.api"/>
          <module name="javax.transaction.api"/>
        </dependencies>
      </module>
     ```

     The module name will be used later for creating data source.

4. Configure JMS in the server profile file (typically, $WF_HOME/standalone/configuration/standalone-full.xml):

   (a) Set JMS to be persistent: `<server name="default" persistence-enabled="true">`.

   (b) Add queue and topic under messaging-activemq -> server:

   ```
    <jms-queue name="LSPS_QUEUE" entries="java:jboss/jms/LSPS_QUEUE"/>
   <jms-topic name="LSPS_TOPIC" entries="java:jboss/jms/LSPS_TOPIC"/>
   ```

   LSPS will use the default JMS connection factory.

5. Configure your data source in the server profile file (typically, $WF_HOME/standalone/configuration/standalone-full.xml).

   The data source must be an XA-datasource with the transaction isolation TRANSACTION_READ_COMMITTED and the JNDI name must be *jdbc/LSPS_DS*.

   - For MySql, add the following under the data sources subsystem:

     ```
     <xa-datasource jndi-name="java:/jdbc/LSPS_DS" pool-name="LSPS_DS" enabled="true" use-jav
      <driver>mysqlxa</driver>
      <xa-datasource-property name="URL">jdbc:mysql://localhost:3306/lsps?useUnicode=true&amp
      <security>
     ```

```
      <user-name>lsps</user-name>
      <password>lsps</password>
    </security>
    <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
    <xa-pool>
      <min-pool-size>10</min-pool-size>
      <max-pool-size>20</max-pool-size>
      <prefill>true</prefill>
    </xa-pool>
  </xa-datasource>
```

- For Microsoft SQL server, add the following under the data sources subsystem:

```
  <xa-datasource jndi-name="java:/jdbc/LSPS_DS" pool-name="LSPS_DS" enabled="true" use-jav
     <driver>mssqlxa</driver>
     <xa-datasource-property name="URL">jdbc:sqlserver://localhost;databaseName=lsps</xa-d
     <security>
       <user-name>lsps</user-name>
       <password>lsps</password>
     </security>
     <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
     <xa-pool>
       <min-pool-size>10</min-pool-size>
       <max-pool-size>20</max-pool-size>
       <prefill>true</prefill>
     </xa-pool>
     <new-connection-sql>select 1</new-connection-sql>
     <validation>
       <check-valid-connection-sql>select 1</check-valid-connection-sql>
     </validation>
   </xa-datasource>
```

- For forOracle 11g XE server, add the following under data sources subsystem:

```
  <xa-datasource jndi-name="java:/jdbc/LSPS_DS" pool-name="LSPS_DS" enabled="true" use-jav
     <driver>oraclexa</driver>
     <xa-datasource-property name="URL">jdbc:oracle:thin:@localhost:1521:XE</xa-datasource
     <security>
       <user-name>lsps</user-name>
       <password>lsps</password>
     </security>
     <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
     <xa-pool>
       <min-pool-size>10</min-pool-size>
       <max-pool-size>20</max-pool-size>
       <prefill>true</prefill>
     </xa-pool>
     <validation>
       <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle
       <stale-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle
       <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleE
     </validation>
   </xa-datasource>
```

6. In the `<drivers>` element, make sure to *delete the h2 driver* (driver with the `name="h2"` attribute) and add `<driver>` for your data source:

   - for MySQL:

```
   <driver name="mysqlxa" module="com.mysql">
    <xa-datasource-class>com.mysql.cj.jdbc.MysqlXADataSource</xa-datasource-class>
  </driver>
```

   - for MSSQL:

```
   <driver name="mssqlxa" module="com.microsoft.sqlserver">
    <xa-datasource-class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-datasource-
  </driver>
```

- for Oracle:

```
<driver name="oraclexa" module="com.oracle.jdbc">
 <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
</driver>
```

- Delete the h2 driver.

7. Set Infinispan acquire-timeout setting to at least 5 minutes in your profile XML, such as standalone-full.xml. For details refer to the respective faq entry

```
    <local-cache name="passivation">
        <locking isolation="REPEATABLE_READ" acquire-timeout="600000"/>
        <transaction mode="BATCH"/>
        <file-store passivation="true" purge="false"/>
   </local-cache>
</cache-container>
```

8. Create a mail session with the JNDI name mail/LSPS_MAIL defined in $WF_HOME/standalone/configuration/standalone-full.xml or the respective WildFly config under the mail subsystem tag.

```
<mail-session name="lspsmail" jndi-name="java:jboss/mail/LSPS_MAIL">
 <smtp-server outbound-socket-binding-ref="mail-smtp"/>
</mail-session>
```

9. Set up the login module for the LSPS realm: add the following to the "security-domains" of the security subsystem tag in $WF_HOME/standalone/configuration/standalone-full.xml or the respective WildFly config referencing the security module from step 2:

```
<security-domain name="lspsRealm" cache-type="default">
 <authentication>
   <login-module code="com.whitestein.lsps.security.jboss.LSPSRealm" flag="required" module=
   <module-option name="dsJndiName" value="/jdbc/LSPS_DS"/>
   </login-module>
 </authentication>
</security-domain>
```

The attribute cache-type of the security-domain element turns on caching of user login data. As a side effect, changes to user credentials will not be used until after the cache is flushed; for example, if a user changes user's password through the web console but the user is already using PDS, they will be able to connect PDS to the server with the old credentials. You can turn the cache off by omitting the cache-type attribute of the security-domain element. The security domain definition will look as follows:

```
<security-domain name="lspsRealm">
 <authentication>
   <login-module code="com.whitestein.lsps.security.jboss.LSPSRealm" flag="required" module=
   <module-option name="dsJndiName" value="/jdbc/LSPS_DS"/>
   </login-module>
 </authentication>
</security-domain>
```

For additional module options, refer to the JavaDoc of the implementing class and super classes. You can also plug in your own authentication module.

10. Configure bean pools.

The application requires configuration of bean and data source connection pools. The configuration depends on the expected load as well as other performance parameters. You can change the configuration in the $W↩ F_HOME/standalone/configuration/standalone-full.xml or the respective WildFly config. **Example settings:**

- Limit the MDB pool and the default pool used for timers, async, and remoting.
- Extend the SLSB pool to 10 $*$ (MDB pool size + default pool size) at least.
- Decrease the acquisition timeouts to reasonable values so that in case of lack of resources long waiting deadlocks do not occur. Set data-source connection pool maximum to MDB pool size + default pool size at least, for example:

```
 <bean-instance-pools>
  <!-- A sample strict max pool configuration -->
  <strict-max-pool name="slsb-strict-max-pool" max-pool-size="200" instance-acquisition-t
  <strict-max-pool name="mdb-strict-max-pool" max-pool-size="20" instance-acquisition-tim
</bean-instance-pools>
~
  <thread-pools>
    <thread-pool name="default">
      <max-threads count="10"/>
      ...
    </thread-pool>
  </thread-pools>
```

11. Configure networking (optional).

    By default WildFly binds to the local host. You can change the configuration in $WF_HO↩ ME/standalone/configuration/standalone-full.xml (or the WildFly config you use). For example, to bind WildFly to any IPv4 address use the following setting:

```
 <interface name="management">
  <any-address/>
</interface>
<interface name="public">
  <any-address/>
</interface>
<interface name="unsecure">
  <any-address/>
</interface>
```

12. Set up the WildFly option in $WF_HOME/bin/standalone.conf (or the WildFly config you use).

    (a) Set up Java memory options:

    - minimum setting: JAVA_OPTS="-Xms128m -Xmx512m -XX:MetaspaceSize=96M -XX:Max↩ MetaspaceSize=256m"
    - recommended setting: JAVA_OPTS="-Xms256m -Xmx1024m -XX:MetaspaceSize=96M -XX:↩ MaxMetaspaceSize=256m"

    JVM might require larger amount of memory depending on the uploaded and used LSPS modules.

    (b) Set up the server startup configuration file to use (pointing to the configuration file you modified), e.g. for standalone-full.xml:

```
JAVA_OPTS="$JAVA_OPTS -Djboss.server.default.config=standalone-full.xml"
```

    (c) Add the following Java system properties to the config file, so it is added to the Java options passed to the JVM:

    - Configure EMF framework to work properly on WildFly

      ```
      JAVA_OPTS="$JAVA_OPTS -Dorg.eclipse.emf.ecore.EPackage.Registry.INSTANCE=org.eclipse
      ```

      Without this configuration property, the LSPS application will fail to run properly.
    - Configure conversion of empty numeric inputs

      ```
      JAVA_OPTS="-Dorg.apache.el.parser.COERCE_TO_ZERO=false $JAVA_OPTS"
      ```

      LSPS needs the system property org.apache.el.parser.COERCE_TO_ZERO to be set to "false" (see https://jsp-spec-public.dev.java.net/issues/show_bug.↩ cgi?id=183). Without the property, empty input in numeric fields in the web application is interpreted as 0. This interpretation causes some functionalities not to work.

13. Copy the LSPS Application EAR to $WF_HOME/standalone/deployments.

14. Run WildFly.

3.2.2.2 Setting up WildFly 10.0.0 with LSPS Application

Important: It is strongly discouraged to run other applications on WildFly (WF) along with the LSPS Application since LSPS is using a customized Hibernate.

To set up WildFly 10.0.0 with LSPS, do the following:

1. Install WildFly.
2. Create WildFly module with the LSPS login module:
   (a) Copy lsps-security-jboss-${lsps.version}.jar to $WF_HOME/modules/com/whitestein/lsps/security/main.
   (b) Create the file $WF_HOME/modules/com/whitestein/lsps/security/main/module.xml with the following content:

```
<module xmlns="urn:jboss:module:1.0" name="com.whitestein.lsps.security">
 <resources>
   <resource-root path="lsps-security-jboss-${lsps.version}.jar"/>
 </resources>
 <dependencies>
   <module name="javax.api"/>
   <module name="javax.transaction.api"/>
   <module name="org.picketbox" />
 </dependencies>
</module>
```

   Take a note of the module name since it is used later for security realm configuration.
3. Create a WildFly module with your JDBC driver. Type IV JDBC drivers are recommended.
   - For MySQL you need to copy MySQL Connector/J 5.5 or later, that is, *mysql-connector-java-<VERSION>-bin.jar*, to $WF_HOME/modules/com/mysql/main. Then create $WF_HOME/modules/com/mysql/main/module.xml with following content:

```
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
 <resources>
   <resource-root path="mysql-connector-java-<VERSION>-bin.jar"/>
 </resources>
 <dependencies>
   <module name="javax.api"/>
   <module name="javax.transaction.api"/>
 </dependencies>
</module>
```

   - For Microsoft SQL server, get the Microsoft SQL driver *sqljdbc4.jar* and copy it to $WF_HOME/modules/com/microsoft/sqlserver/mainr.
     Also consider adding the `sendStringParametersAsUnicode=false` property.

```
<module xmlns="urn:jboss:module:1.0" name="com.microsoft.sqlserver">
 <resources>
   <resource-root path="sqljdbc4.jar"/>
 </resources>
 <dependencies>
   <module name="javax.api"/>
   <module name="javax.transaction.api"/>
 </dependencies>
</module>
```

   - For Oracle 11g, copy ojdbc6.jar to $WF_HOME/modules/com/oracle/jdbc/main

```
<module xmlns="urn:jboss:module:1.0" name="com.oracle.jdbc">
 <resources>
   <resource-root path="ojdbc6.jar"/>
 </resources>
 <dependencies>
   <module name="javax.api"/>
   <module name="javax.transaction.api"/>
 </dependencies>
</module>
```

The module name will be used later for creating data source.

4. Configure JMS in the server profile file (typically, $WF_HOME/standalone/configuration/standalone-full.xml):

   (a) Set JMS to be persistent: `<server name="default" persistence-enabled="true">`.

   (b) Add queue and topic under messaging-activemq -> server:

```
 <jms-queue name="LSPS_QUEUE" entries="java:jboss/jms/LSPS_QUEUE"/>
<jms-topic name="LSPS_TOPIC" entries="java:jboss/jms/LSPS_TOPIC"/>
```

   LSPS will use the default JMS connection factory.

5. Configure your data source in the server profile file (typically, $WF_HOME/standalone/configuration/standalone-full.xml).

   The data source must be an XA-datasource with the transaction isolation TRANSACTION_READ_COMM↩ ITTED and the JNDI name must be *jdbc/LSPS_DS*.

   - For MySql, add the following under the data sources subsystem:

```
<xa-datasource jndi-name="java:/jdbc/LSPS_DS" pool-name="LSPS_DS" enabled="true" use-jav
 <driver>mysqlxa</driver>
 <xa-datasource-property name="URL">jdbc:mysql://localhost:3306/lsps?useUnicode=true&amp
 <security>
   <user-name>lsps</user-name>
   <password>lsps</password>
 </security>
 <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
 <xa-pool>
   <min-pool-size>10</min-pool-size>
   <max-pool-size>20</max-pool-size>
   <prefill>true</prefill>
 </xa-pool>
</xa-datasource>
```

   - For Microsoft SQL server, add the following under the data sources subsystem:

```
<xa-datasource jndi-name="java:/jdbc/LSPS_DS" pool-name="LSPS_DS" enabled="true" use-jav
   <driver>mssqlxa</driver>
   <xa-datasource-property name="URL">jdbc:sqlserver://localhost;databaseName=lsps</xa-d
   <security>
     <user-name>lsps</user-name>
     <password>lsps</password>
   </security>
   <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
   <xa-pool>
     <min-pool-size>10</min-pool-size>
     <max-pool-size>20</max-pool-size>
     <prefill>true</prefill>
   </xa-pool>
   <new-connection-sql>select 1</new-connection-sql>
   <validation>
     <check-valid-connection-sql>select 1</check-valid-connection-sql>
   </validation>
 </xa-datasource>
```

- For forOracle 11g XE server, add the following under data sources subsystem:

```
<xa-datasource jndi-name="java:/jdbc/LSPS_DS" pool-name="LSPS_DS" enabled="true" use-jav
   <driver>oraclexa</driver>
   <xa-datasource-property name="URL">jdbc:oracle:thin:@localhost:1521:XE</xa-datasource
   <security>
     <user-name>lsps</user-name>
     <password>lsps</password>
   </security>
   <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
   <xa-pool>
     <min-pool-size>10</min-pool-size>
     <max-pool-size>20</max-pool-size>
     <prefill>true</prefill>
   </xa-pool>
   <validation>
     <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle
     <stale-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle
     <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleE
   </validation>
 </xa-datasource>
```

6. In the <drivers> element, make sure to *delete the h2 driver* (driver with the name="h2" attribute) and add <driver> for your datasource:

   - for MySQL:

```
<driver name="mysqlxa" module="com.mysql">
   <xa-datasource-class>com.mysql.cj.jdbc.MysqlXADataSource</xa-datasource-class>
 </driver>
```

   - for MSSQL:

```
   <driver name="mssqlxa" module="com.microsoft.sqlserver">
   <xa-datasource-class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-datasourc
 </driver>
```

   - for Oracle:

```
  <driver name="oraclexa" module="com.oracle.jdbc">
   <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
 </driver>
```

7. Set Infinispan acquire-timeout setting to at least 5 minutes in your profile XML, such as standalone-full.xml. For details refer to the respective faq entry

```
  <local-cache name="passivation">
      <locking isolation="REPEATABLE_READ" acquire-timeout="600000"/>
      <transaction mode="BATCH"/>
      <file-store passivation="true" purge="false"/>
 </local-cache>
</cache-container>
```

8. Create a mail session with the JNDI name mail/LSPS_MAIL defined in $WF_HOME/standalone/configuration/standalone-full.xml or the respective WildFly config under the mail subsystem tag.

```
<mail-session name="lspsmail" jndi-name="java:jboss/mail/LSPS_MAIL">
 <smtp-server outbound-socket-binding-ref="mail-smtp"/>
</mail-session>
```

   Important: If you are using Microsoft Exchange Server as your mail server, make sure you have enabled the smtp feature.

9. Set up the login module for the LSPS realm: add the following to the "security-domains" of the security subsystem tag in $WF_HOME/standalone/configuration/standalone-full.xml or the respective WildFly config referencing the security module from step 2:

```
 <security-domain name="lspsRealm" cache-type="default">
  <authentication>
    <login-module code="com.whitestein.lsps.security.jboss.LSPSRealm" flag="required" module=
    <module-option name="dsJndiName" value="/jdbc/LSPS_DS"/>
    </login-module>
  </authentication>
</security-domain>
```

The attribute cache-type of the security-domain element turns on caching of user login data. As a side effect, changes to user credentials will not be used until after the cache is flushed; for example, if a user changes user's password through the web console but the user is already using PDS, they will be able to connect PDS to the server with the old credentials. You can turn the cache off by omitting the cache-type attribute of the security-domain element. The security domain definition will look as follows:

```
 <security-domain name="lspsRealm">
  <authentication>
    <login-module code="com.whitestein.lsps.security.jboss.LSPSRealm" flag="required" module=
    <module-option name="dsJndiName" value="/jdbc/LSPS_DS"/>
    </login-module>
  </authentication>
</security-domain>
```

For additional module options, refer to the JavaDoc of the implementing class and super classes. You can also plug in your own authentication module.

10. Configure bean pools.

    The application requires configuration of bean and data source connection pools. The configuration depends on the expected load as well as other performance parameters. You can change the configuration in the $W↩ F_HOME/standalone/configuration/standalone-full.xml or the respective WildFly config. **Example settings:**

    - Limit the MDB pool and the default pool used for timers, async, and remoting.
    - Extend the SLSB pool to 10 ∗ (MDB pool size + default pool size) at least.
    - Decrease the acquisition timeouts to reasonable values so that in case of lack of resources long waiting deadlocks do not occur. Set data-source connection pool maximum to MDB pool size + default pool size at least, for example:

      ```
       <bean-instance-pools>
        <!-- A sample strict max pool configuration -->
        <strict-max-pool name="slsb-strict-max-pool" max-pool-size="200" instance-acquisition-t
        <strict-max-pool name="mdb-strict-max-pool" max-pool-size="20" instance-acquisition-tim
      </bean-instance-pools>
      ~
      <thread-pools>
        <thread-pool name="default">
          <max-threads count="10"/>
          ...
        </thread-pool>
      </thread-pools>
      ```

11. Configure networking (optional).

    By default WildFly binds to the local host. You can change the configuration in $WF_HO↩ ME/standalone/configuration/standalone-full.xml (or the WildFly config you use). For example, to bind WildFly to any IPv4 address use the following setting:

```
 <interface name="management">
  <any-address/>
</interface>
<interface name="public">
  <any-address/>
</interface>
<interface name="unsecure">
  <any-address/>
</interface>
```

12. Set up the WildFly option file in $WF_HOME/bin/standalone.conf (or the WildFly config you use).
    (a) Set up Java memory options:
        - minimum setting: JAVA_OPTS="-Xms128m -Xmx512m -XX:MetaspaceSize=96M -XX:Max↩ MetaspaceSize=256m"
        - recommended setting: JAVA_OPTS="-Xms256m -Xmx1024m -XX:MetaspaceSize=96M -XX:↩ MaxMetaspaceSize=256m"

        JVM might require larger amount of memory depending on the uploaded and used LSPS modules.
    (b) Set up the server startup configuration file to use (pointing to the configuration file you modified), e.g. for standalone-full.xml:

        ```
        JAVA_OPTS="$JAVA_OPTS -Djboss.server.default.config=standalone-full.xml"
        ```

    (c) Add the following Java system properties to the config file, so it is added to the Java options passed to the JVM:
        - Configure EMF framework to work properly on WildFly:

          ```
          JAVA_OPTS="$JAVA_OPTS -Dorg.eclipse.emf.ecore.EPackage.Registry.INSTANCE=org.eclipse
          ```
        - Configure conversion of empty numeric inputs

          ```
          JAVA_OPTS="-Dorg.apache.el.parser.COERCE_TO_ZERO=false $JAVA_OPTS"
          ```

          LSPS needs the system property org.apache.el.parser.COERCE_TO_ZERO to be set to "false" (see https://jsp-spec-public.dev.java.net/issues/show_bug.↩ cgi?id=183). Without the property, empty input in numeric fields in the web application is interpreted as 0. This interpretation causes some functionalities not to work.
13. Copy the LSPS Application EAR to $WF_HOME/standalone/deployments.
14. Run WildFly.
15. Optionally, exclude useless vaadin warnings from the log:

    ```
    <logger category="com.vaadin.ui.ConnectorTracker" use-parent-handlers="true">
      <level name="ERROR"/>
    </logger>
    ```

3.2.2.3 Setting up WildFly 9.0.2 with LSPS Application

Important: It is strongly discouraged to run other applications on WildFly Enterprise Application Platform (WF) with LSPS since LSPS is using a customized Hibernate.

To set up WildFly 9.0.2 with LSPS, do the following:

1. Install WildFly.
2. Create WildFly module with the LSPS login module:
   (a) Copy lsps-security-jboss-${lsps.version}.jar to $WF_HOME/modules/com/whitestein/lsps/security/main.
   (b) Create the file $WF_HOME/modules/com/whitestein/lsps/security/main/module.xml with the following content:

   ```
   <module xmlns="urn:jboss:module:1.0" name="com.whitestein.lsps.security">
    <resources>
      <resource-root path="lsps-security-jboss-${lsps.version}.jar"/>
    </resources>
    <dependencies>
      <module name="javax.api"/>
      <module name="javax.transaction.api"/>
      <module name="org.picketbox" />
    </dependencies>
   </module>
   ```

Take a note of the module name since it is used later for security realm configuration.

3. Create a WildFly module with your JDBC driver. Type IV JDBC drivers are recommended.

   - For MySQL you need to copy MySQL Connector/J 5.5 or later, that is, *mysql-connector-java-<VERSION>-bin.jar*, to $WF_HOME/modules/com/mysql/main. Then create $WF_HOME/modules/com/mysql/main/module.xml with following content:

```
<module xmlns="urn:jboss:module:1.0" name="com.mysql">
 <resources>
   <resource-root path="mysql-connector-java-<VERSION>-bin.jar"/>
 </resources>
 <dependencies>
   <module name="javax.api"/>
   <module name="javax.transaction.api"/>
 </dependencies>
</module>
```

   - For Microsoft SQL server, get the Microsoft SQL driver *sqljdbc4.jar* and copy it to $WF_HOME/modules/com/microsoft/sqlserver/mainr.
     Also consider adding the `sendStringParametersAsUnicode=false` property.

```
<module xmlns="urn:jboss:module:1.0" name="com.microsoft.sqlserver">
 <resources>
   <resource-root path="sqljdbc4.jar"/>
 </resources>
 <dependencies>
   <module name="javax.api"/>
   <module name="javax.transaction.api"/>
 </dependencies>
</module>
```

   - For Oracle 11g, copy ojdbc6.jar to $WF_HOME/modules/com/oracle/jdbc/main

```
<module xmlns="urn:jboss:module:1.0" name="com.oracle.jdbc">
 <resources>
   <resource-root path="ojdbc6.jar"/>
 </resources>
 <dependencies>
   <module name="javax.api"/>
   <module name="javax.transaction.api"/>
 </dependencies>
</module>
```

   The module name will be used later for creating data source.

4. Configure JMS in the server profile file (typically, $WF_HOME/standalone/configuration/standalone-full.xml):

   (a) Set JMS to be persistent: `<persistence-enabled>true</persistence-enabled>`

```
<subsystem xmlns="urn:jboss:domain:messaging:3.0">
  <hornetq-server>
     <persistence-enabled>true</persistence-enabled>
     <journal-file-size>102400</journal-file-size>
...
```

   (b) Add queue and topic under jms-destinations:

```
 <jms-queue name="LSPS_QUEUE">
  <entry name="java:jboss/jms/LSPS_QUEUE"/>
</jms-queue>
<jms-topic name="LSPS_TOPIC">
  <entry name="java:jboss/jms/LSPS_TOPIC"/>
</jms-topic>
```

   LSPS will use the default JMS connection factory.

5. Configure your data source in the server profile file (typically, $WF_HOME/standalone/configuration/standalone-full.xml).

   The data source must be an XA-datasource with the transaction isolation TRANSACTION_READ_COMMITTED and the JNDI name must be *jdbc/LSPS_DS*.

   - For MySql, add the following under the data sources subsystem:

```
<xa-datasource jndi-name="java:/jdbc/LSPS_DS" pool-name="LSPS_DS" enabled="true" use-jav
 <driver>mysqlxa</driver>
 <xa-datasource-property name="URL">jdbc:mysql://localhost:3306/lsps?useUnicode=true&amp
 <security>
   <user-name>lsps</user-name>
   <password>lsps</password>
 </security>
 <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
 <xa-pool>
   <min-pool-size>10</min-pool-size>
   <max-pool-size>20</max-pool-size>
   <prefill>true</prefill>
 </xa-pool>
</xa-datasource>
```

   - For Microsoft SQL server, add the following under the data sources subsystem:

```
<xa-datasource jndi-name="java:/jdbc/LSPS_DS" pool-name="LSPS_DS" enabled="true" use-jav
 <driver>mssqlxa</driver>
 <xa-datasource-property name="URL">jdbc:sqlserver://localhost;databaseName=lsps</xa-dat
 <security>
   <user-name>lsps</user-name>
   <password>lsps</password>
 </security>
 <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
 <xa-pool>
   <min-pool-size>10</min-pool-size>
   <max-pool-size>20</max-pool-size>
   <prefill>true</prefill>
 </xa-pool>
 <new-connection-sql>select 1</new-connection-sql>
 <validation>
   <check-valid-connection-sql>select 1</check-valid-connection-sql>
 </validation>
</xa-datasource>
```

   - For forOracle 11g XE server, add the following under data sources subsystem:

```
<xa-datasource jndi-name="java:/jdbc/LSPS_DS" pool-name="LSPS_DS" enabled="true" use-jav
 <driver>oraclexa</driver>
 <xa-datasource-property name="URL">jdbc:oracle:thin:@localhost:1521:XE</xa-datasource-p
 <security>
   <user-name>lsps</user-name>
   <password>lsps</password>
 </security>
 <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
 <xa-pool>
   <min-pool-size>10</min-pool-size>
   <max-pool-size>20</max-pool-size>
   <prefill>true</prefill>
 </xa-pool>
 <validation>
   <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.O
   <stale-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.O
   <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.oracle.OracleExc
 </validation>
</xa-datasource>
```

6. In the `<drivers>` element, make sure to *delete the h2 driver* (driver with the `name="h2"` attribute) and add `<driver>` for your data source:

- for MySql:

```
<driver name="mysqlxa" module="com.mysql">
<xa-datasource-class>com.mysql.cj.MysqlXADataSource</xa-datasource-class>
</driver>
```

- for MSSql:

```
<driver name="mssqlxa" module="com.microsoft.sqlserver">
<xa-datasource-class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-datasourc
</driver>
```

- for Oracle:

```
<driver name="oraclexa" module="com.oracle.jdbc">
<xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
</driver>
```

7. Set Infinispan acquire-timeout setting to at least 5 minutes in your profile XML, such as standalone-full.xml. For details refer to the respective faq entry

```
    <local-cache name="passivation">
        <locking isolation="REPEATABLE_READ" acquire-timeout="600000"/>
        <transaction mode="BATCH"/>
        <file-store passivation="true" purge="false"/>
   </local-cache>
</cache-container>
```

8. Create a mail session with the JNDI name mail/LSPS_MAIL defined in $WF_HOME/standalone/configuration/standalone-full.xml or the respective WildFly config under the mail subsystem tag.

```
    <mail-session name="lspsmail" jndi-name="java:jboss/mail/LSPS_MAIL">
    <smtp-server outbound-socket-binding-ref="mail-smtp"/>
  </mail-session>
```

   Important: If you are using Microsoft Exchange Server as your mail server, make sure you have enabled the smtp feature.

9. Set up the login module for the LSPS realm: add the following to the "security-domains" of the security subsystem tag in $WF_HOME/standalone/configuration/standalone-full.xml or the respective WildFly config referencing the security module from step 2:

```
 <security-domain name="lspsRealm" cache-type="default">
  <authentication>
    <login-module code="com.whitestein.lsps.security.jboss.LSPSRealm" flag="required" module=
       <module-option name="dsJndiName" value="/jdbc/LSPS_DS"/>
    </login-module>
  </authentication>
</security-domain>
```

   The attribute cache-type of the security-domain element turns on caching of user login data. As a side effect, changes to user credentials will not be used until after the cache is flushed; for example, if a user changes user's password through the web console but the user is already using PDS, they will be able to connect PDS to the server with the old credentials. You can turn the cache off by omitting the cache-type attribute of the security-domain element. The security domain definition will look as follows:

```
 <security-domain name="lspsRealm">
  <authentication>
    <login-module code="com.whitestein.lsps.security.jboss.LSPSRealm" flag="required" module=
    <module-option name="dsJndiName" value="/jdbc/LSPS_DS"/>
    </login-module>
  </authentication>
</security-domain>
```

   For additional module options, refer to the JavaDoc of the implementing class and super classes. You can also plug in your own authentication module.

10. Configure bean pools.

    The application requires configuration of bean and data source connection pools. The configuration depends on the expected load as well as other performance parameters. You can change the configuration in the $WF_HOME/standalone/configuration/standalone-full.xml or the respective WildFly config.

    - Limit the MDB pool and the default pool used for timers, async, and remoting.
    - Extend the SLSB pool to 10 ∗ (MDB pool size + default pool size) at least.
    - Decrease the acquisition timeouts to reasonable values so that in case of lack of resources long waiting deadlocks do not occur. Set data-source connection pool maximum to MDB pool size + default pool size at least. For example:

```
 <bean-instance-pools>
  <strict-max-pool name="slsb-strict-max-pool" max-pool-size="200" instance-acquisition-t
  <strict-max-pool name="mdb-strict-max-pool" max-pool-size="20" instance-acquisition-tim
~
<thread-pools>
  <thread-pool name="default">
    <max-threads count="10"/>
    ...
  </thread-pool>
</thread-pools>
```

11. On Wildfly 8 and 9 enable stateless session beans pooling (on WF 10, the feature is enabled by default, see https://developer.jboss.org/message/881747):

```
<session-bean>
 <stateless>
   <bean-instance-pool-ref pool-name="slsb-strict-max-pool"/>
 </stateless>
```

12. Configure networking (optional).

    By default WildFly binds to the local host. You can change the configuration in $WF_HO↩ ME/standalone/configuration/standalone-full.xml (or the WildFly config you use). For example, to bind WildFly to any IPv4 address use the following setting:

```
<interface name="management">
   <any-ipv4-address/>
 </interface>
 <interface name="public">
   <any-ipv4-address/>
 </interface>
 <interface name="unsecure">
   <any-ipv4-address/>
 </interface>
```

13. Set up the WildFly option file in $WF_HOME/bin/standalone.conf (or the WildFly config you use; on Windows, standalone.conf.bat).

    (a) Set up Java memory options:

    - minimum setting: JAVA_OPTS="-Xms128m -Xmx512m -XX:MaxPermSize=128m"
    - recommended setting: JAVA_OPTS="-Xms256m -Xmx1024m -XX:MaxPermSize=256m"

    JVM might require larger amount of memory depending on the uploaded and used LSPS modules.

    (b) Set up the server startup configuration file to use (pointing to the configuration file you modified), e.g. for standalone-full.xml:

```
JAVA_OPTS="$JAVA_OPTS -Djboss.server.default.config=standalone-full.xml"
```

    (c) Add the following Java system properties to the config file, so it is added to the Java options passed to the JVM:

    - Configure EMF framework to work properly on WildFly.

```
JAVA_OPTS="$JAVA_OPTS -Dorg.eclipse.emf.ecore.EPackage.Registry.INSTANCE=org.eclipse
```

- Configure the conversion of empty numeric inputs.

  ```
  JAVA_OPTS="-Dorg.apache.el.parser.COERCE_TO_ZERO=false $JAVA_OPTS"
  ```

  LSPS needs the system property org.apache.el.parser.COERCE_TO_ZERO to be set to "false" (see https://jsp-spec-public.dev.java.net/issues/show_bug.cgi?id=183). Without the property, empty input in numeric fields in the web application is interpreted as 0. This interpretation causes some functionalities not to work.

14. Copy the LSPS Application EAR to $WF_HOME/standalone/deployments.
15. Run WildFly.
16. Optionally, exclude useless vaadin warnings from the log:

```
<logger category="com.vaadin.ui.ConnectorTracker" use-parent-handlers="true">
  <level name="ERROR"/>
</logger>
```

3.2.2.4 Integration with CAS

Integration with CAS allows the user to authenticate against the CAS server and use the authentication to access the LSPS Application User Interface and Management Console.

Before setting up CAS integration, make sure that:

- WildFly is configured correctly.
- LSPS EAR can be deployed to the application server successfully.
- WildFly is configured to support HTTPS on port 8443.

To integrate LSPS applications with CAS, do the following:

1. Download the module with Java CAS Client classes to WildFly. They are available on the apereo webpage.
2. Extract the archive to the WildFly modules directory <WILDFLY_HOME>/modules/

   The resulting structure is <WF_HOME>/modules/org/jasig/cas/main/
3. Add the org.jasig.cas module as a global module of the *urn:jboss:domain:ee:3.0* subsystem.

```
<subsystem xmlns="urn:jboss:domain:ee:3.0">
  <global-modules>
      <module name="org.jasig.cas" slot="main"/>
  </global-modules>
  ...
</subsystem>
```

4. Configure the *lspsRealm* security domain to use the cas login-module.

   This is the security domain used by the LSPS EAR. By default, LSPS uses the LSPS database for authorization and authentication.

```
 <!--ORIGINAL REALM
<security-domain name="lspsRealm" cache-type="default">
    <authentication>
        <login-module code="com.whitestein.lsps.security.jboss.LSPSRealm" flag="required" mod
            <module-option name="dsJndiName" value="/jdbc/LSPS_DS"/>
        </login-module>
    </authentication>
</security-domain>-->
<!--REALM USES CAS -->
<security-domain name="lspsRealm">
    <authentication>
        <login-module code="org.jasig.cas.client.jaas.CasLoginModule" flag="required" module=
            <module-option name="ticketValidatorClass" value="org.jasig.cas.client.validation
            <module-option name="casServerUrlPrefix" value="https://localhost:8181/cas"/>
            <module-option name="tolerance" value="20000"/>
            <!--defaultRole user will be granted to any user that successfully authenticates
            <module-option name="defaultRoles" value="user"/>
            <module-option name="roleAttributeNames" value="memberOf,eduPersonAffiliation,aut
            <module-option name="principalGroupName" value="CallerPrincipal"/>
            <module-option name="roleGroupName" value="Roles"/>
            <module-option name="cacheAssertions" value="true"/>
            <module-option name="cacheTimeout" value="480"/>
        </login-module>
    </authentication>
</security-domain>
```

5. Configure lsps web applications in individual web.xml files:
   - Add the CAS servlet filter to the *beginning* of the servlet filter chain:

```
 <filter>
    <filter-name>Servlet3 Authentication Filter</filter-name>
    <filter-class>org.jasig.cas.client.jaas.Servlet3AuthenticationFilter</filter-class>
    <init-param>
        <param-name>serverName</param-name>
        <param-value> https://localhost:8443</param-value>
    </init-param>
</filter>
<filter>
    <filter-name>CAS Authentication Filter</filter-name>
    <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>
    <init-param>
        <param-name>casServerLoginUrl</param-name>
        <param-value> https://localhost:8181/cas/login</param-value>
    </init-param>
    <init-param>
        <param-name>serverName</param-name>
        <param-value> https://localhost:8443</param-value>
    </init-param>
</filter>
```

   - In the *session-config* element, set the COOKIE tracking mode:

```
 <session-config>
    <session-timeout>30</session-timeout>
    <tracking-mode>COOKIE</tracking-mode>
</session-config>
```

6. Recompile and redeploy your applications.

3.2.2.5 Troubleshooting Data Source

3.2.2.5.1 XAException on Oracle 10g R2 or 11g

The server returns the following XA exception:

```
WARN  [com.arjuna.ats.jta.logging.loggerI18N] [com.arjuna.ats.internal.jta.recovery.xarecovery1
```

To remedy the situation, do the following:

- For Oracle, make sure the Oracle user has access to the appropriate tables so they can accomplish the recovery:

```
 GRANT SELECT ON sys.dba_pending_transactions TO user;
GRANT SELECT ON sys.pending_trans$ TO user;
GRANT SELECT ON sys.dba_2pc_pending TO user;
GRANT EXECUTE ON sys.dbms_xa TO user;
```

The above assumes that the user is the user defined to connect from WildFly to Oracle. It also assumes that either Oracle 10g R2 (patched for bug 5945463) or 11g is used. If an unpatched version, that is, a version older that 11g, is used, change the last GRANT EXECUTE to the following:

```
GRANT EXECUTE ON sys.dbms_system TO user;
```

3.2.3 Oracle WebLogic

To set up LSPS on Oracle WebLogic application server 11gR1 (10.3.2) and newer, do the following:

1. Make sure the JVM runs with the recommended memory settings:
   - Minimum: -Xmx512m -XX:PermSize=128m (Usually satisfied by default application server settings)
   - Optimum: -Xmx1024m -XX:PermSize=256m

   The amount of memory required might increase depending on uploaded and used lsps modules.
2. Ensure JPA2 is supported on app server (see http://wiki.eclipse.org/EclipseLink/↩ Development/JPA_2.0/weblogic#20110115:_JPA_2.0_using_EclipseLink_on_↩ WebLogic_10.3.4.0 for further information).
3. If you are using WebLogic 12.2. and newer, run Java with the following properties:

   ```
   -Dcom.sun.xml.ws.spi.db.BindingContextFactory=com.sun.xml.ws.db.glassfish.JAXBRIContextFacto
   ```
4. Install WebLogic.
5. If necessary, copy your database JDBC driver libraries to to the `lib` directory of your domain (`$WL_HO↩ ME/server/lib` or `$WL_HOME/user_projects/domains/<YOUR_DOMAIN>/lib`).
6. Copy LSPS security module *lsps-security-weblogic-$LSPS_VERSION.jar* to $WL_HOME/server/lib/mbeantypes.
7. Start WebLogic.
8. Create JDBC XA data source named *LSPS_DS* with JNDI name *jdbc/LSPS_DS*.
   Example configuration for an Oracle database installed on localhost

   ```
    JDBC Data Source:
   Name: LSPS_DS
   JNDI Name: jdbc/LSPS_DS
   Database Type: Oracle
   Database Driver: Oracle's Driver (Thin XA)
   Database Name: LSPS
   Host Name: localhost
   Port: 1521
   Database User Name: lsps
   Password: lsps
   ```

9. Create JMS resouces: Create a JMS server, you can also use your own instance.

   Example with a default persistent store

   - JMS Server:
     - Name: LSPS_JMS_SERVER
     - Persistent Store: (none)
     - Target: (your server name)
   - JMS Module:
     - Name: LSPS_JMS
     - Target: (your server name)
   - Resources inside JMS Module LSPS_JMS:
     - Subdeployment:
       * Name: LSPS_DEPL
       * Targets:
         · JMS Servers: LSPS_JMS_SERVER
   - Queue:
     - Name: LSPS_QUEUE
     - JNDI Name: jms/LSPS_QUEUE
     - Subdeployments: LSPS_DEPL
   - Connection Factory:
     - Name: LSPS_CF
     - JNDI Name: jms/LSPS_CF
     - Targets: (your server name)
     - XA Connection Factory Enabled: true (Detail/Configuration/Transactions)
   - Topic:
     - Name: LSPS_TOPIC
     - JNDI Name: jms/LSPS_TOPIC
     - Subdeployments: LSPS_DEPL
     - Targets: (your server name)

10. Optionally, set up LSPS authentication provider. Use your own authentication providers if necessary.

    (a) Open your default security realm and go to Providers.
    (b) Create new authentication provider:

    ```
    Authentication Provider:
     Name: LSPS_AUTHENTICATOR
     Type: LSPSAuthenticator (this should be in the list if you copied the LSPS security m
    ```

11. Set the authentication provider:

    (a) Reorder authentication providers so that LSPS_AUTHENTICATOR is the first provider.
    (b) Set the provider control flag to SUFFICIENT (you can find the control flag in the provider detail page)

    Alternatively, set the control flag on all providers to OPTIONAL.

12. Restart WebLogic.

13. If you are using WLS 12.1.1, add the following line to the *bin/setDomainEnv.cmd* file of your domain to change the default JAXB provider:

    ```
    SET EXT_PRE_CLASSPATH=WL_HOME%/../modules/databinding.override_1.0.0.0.jar
    ```

14. Deploy $WL_HOME/common/deployable-libraries/jsf-1.2.war as a library to the WebLogic server.

15. Deploy the LSPS application EAR.

Now you can upload your modules to the server, for example, with the cli console.

3.2.4 IBM WebSphere

To set up LSPS on IBM WebSphere Application Server, do the following:

1. Make sure the JVM runs with the recommended memory settings:
   - Minimum: -Xmx512m -XX:PermSize=128m (Usually satisfied by default application server settings)
   - Optimum: -Xmx1024m -XX:PermSize=256m

   The amount of memory required might increase depending on uploaded and used lsps modules.
2. Ensure JPA2 is supported on app server (Feature Pack for OSGi Applications and JPA 2.0 is installed).
3. Install WebSphere.
4. Create JDBC provider and JDBC XA data source with JNDI name jdbc/LSPS_DS and transaction isolation read-committed.

   Example configuration for a DB2 database installed on localhost

   (a) Go to security/Global Security/Java Authentication and Authorization Service/J2C authentication data and create new entry:

   ```
    Alias: LSPS_DS_AUTH
   User ID: db2admin
   Password: db2admin
   ```

   (b) Go to Resources/JDBC/JDBC providers and create JDBC provider:

   ```
    DatabaseType: DB2
   Provider type: DB2 Universal JDBC driver provider
   Implementation type: XA Datasource
   Name: DB2 Universal JDBC Driver Provider (XA) - default name
   Look at classpath:
     ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc4.jar
     ${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar
     ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cisuz.jar
   ```

   Make sure the WAS variable DB2UNIVERSAL_JDBC_DRIVER_PATH is set correctly.

   (c) Go to Resources/JDBC/Data sources:

   i. Create the JDBC data source:

   ```
    Data source name: LSPS_DS
   JNDI name: jdbc/LSPS_DS
   JDBC provider: DB2 Universal JDBC driver provider
   Driver type: 4
   Database name: LSPS
   Server name: localhost
   Port number: 50000
   Use this data source in container managed persistence (CMP): true
   Authentication alias for XA recovery: LSPS_DS_AUTH
   Component-managed authentication alias: LSPS_DS_AUTH
   ```

   ii. Add custom properties:
   - webSphereDefaultIsolationLevel with value 2 (this means transaction isolation level read-commited)
   - progressiveStreaming with value 2 (see https://issues.jboss.org/browse/JB↩ PAPP-2613, also https://www.ibm.com/developerworks/community/forums/html/topi or properties `fullyMaterializeLobData=true`, `fullyMaterializeInput↩ Streams=true`, `progressiveStreaming=2`, `progresssiveLocators=2`)
   - downgradeHoldCursorsUnderXa=true
   - resultSetHoldabilityForCatalogQueries=1

   iii. Test the connection.

5. Create a **JMS queue** with the JNDI name *jms/LSPS_QUEUE*, **JMS queue connection factory** with the JNDI name *jms/LSPS_CF*, and the corresponding **JMS activation specifications** with the JNDI names *jms/LS↩ PS_AS* and *jms/LSPS_WS_AS*.

   Example configuration using the default messaging provider and LSPS database to store messages

   (a) Go to Service Integration/Buses:

      i. Create a new bus:

```
 Name: LSPS_BUS
Security: NO
```

      ii. In LSPS_BUS detail page, click Bus members link and create a bus member:

```
 Server: default value
Message store: Data store
Use existing data source
Data source JNDI name: jdbc/LSPS_DS
Schema name: empty
Authentication alias: LSPS_DS_AUTH
```

      iii. In LSPS_BUS detail page, click Destinations link and create the following destinations:

      - queue:

```
 Destination type: QUEUE
Identifier: LSPS_DEST
Bus member: Your local server node
```

      - topic:

```
 Destination type: TOPIC SPACE
Identifier: LSPS_TOPIC
Bus member: Your local server node
```

   (b) Go to Resources/JMS/Connection Factories and create a connection factory:

```
 Provider: Default messaging provider
Name: LSPS_CF
JNDI Name: jms/LSPS_CF
BUS Name: LSPS_BUS
```

   (c) Go to Resources/JMS/Queues and create a queue:

```
 Name: LSPS_QUEUE
JNDI Name: jms/LSPS_QUEUE
Bus name: LSPS_BUS
Queue name: LSPS_DEST
```

   (d) Go to Resources/JMS/Topics and create a topic:

```
 Name: LSPS_TOPIC
JNDI Name: jms/LSPS_TOPIC
Bus name: LSPS_BUS
Queue name: LSPS_TOPIC
```

   (e) Go to Resources/JMS/Activation specifications and create activation specifications:

```
 Name: LSPS_AS
JNDI Name: jms/LSPS_AS
Destination type: Queue
BUS_NAME: LSPS_BUS
Destination JNDI name: jms/LSPS_QUEUE

 Name: LSPS_WS_AS
JNDI Name: jms/LSPS_WS_AS
Destination type: Topic
BUS_NAME: LSPS_BUS
Destination JNDI name: jms/LSPS_TOPIC
```

6. Turn on and set up security. This step is required only if you want to use LSPS custom authentication. You can also use your own authentication, such as, LDAP.

   (a) Copy lsps-security-websphere-$LSPS_VERSION.jar to $WAS_HOME/lib/ext and restart WebSphere.

   (b) Go to Security/Global Security:

i. Enable administrative security: true
ii. Enable application security: true

(c) Go to Security/Global Security and configure User account repository:

i. Choose Standalone custom registry and click Set as current
ii. Click Configure:
- Primary administrative user name: admin
- Custom registry class name: com.whitestein.lsps.security.LSPSUserRegistry

7. Create mail session with JNDI name mail/LSPS_MAIL

(a) Go to Resources/Mail/Mail Sessions and create mail session:

```
 Name: LSPS_MAIL
JNDI Name: mail/LSPS_MAIL
Outgoing Mail Properties/Server: <your_mail_server>
Outgoing Mail Properties/Protocol: smtp/smtps
Outgoing Mail Properties/User: <username>
Outgoing Mail Properties/Password: <password>
Outgoing Mail Properties/Verify Password: <password>
Outgoing Mail Properties/Return e-mail address: <return_e-mail_address>
```

Important: If you are using Microsoft Exchange Server as your mail server, make sure you have enabled the smtp feature.

8. Set org.apache.el.parser.COERCE_TO_ZERO property to false:

(a) Go to Servers/Server types/WebSphere application servers
(b) Select your server.
(c) Go to Java and Process Management/Process definition/Java Virtual Machine/Custom properties and define the property *org.apache.el.parser.COERCE_TO_ZERO* with the value *false*.
(d) Restart server.

9. Deploy the `LSPS application EAR`.

Now you can upload your modules to the server, for example, `with the cli console`.

Chapter 4

Updating

When deploying new artifacts, regardless of whether you are upgrading the entire LSPS stack, your modules, make sure that the version of the LSPS system database corresponds to the LSPS version your application and modules are created in: If you migrated the LSPS system database to a particular version of LSPS, then your modules and potentially your business database schema with your data should be migrated to the same version.

The resources for the migration should be provided along with the updated application EAR or modules. For information on the resources, refer to developer documentation.

4.1 Uploading Updated Modules

Required artifacts:

- zip files with modules and models
- optionally, model update definition files
- optionally, database schema script

To update modules on your LSPS Server:

1. Terminate or suspend running model instances.
2. If applicable, migrate the business database schema.
3. Upload new models with the database update strategy set to *Do nothing with database schema*, *Update the schema by model*, or *Validate schema first*. If you have migrated the database in the previous step, set the strategy to *Do nothing with database schema*.
4. If applicable, update the suspended model instances.

   Important: Model update can be very complex. Prepare the model update resources beforehand and test it properly.
5. Continue the suspended model instances.

4.2 Updating the LSPS Application

Required artifacts:

- LSPS Application EAR

Provided the new LSPS Application has been created in the same minor version of PDS, for example 3.3, it is enough to deploy the new LSPS Application EAR to the application server. Otherwise, refer to Upgrading LSPS.

4.3 Upgrading LSPS

Required artifacts:

- LSPS Application EAR
- zip files with modules and models exported with GO-BPMN export
- optionally, model update definition files
- optionally, database schema script for business data when applicable

When upgrading your LSPS, you will need to upgrade the entire stack:

- LSPS system database: Mind that the database holds the LSPS system data *and possibly* your business data based on the LSPS data type models.
- LSPS Application: apart from updating the EAR, you might need to upgrade also the version of your database and application server.
- Models and modules: even if you have not introduced changes to your models, you need to at least upgrade the standard library modules they use and make sure you models work with them correctly.

  Important: Make sure to test the upgrade in a testing environment that is **identical to the production environment** and **back up your database** before you perform any changes on production.

To upgrade to a newer LSPS version, do the following:

1. Make sure the environment meets the requirements of LSPS.
2. `Suspend any running model instances`.
3. Stop the application server.
4. Migrate the LSPS system database.
5. Deploy the EAR file to your LSPS server (refer to instructions on `how to set up a server for LSPS`.
6. Upload updated modules and update their instances.
7. Migrate business data.
8. Start the server.

4.3.1 Migrating the LSPS System Database Tables

When upgrading your LSPS application, you will need to upgrade the LSPS system database.

> **Important:** Make sure to **back up the database** and **test the migration process in a testing environment that is identical to the production environment** before you perform any changes on production.

To upgrade the LSPS system database to correspond to a newer LSPS version, run the migration script:

- If **your LSPS database contains the `LSPS_SCHEMA_VERSION` table**, run the migration script with the databaseUrl and databaseType parameter: the script will migrate your current database to the database version of the Runtime Suite.

```
~/lsps-runtime/db-migration$ ./migrate.sh databaseUrl:jdbc:h2:tcp://localhost/./h2/h2 user:eko
      password:mypasswd\$\'\`\"
```

- Otherwise, find the schema version in the migration tool:

  1. Get current database version: open <YOUR_OLD_RUNTIME_SUITE>/db-migration/lsps-db-migration-<VERSION>.jar > db > migration:
     (a) Check the most recent java migration class in the location, for example, *V3_1_0_010__Upgrade.class*
     (b) Open the directory of your database, for example *mysql* and check the latest SQL script, for example, *MYSQL_V3_1_0_003__Upgrade.sql*

     Your current version is the higher version (in the example case, it is *3.1.0.010*).
  2. Run the migration script with the *initialVersion* parameter set to the current version of the LSPS database:
     First database migration with the user **eko** and password *mypasswd$'"`*

```
~/lsps-runtime/db-migration$ ./migrate.sh databaseUrl:jdbc:h2:tcp://localhost/./h2/h2 user:eko
      password:mypasswd\$\'\`\" initialVersion:3.1.0.010
```

The database migration script accepts the following parameters with the respective value entered after a colon:

databaseUrl∗ URI of the database with LSPS runtime data

> **Note:** For SQL databases, the instance property is supported: add the instance to the url as `instance=<name>`, for example, `./migrate.sh databaseType:SQLSERVER databaseUrl:jdbc:sqlserver://localhost/lsps;instance=whitestein`.

databaseType type of the database (The migration tool attempts to identify the database type automatically from the databaseUrl parameter. However, if necessary, you can define the database type explicitly. The possible values are `H2`, `DB2`, `ORACLE`, `SQLSERVER`, `MYSQL`.

initialVersion The current version of LSPS database: it corresponds to the LSPS that created the database. The parameter is required if the database is being migrated for the first time: On the first migration, the underlying tooling creates a database table with information on the previous and current schema version. Next migrations use this data to identify the initial database version so that the parameter is no longer required.

targetVersion The parameter is optional. If undefined, the database is migrated to the Runtime Suite database version.

When running the DB migration tool from the command line, make sure to escape any special characters in parameter values as required by your operating system or shell. For example, if using Bash and a parameter contains characters like $ ‘ ' " and so on then these characters must be escaped: Read the manual to your command line to learn how to escape special characters.

Chapter 5

Configuration and Monitoring

You can configure and monitor the LSPS Server via JMX or in the LSPS_SETTINGS database table and can be changed via JMX:

- Monitoring (HealthCheck) provides statistics on LSPS Server
- Settings exposes server and database related setting

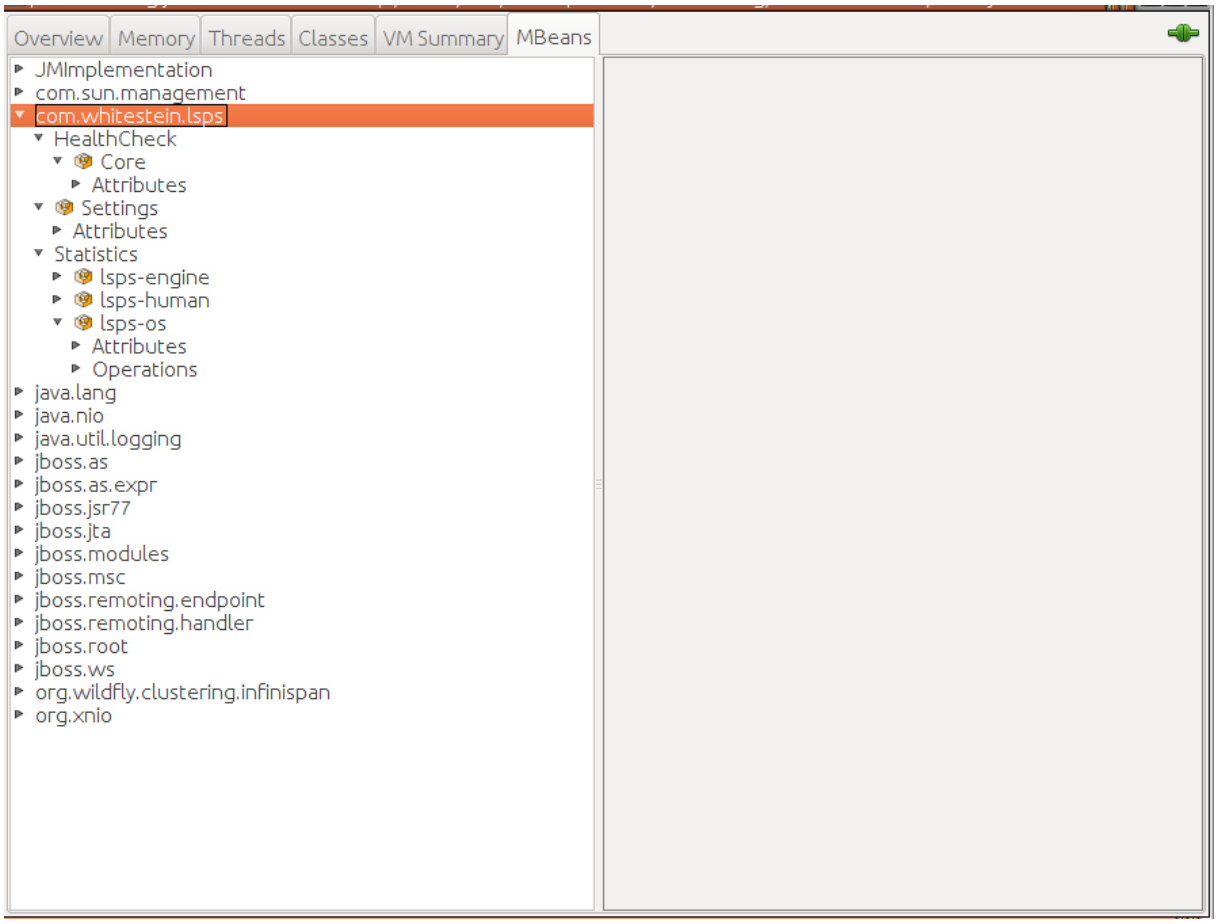


Figure 5.1 LSPS Server settings in jconsole

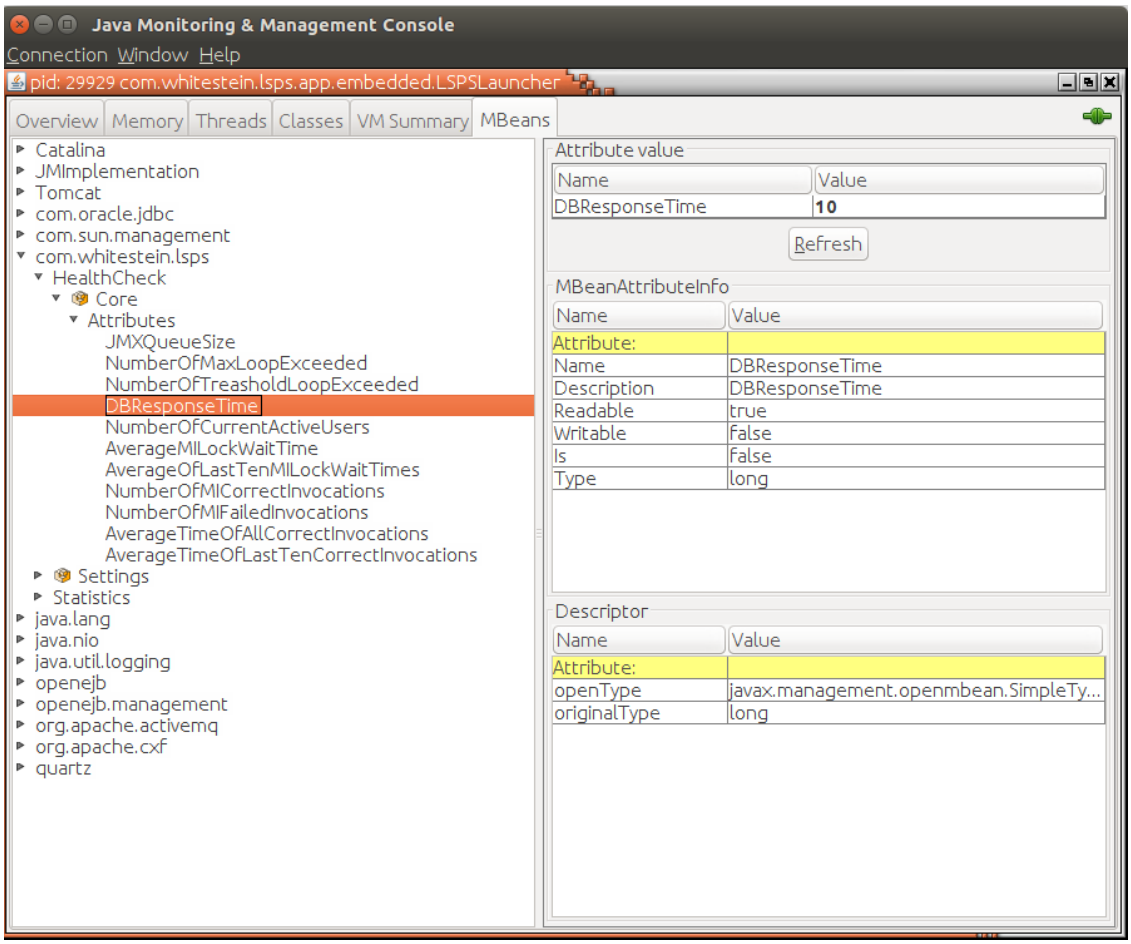
5.1 Monitoring the LSPS Application

To check the health of the LSPS Application, open a JMX-compliant monitoring tool, such as JConsole, and check the attributes under **com.whitestein.lsps.HealtCheck**:

- JMXQueueSize: current amount of JMX messages queued for processing
- NumberOfMICorrectInvocations: the number of successful invocations on all model instances since server start
- NumberOfMIFailedInvocations: the number of failed invocation on all model instances since server start
- AverageTimeOfAllCorrectInvocations: the average execution time of all successful invocations in milliseconds
- AverageTimeOfLastTenCorrectInvocations: the average execution time of the last 10 invocations in milliseconds
- AverageMILockWaitTime and AverageOfLastTenMILockWaitTimes: total average and average of the last 10 waiting times when accessing model instances

  Model instances run in a single thread and can be hence accessed by one entity at a time. While accessed, the instance is locked and other entities wait for the lock to be released: this time is used to calculate the average model-instance-lock wait time. when synchronizing model instances
- NumberOfMaxLoopExceeded: the number of times that a loop exceeded the maximum allowed number of loop runs

  The maximum number of loops is set by the MaxNumberOfEngineLoops setting.
- NumberOfThresholdLoopExceeded: the number of times that a loop exceeded the threshold number of loop runs
- DBResponseTime: time to receive database response
- NumberOfCurrentActiveUsers: number of active users (only relevant if `user tracking` is enabled.)



5.2 Server and Database Settings

LSPS configuration is generally stored in the LSPS_SETTINGS database table and can be changed directly in the database or via JMX. For some setting changes, the server needs to be restarted: the applicable information is provided with individual settings.

Note that some settings, such as debugging settings, are passed as system properties to the server, that is, in the format `-Dkey=value`.

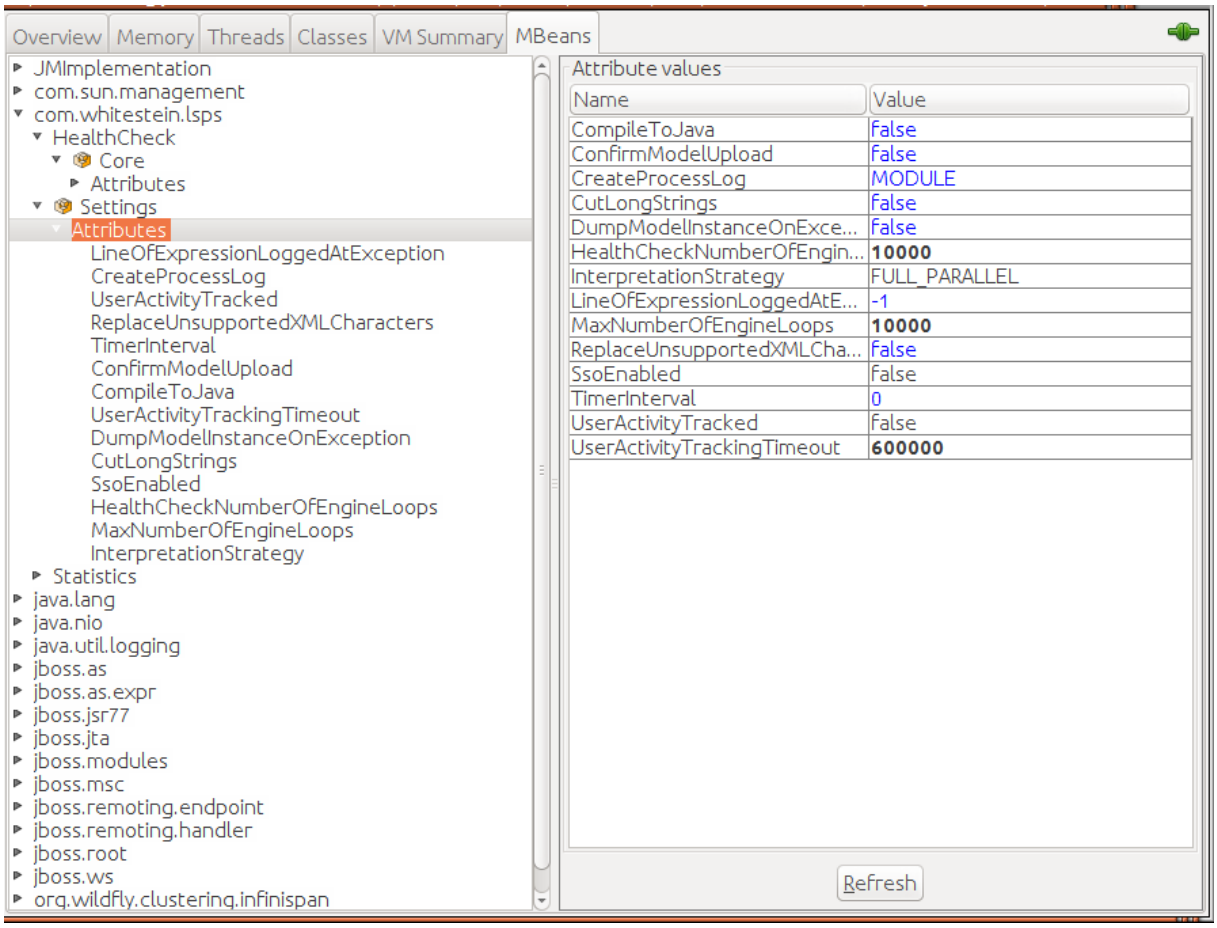


Figure 5.2 LSPS settings in MBeans

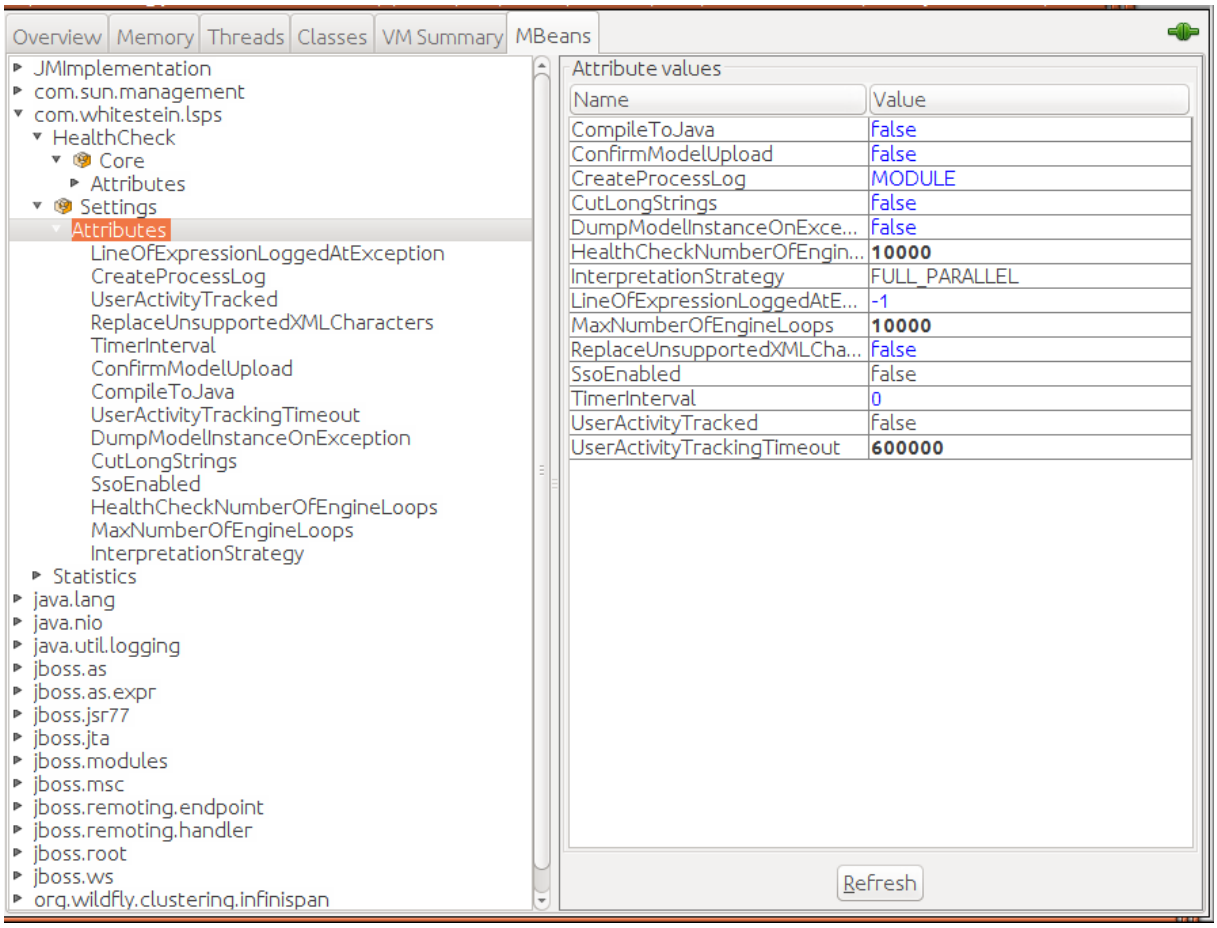


Figure 5.3 LSPS setting in database

5.2.1 General Settings

5.2.1.1 SSO_ENABLED (SsoEnabled): Enabling SSO in UI

When **true** and the application server is configured to use SSO, the application checks if the current user exists in the application and is active when creating a new Vaadin UI.

5.2.1.2 CUT_LONG_STRINGS (CutLongString): Cutting a String in Database

- When **true**, strings longer that the length defined in database are cut.
- When **false**, write of such strings results in an error.

Default setting: *false*

The setting is loaded on server startup and can be changed in the LSPS_SETTINGS table or via JMX. When changed via JMX, the database value remains unchanged.

5.2.1.3 REPLACE_UNSUPPORTED_XML_CHARACTERS (ReplaceUnsupportedXMLCharacters): Replacing Unsupported XML Characters

- When **true**, unsupported XML characters are replaced with the replacement character \uFFFD on model-instance marshalling.
- When **false**, unsupported XML characters cause an exception on model-instance marshalling.

Default setting: *false*

The setting is loaded on server startup and can be changed in the LSPS_SETTINGS table or via JMX. When changed via JMX, the database value remains unchanged.

5.2.2 Model Related Settings

5.2.2.1 ENABLE_DROP_CREATE: Enabling Drop-Create Database-Schema Update

- When **true**, models with the drop-create strategy can be uploaded.
- When **false** and the user attempts to upload a model with the drop-create strategy, the upload fails with an exception.

Default setting: *true*

When changed in the LSPS_SETTINGS table, the change is reflected in runtime immediately.

5.2.2.2 INITIAL_MODELS_SQL: SQL with Model IDS loaded on Launch

SQL that returns IDs of model that should be loaded on server launch.

The setting is loaded only on server launch.

5.2.2.3 CONFIRM_MODEL_UPLOAD (ConfirmModelUpload): Requiring Confirmation on Model Update

- When **true**, the user is prompted to confirm model upload on each upload.

The setting is loaded on server launch and can be changed in runtime via JMX (When changed via JMX, the database value remains unchanged).

Default setting: *false*

5.2.2.4 SERIALIZE_MODEL_UPLOAD: Serializing Models on Upload

- When **serialize**, models are serialized on upload.

Default setting: *serialize*

When changed in the LSPS_SETTINGS table, the change is reflected in runtime immediately.

5.2.2.5 MaxNumberOfEngineLoops and ThresholdNumberOfEngineLoops: Preventing Infinite Looping

A single execution step in a model instance is referred to as a *loop*: A loop moves tokens and checks repeatedly the status of various elements and data to make sure that the data, such as goal statuses, is correct when the loop finishes.

If the loop ends up in an infinite loop, for example, if one condition is switched from true to false within the loop continuously, the server terminates the looping based on the MaxNumberOfEngineLoops and ThresholdNumber↩ OfEngineLoops setting:

- When a loop is repeated for more times than the number defined by ThresholdNumberOfEngineLoops, the NumberOfThresholdLoopExceeded attribute is bumped.
- When a loop is repeated for more time than the number defined by MaxNumberOfEngineLoops, the execution fails with a runtime exception, the transaction is rolled back and the NumberOfThresholdLoopExceeded attribute is bumped.

5.2.3 User Tracking

5.2.3.1 USER_ACTIVITY_TRACKING (UserActivityTracked): Enabling User-Activity Tracking

- When **true**, periods when the user is active are logged in the LSPS_USER_ACTIVITY_TRACK table.

Default setting: *false*

On change, restart the server to apply the change.

5.2.3.2 USER_ACTIVITY_TRACKING_TIMEOUT (UserActivityTrackingTimeout): Setting Timeout for User Activity

- Time period in milliseconds: if a user is idle for the specified time period, logged period of activity is finished (the user is considered inactive).

Default setting: *600000* (10 minutes)

> **Note:** You can access the user-activity data through the `UserTrack` shared record.

5.2.4 Performance

5.2.4.1 INTERPRETATION_STRATEGY (InterpretationStrategy): Setting Interpretation Strategy of a Goal Process

Interpretation strategy of the LSPS Execution Engine

- When set to **FULL_PARALLEL**, Goal conditions are checked whenever any of the Goals changes its status or a token is moved.
- When set to **BPMN_FIRST**, the engine first pushes all the tokens *in Plans* as far as possible and only then checks Goals and their conditions.

Default setting: FULL_PARALLEL

The setting is loaded on server startup and can be changed in runtime via JMX (When changed via JMX, the database value remains unchanged).

5.2.4.2 TIMER_INTERVAL (TimerInterval): Setting Interval for Timed Trigger

The smallest interval for sending a timed trigger in milliseconds. If there are multiple time notifications scheduled for a Model instance (for example, by multiple Timer Intermediate Events), and the time difference between the notifications is smaller that the TIMER_INTERVAL value, the notifications are merged into one (multiple Timer Intermediate Events are notified by a single timer notification)

If set to *0*, timer notifications are not merged.

Default setting: 0

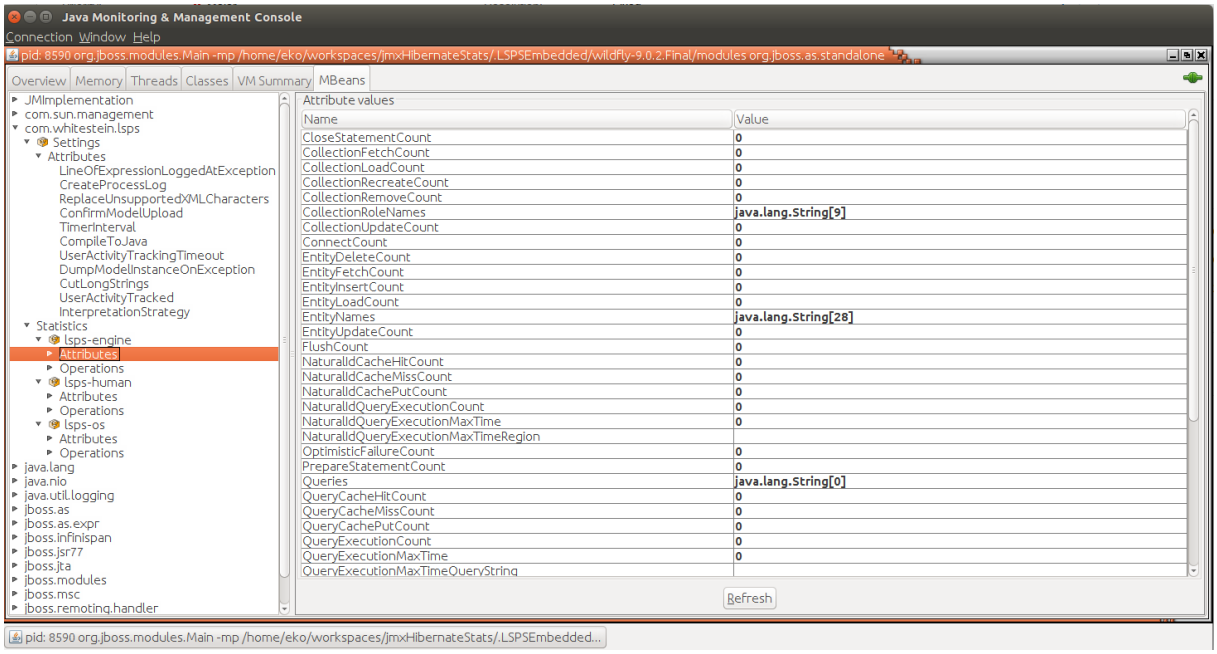
The setting is loaded on server startup and can be changed in runtime via JMX (When changed via JMX, the database value remains unchanged).

5.2.4.3 STATISTICS_ENABLED: Enabling Hibernate Statistics

Availability of Hibernate statistics (equivalent of `hibernate.generate_statistics`)

The option is by default disabled. You can enable it from JConsole: you can enable the setting for individual modules from the *MBeans* tab under *com.whitestein.lsps* > *Statistics* > *MODULE* > *Attributes* > *Statistics↩ Enabled*. To enable the setting for the entire application, modify the *server.properties* file in the application EAR (`lsps-app-ear/lsps-app-ejb.jar/server.properties`)

Do not enable the setting in production environments since it can cause performance issues.



5.2.5 Logging and Exceptions

5.2.5.1 DUMP_MODEL_INSTANCE_ON_EXCEPTION (DumpModelInstanceOnException): Dumping A Module Instance on Exception

If set to true, model instance state is dumped when an exception occurs. The setting is loaded on server startup and can be changed in runtime via JMX (When changed via JMX, the database value remains unchanged).

Default setting: *false*

5.2.5.2 CREATE_PROCESS_LOGS(CreateProcessLog): Creating Process Logs

The setting defines whether events that occur during model-instance execution are logged. The events include such information as when the model and process started, when a process elements was executed, etc.

They are stored in the LSPS_PROCESS_LOGS system-database table. The data is used, for example, by views in the Management Perspective and Management Console.

> **Note:** If you are looking for information on other logging mechanisms, please, refer to FAQ

Possible values:

- MODULE: the setting of the module is respected.
- YES: process logs are created for all modules regardless of the *Create process log* module setting.
- NO: process logs are not created for any modules regardless of the *Create process log* module setting.

Default setting: *MODULE*

To clean old entries from the process log, use the delete-old-process-logs.<DB_NAME>.sql script available in <LSPS-RUNTIME>/scripts/lsps/<YOUR_DB>/

> **Important:** If you set the property to NO, *information about model-instance status and execution history*, such as, when a model instance started, a to-do was submitted, timer event triggered, etc. will not be available. As a result:
>
> - Model Instance details in the Management Perspective and Management Console will not be available.
> - BAM reports might not contain correct data.

5.2.5.3 LINES_OF_EXPRESSION_LOGGED_IN_EXCEPTION (LineOfExpressionLoggedAtException): Numbers of Expression Lines in Exception

Number of lines before and after the statement with the problem included in the stack trace (0 means all: the entire expression is returned). The setting is loaded on server startup and can be changed in runtime via JMX (When changed via JMX, the database value remains unchanged).

Default setting: *-1*

> **Note:** You can display SQL statements issued by PDS to the server in the console to troubleshoot the statements. Refer to information on performance tuning.

5.3 Debugging

Important: In production, the debugging parameters must not be active.

The debugging parameters are passed as JVM properties `-Dkey=value`.

- *com.whitestein.lsps.vaadin.ui.debug*

If true, the modeling ID set on form components is used as Vaadin debug ID on all Vaadin components.

- *lspsDebug*

If true, the Execution Engine runs in debug mode (usually activated in combination with remote socket debugging).

- *lspsProfile*

If set to `true`, `profiler` is enabled.

5.4 Authentication

LSPS uses the authentication mechanism of the underlying application server. The users are authenticated against the LSPSRealm.

If you want to authenticate users against other directory services, modify the configuration of your application server.

For example, to authenticate against LDAP in WildFly, add another login module to the security domain.

Note that you need to make sure that all persons from your directory service have their counterpart in the LSPS: For example, insert a new person to the LSPS using the web service API. Similarly, remove any removed persons.

Example WildFly login module configuration

```
<subsystem xmlns="urn:jboss:domain:security:1.1">
   <security-domains>
       <security-domain name="lspsRealm" cache-type="default">
           <authentication>
               <login-module code="com.whitestein.lsps.security.jboss.LSPSRealm" flag="optional"
                   <module-option name="password-stacking" value="useFirstPass" />
                   <module-option name="dsJndiName" value="/jdbc/LSPS_DS" />
               </login-module>
               <login-module code="org.jboss.security.auth.spi.LdapExtLoginModule" flag="optional
                   <module-option name="password-stacking" value="useFirstPass" />
                   <module-option name="java.naming.factory.initial" value="com.sun.jndi.ldap.Lda
                   <module-option name="java.naming.provider.url" value="ldap://myurl:111" />
                   <module-option name="bindDN" value="cn=SA_LDAPReadOnly,ou=Service Accounts,dc=
                   <module-option name="bindCredential" value="passwd" />
                   <module-option name="baseCtxDN" value="ou=INCOMM,dc=uss,dc=net" />
                   <module-option name="baseFilter" value="(sAMAccountName={0})" />
                   <module-option name="rolesCtxDN" value="ou=INCOMM,dc=uss,dc=net" />
                   <module-option name="roleFilter" value="(sAMAccountName={0})" />
                   <module-option name="roleAttributeID" value="memberOf" />
                   <module-option name="roleAttributeIsDN" value="true" />
```

```
                <module-option name="roleNameAttributeID" value="cn" />
                <module-option name="parseRoleNameFromDN" value="false" />
                <module-option name="allowEmptyPasswords" value="false" />
                <module-option name="searchScope" value="SUBTREE_SCOPE" />
                <module-option name="allowEmptyPasswords" value="false" />
                <module-option name="throwValidateError" value="false" />
            </login-module>
            <login-module code="org.jboss.security.auth.spi.RoleMappingLoginModule" flag="opti
                <module-option name="rolesProperties" value="../standalone/configuration/roles
                <module-option name="replaceRole" value="false" />
            </login-module>
        </authentication>
    </security-domain>
```

Chapter 6

Maintenance

6.1 Cleaning Database Logs

To delete logs from the LSPS system database, run the SQL scripts in the `scripts/lsps/<YOUR_DB>` directory of the LSPS Runtime package.