WHITESTEIN
Technologies

Living Systems® Process Suite

Deployment and Configuration

Living Systems Process Suite Documentation

3.1
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

Main Page

The resources and tools for the 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 LSPS Application, the suite contains the command-line tool for management of the server as well as database migration tools.

1.1 Requirements

Make sure you environment meets the following requirements:

1.1.1 Hardware

Requirements for 50 users, and about 10 concurrent sessions:

- for a **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
- for a **dedicated database server**:
  - *Minimum*: i7 x86, 64bit, 8 core CPU. 16 GB RAM
  - *Recommended*: i7 x86, 64bit, 16 core CPU. 32 GB RAM

1.1.2 Software

The LSPS Application can be deployed in an environment with the following setup options:

- databases:
  - MySQL 5.6
  - MS SQL 2008 R2
  - MS SQL 2014
  - DB2 10.5

  - Oracle 11g R2

- application servers:
  - JBoss EAP 6.4.4
  - JBoss EAP 7.0.0
  - WildFly 9.0.2
  - WildFly 10.0.0
  - WebSphere 8.5.0.2
  - Weblogic 12.1.1

  Important: Make sure that your server has all the necessary security vulnerabilities fixed.

- Java SDK:
  - Oracle JDK 1.7.x
  - Oracle JDK 1.8.x if supported by your application server

  Important: JVM should be run with at least `-Xmx1024m`

1.1.3 Environment Recommendations

Consider having the following environments set up:

- 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;

Chapter 2

Installation

To deploy the LSPS Application, you will need to set up a database and an application server. Make sure to select a compatible combination of database, application server, and LSPS Application.

Continue to Database Setup and Application Server Setup and Application Deployment.

2.1 Database Setup

Make sure you perform the following steps:

1. Create a database; make sure you are using a database supported by LSPS as well as your application server.
2. If applicable, initialize the database with the db-migration tool from the runtime suite (refer to database migration).

   Important: If you are using MySQL with multibyte encoding, such as, UTF-8, set the global `innodb_file_format=Barracuda;set global innodb_large_prefix=on;ROW_FORMAT DYNAMIC`

   Otherwise, the initialization of the LSPS database will fail with the exception:

   ```
   SEVERE: Migration of schema `lsps` to version <DB_VERSION> failed! Please restore backups > and roll back
          database and code!
   ```

   This occurs because MySQL uses the InnoDB engine by default, which restricts column length. Since LSPS makes use of wider columns, you need to substitute this engine with the Barracuda engine.

2.2 Application Server Setup and Application Deployment

You can deploy LSPS on the following application servers:

- JBoss
- WildFly
- Oracle WebLogic
- IBM WebSphere

Make sure the selected application server supports your database.

2.2.1 JBoss

2.2.1.1 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, that is, *mysql-connector-java-<VERSION>-.jar* to $EAP_HOME/modules/com/mysql/main. Then create $EAP_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-8.0.15.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_HOME/modules/com/microsoft/sqlserver/mainr. Additional steps may be required to enable the XA transaction support on Microsoft SQL server. 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). You may also need to install MSDTC in Windows (Add/remove Windows components -> Application Server -> Details -> Enable network DTC access).
     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 persist messages: `<server name="default" persistence-enabled="true">`

   (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-java
     <driver>mysqlxa</driver>
     <xa-datasource-property name="URL">jdbc:mysql://localhost:3306/lsps</xa-datasource-prop
     <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-ja
       <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-ja
  <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 (make sure the driver is 5.5 compatible):

```
<driver name="mysqlxa" module="com.mysql">
  <!-- for version 8: -->
  <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. 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
```

8. 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.

9. 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 .
   - Adapt the acquisition timeouts to reasonable values to prevent long waiting deadlocks in case of lack of resources. Set data-source connection pool the maximum to MDB pool size + the 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"/>
```

10. Configure networking (optional).

    By default JBoss binds to the local host. You can change the configuration in $EAP_HOME/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>
```

11. Set up the JBoss option 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:MaxMetaspaceSize=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.e
```

    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.

12. Deploy the LSPS Application EAR to the server.

    - To deploy the default LSPS Application, deploy lsps-application-<VERSION>.ear located in the lsps-runtime directory.
    - To create and deploy a customized LSPS Application, install PDS with SDK and proceed as instructed here.

13. 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>
```

14. Run JBoss.

```
$EAP_HOME/bin/standalone.sh
```

15. Open your browser and go to url http://localhost:8080/lsps-application

16. Authenticate with the credentials:

    - user: admin

- password: admin

17. To shut down JBoss, use the command-line administration client:

```
$EAP_HOME/bin/jboss-cli.sh --connect :shutdown
```

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

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

2.2.1.2 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, that is, *mysql-connector-java-<VERSION>.jar* to $EAP_HOME/modules/com/mysql/main. Then create $EAP_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-8.0.15.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_HOME/modules/com/microsoft/sqlserver/mainr. Additional steps may be required to enable the XA transaction support on Microsoft SQL server. 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). You may also need to install MSDTC in Windows (Add/remove Windows components -> Application Server -> Details -> Enable network DTC access).
  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 with `<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>
~
      <connectors>
        <http-connector name="http-connector" socket-binding="http">
           <param key="http-upgrade-endpoint" value="http-acceptor"/>
         </http-connector>
```

   (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_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-java
  <driver>mysqlxa</driver>
  <xa-datasource-property name="URL">jdbc:mysql://localhost:3306/lsps</xa-datasource-prop
  <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-ja
  <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-ja
  <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 (make sure the driver is 5.5 compatible):

```
<driver name="mysqlxa" module="com.mysql">
  <!-- for version 8: -->
  <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. 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
```

8. 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>
```

9. 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.

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 $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 .
    - Adapt the acquisition timeouts to reasonable values to prevent long waiting deadlocks in case of lack of resources. Set the data-source connection pool maximum to MDB pool size + the 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"/>
```

11. 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>
```

12. Set up the JBoss option 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.e
```

    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.

13. Deploy the LSPS Application to $EAP_HOME/standalone/deployments:

- To deploy the default LSPS Application, deploy lsps-application-<VERSION>.ear located in the lsps-runtime directory.
- To create and deploy a customized LSPS Application, install PDS with SDK and proceed as instructed here.

14. Run JBoss.

```
$EAP_HOME/bin/standalone.sh
```

15. Open your browser and go to url http://localhost:8080/lsps-application
16. Authenticate with the credentials:
    - user: admin
    - password: admin
17. To shut down JBoss, use the command-line administration client:

```
$EAP_HOME/bin/jboss-cli.sh --connect :shutdown
```

18. Optionally, exclude the useless vaadin warning from the log:

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

2.2.1.3 Troubleshooting Data Source

2.2.1.3.1 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] Local
       XARecoveryModule.xaRecovery got XA exception javax.transaction.xa.XAException, XAException.XAER_RMERR
```

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 JBoss 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;
```

2.2.1.3.2 InvalidConnectionAttributeException with MySQL

If on MySQL with the JDBC driver, you might get the following exception:

```
Caused by: com.mysql.cj.exceptions.InvalidConnectionAttributeException:
The server time zone value 'CEST' is unrecognized or represents more than one time zone.
You must configure either the server or JDBC driver (via the serverTimezone configuration property)
to use a more specific time zone value if you want to utilize time zone support.
```

To remedy the situation, set the time zone:

- in the database if one timezone is required, for example, 'SET GLOBAL time_zone = '+3:00';'
- in the data source URL if multiple zones are required, for example,

```
<xa-datasource-property
   name="URL">jdbc:mysql://localhost:3306/lsps?
              useUnicode=true&
              characterEncoding=utf-8
              &useJDBCCompliantTimezoneShift=true
              &useLegacyDatetimeCode=false
              &serverTimezone=UTC
</xa-datasource-property>
```

2.2.2 WildFly

2.2.2.1 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, 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-8.0.15-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. Additional steps may be required to enable the XA transaction support on Microsoft SQL server. 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). You may also need to install MSDTC in Windows (Add/remove Windows components -> Application Server -> Details -> Enable network DTC access).
  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 with `<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>
~
      <connectors>
        <http-connector name="http-connector" socket-binding="http">
           <param key="http-upgrade-endpoint" value="http-acceptor"/>
       </http-connector>
```

   (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-java
  <driver>mysqlxa</driver>
  <xa-datasource-property name="URL">jdbc:mysql://localhost:3306/lsps</xa-datasource-prop
  <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-ja
  <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-ja
  <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 (make sure the driver is 5.5 compatible):

     ```
     <driver name="mysqlxa" module="com.mysql">
       <!-- for version 8: -->
       <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. 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>
   ```

8. 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.

9. 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.
   - Adapt the acquisition timeouts to reasonable values to prevent long waiting deadlocks in case of lack of resources. Set data-source connection pool the maximum to MDB pool size + the default pool size at least.

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

10. Enable stateless session beans pooling: >

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

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-ipv4-address/>
</interface>
<interface name="public">
  <any-ipv4-address/>
</interface>
<interface name="unsecure">
  <any-ipv4-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: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 WildFly.
    ```
    JAVA_OPTS="$JAVA_OPTS -Dorg.eclipse.emf.ecore.EPackage.Registry.INSTANCE=org.eclipse.e
    ```
    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.

13. Deploy the application lsps-application-$LSPS_VERSION.ear to $WF_HOME/standalone/deployments.
14. Run WildFly.

```
$WF_HOME/bin/standalone.sh
```

15. Open your browser and go to url http://localhost:8080/lsps-application
16. Authenticate with the credentials:
    - user: admin
    - password: admin
17. To shut down WildFly, use the command-line administration client:

```
$WF_HOME/bin/jboss-cli.sh --connect :shutdown
```

18. Optionally, exclude useless vaadin warnings from the log:

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

19. Deploy the LSPS Application to $EAP_HOME/standalone/deployments:
    - To deploy the default LSPS Application, deploy lsps-application-<VERSION>.ear located in the lsps-runtime directory.
    - To create and deploy a customized LSPS Application, install PDS with SDK and proceed as instructed here.

2.2.2.2 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, 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-8.0.15-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. Additional steps may be required to enable the XA transaction support on Microsoft SQL server. 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). You may also need to install MSDTC in Windows (Add/remove Windows components -> Application Server -> Details -> Enable network DTC access).
     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 the lsps 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-java
  <driver>mysqlxa</driver>
  <xa-datasource-property name="URL">jdbc:mysql://localhost:3306/lsps</xa-datasource-prop
  <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-ja
  <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-ja
  <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 (make sure the driver is 5.5 compatible):

```
<driver name="mysqlxa" module="com.mysql">
  <!-- for version 8: -->
```

```
      <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. 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>
```

8. 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.

9. 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.
   - Adapt the acquisition timeouts to reasonable values to prevent long waiting deadlocks in case of lack of resources. Set data-source connection pool the maximum to MDB pool size + the 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-
   <strict-max-pool name="mdb-strict-max-pool" max-pool-size="20" instance-acquisition-ti
 ~
   <thread-pools>
     <thread-pool name="default">
       <max-threads count="10"/>
```

10. 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>
```

11. 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:

    i. Configure EMF framework to work properly on WildFly:

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

    ii. 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.

12. Deploy the LSPS Application EAR, lsps-application-$LSPS_VERSION.ear, to $WF_HOME/standalone/deployments↩ :

    - To deploy the default LSPS Application, deploy lsps-application-<VERSION>.ear located in the lsps-runtime directory.
    - To create and deploy a customized LSPS Application, install PDS with SDK and proceed as instructed here.

13. Run WildFly.

```
$WF_HOME/bin/standalone.sh
```

14. Open your browser and go to url http://localhost:8080/lsps-application

15. Authenticate with the credentials:

- user: admin
- password: admin

16. To shut down WildFly, use the command-line administration client:

```
$WF_HOME/bin/jboss-cli.sh --connect :shutdown
```

17. Optionally, exclude useless vaadin warnings from the log:

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

2.2.2.3 Troubleshooting Data Source

2.2.2.3.1 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] Local
       XARecoveryModule.xaRecovery got XA exception javax.transaction.xa.XAException, XAException.XAER_RMERR
```

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;
```

2.2.2.3.2 InvalidConnectionAttributeException with MySQL

If on MySQL with the JDBC driver, you might get the following exception:

```
Caused by: com.mysql.cj.exceptions.InvalidConnectionAttributeException:
The server time zone value 'CEST' is unrecognized or represents more than one time zone.
You must configure either the server or JDBC driver (via the serverTimezone configuration property)
to use a more specific time zone value if you want to utilize time zone support.
```

To remedy the situation, set the time zone:

- in the database if one timezone is required, for example, 'SET GLOBAL time_zone = '+3:00';'
- in the data source URL if multiple zones are required, for example,

```
<xa-datasource-property
   name="URL">jdbc:mysql://localhost:3306/lsps?
               useUnicode=true&amp;
               characterEncoding=utf-8
               &amp;useJDBCCompliantTimezoneShift=true
               &amp;useLegacyDatetimeCode=false
               &amp;serverTimezone=UTC
</xa-datasource-property>
```

2.2.3 Oracle WebLogic

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

1. 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).
2. 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.JAXBRIContextFactor
   ```

3. Install WebLogic.
4. 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`).
5. Copy LSPS security module *lsps-security-weblogic-$LSPS_VERSION.jar* to $WL_HOME/server/lib/mbeantypes.
6. Start WebLogic.
7. 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
   ```

8. 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)

9. 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 module in the step 2
       correctly)
```

10. Reorder authentication providers so that LSPS_AUTHENTICATOR is the first provider and 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.
11. Restart WebLogic.
12. 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
```

13. Deploy $WL_HOME/common/deployable-libraries/jsf-1.2.war as a library to the WebLogic server.
14. Deploy the application lsps-application-$LSPS_VERSION.ear to the WebLogic server.
15. Open http://localhost:7001/lsps-application in your browser and log in as *admin* with the password *admin*.

2.2.3.1 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.

2.2.4 IBM WebSphere

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

1. Ensure JPA2 is supported on app server (Feature Pack for OSGi Applications and JPA 2.0 is installed).
2. Install WebSphere.
3. 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/JBAPP-2613, also https://www.ibm.com/developerworks/community/forums/html/topi or properties `fullyMaterializeLobData=true`, `fullyMaterializeInputStreams=true`, `progressiveStreaming=2`, `progresssiveLocators=2`)
- downgradeHoldCursorsUnderXa=true
- resultSetHoldabilityForCatalogQueries=1

iii. Test the connection.

4. 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/LSPS_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
```

5. 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

6. 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>
```

7. Deploy JSF 2.0 and define it in shared libraries (although WAS 8.5 contains JSF 2.0 implementation - My↩ Faces - for the selenium tests to work correctly, Mojarra implementation is required).

   (a) Go to Environment/Shared libraries and add a shared library for JSF 2.0 (API and implementation JARs are needed). Provide class path according to their location.

   (b) Select the isolated class loader option.

   The LSPS application deployment then needs a shared library reference to JSF 2.0 shared library definition created above.

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 application lsps-application-$LSPS_VERSION.ear to WebSphere.
10. Open your browser and browse to url http://localhost:9081/lsps-application, user: admin, password: admin.

2.2.4.1 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.

Chapter 3

Configuration

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

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

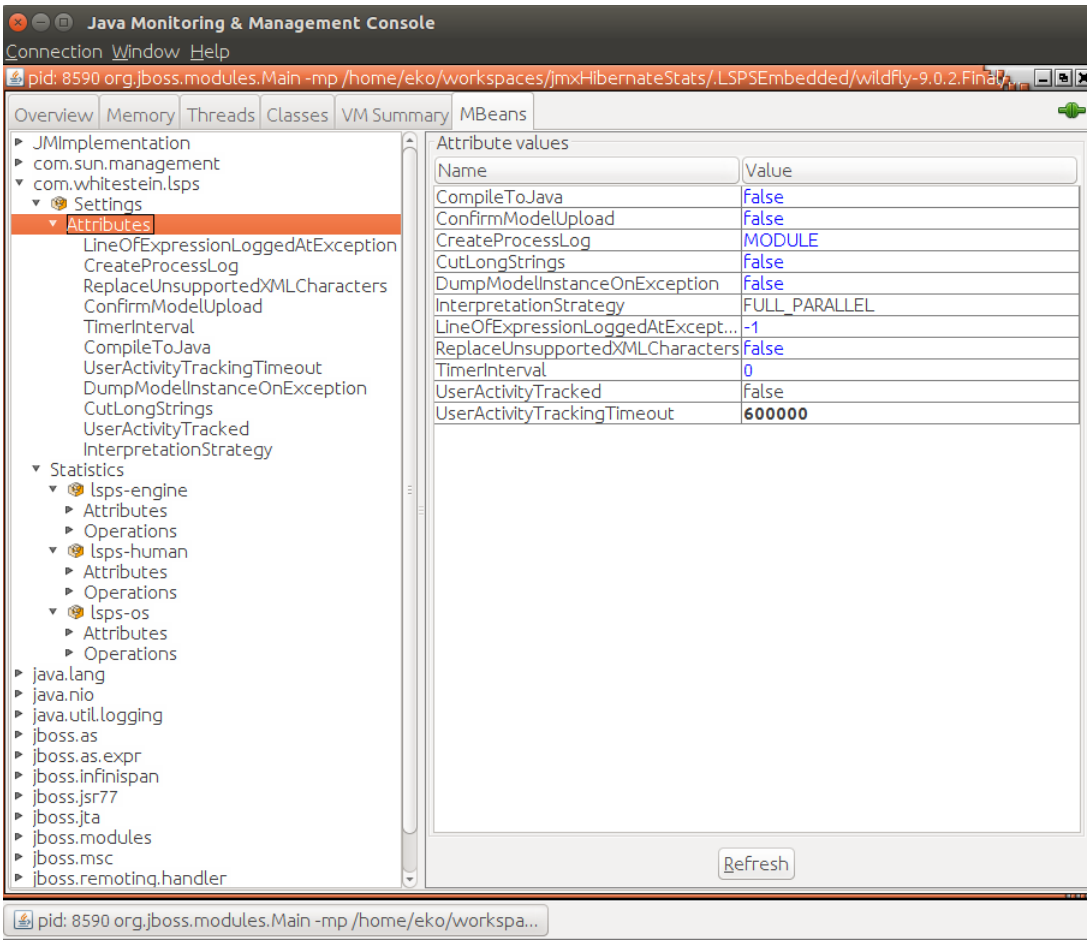


Figure 3.1 LSPS Application settings in jconsole

3.1 General Database Settings

3.1.1 CUT_LONG_STRINGS (CutLongString)

- 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 runtime via JMX (When changed via JMX, the database value remains unchanged).

3.1.2 REPLACE_UNSUPPORTED_XML_CHARACTERS (ReplaceUnsupportedXMLCharacters)

- 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 runtime via JMX (When changed via JMX, the database value remains unchanged).

3.2 Model Related Settings

3.2.1 ENABLE_DROP_CREATE

- 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.

3.2.2 INITIAL_MODELS_SQL

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

The setting is loaded only on server launch.

3.2.3 CONFIRM_MODEL_UPLOAD (ConfirmModelUpload)

- 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*

3.2.4 SERIALIZE_MODEL_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.

3.3 User Tracking

3.3.1 USER_ACTIVITY_TRACKING (UserActivityTracked)

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

Default setting: *false*

3.3.2 USER_ACTIVITY_TRACKING_TIMEOUT (UserActivityTrackingTimeout)

- 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.

3.4 Performance

3.4.1 INTERPRETATION_STRATEGY (InterpretationStrategy)

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 as far as possible in Plans 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).

3.4.2 TIMER_INTERVAL (TimerInterval)

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

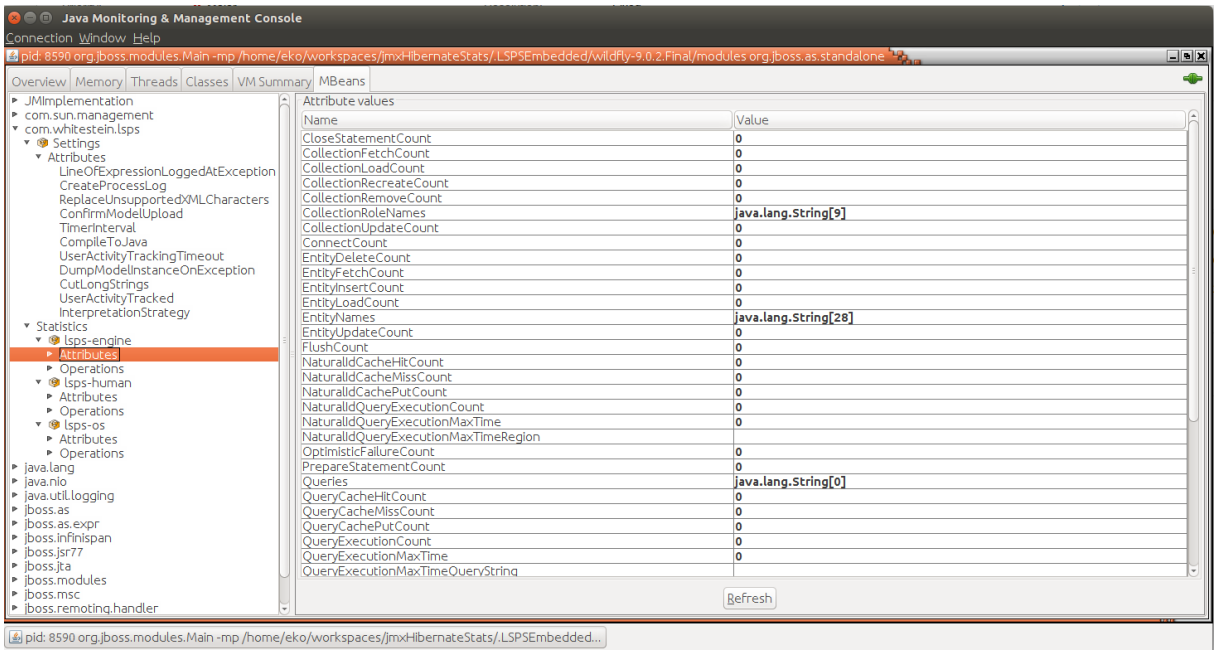
The setting is loaded on server startup and can be changed in runtime via JMX (When changed via JMX, the database value remains unchanged).

3.4.3 STATISTICS_ENABLED

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* > *StatisticsEnabled*. 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.



3.5 Logging and Exceptions

3.5.1 DUMP_MODEL_INSTANCE_ON_EXCEPTION (DumpModelInstanceOnException)

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*

3.5.2 CREATE_PROCESS_LOGS (CreateProcessLog)

Whether to allow entry of log events in to the LSPS_PROCESS_LOG table. The events include information about model and process instantiation, any state changes of process elements, etc.

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.

For production environments, consider using the NO setting.

Default setting: *MODULE*

> **Important:** If process logs are not created, BAM reports will not contain relevant data and no details on model instances is available (the data is used, for example, in the Model Instance views of the Management perspective). The setting is loaded on server startup and can be changed in runtime via JMX (When changed via JMX, the database value remains unchanged).

3.5.3 LINES_OF_EXPRESSION_LOGGED_IN_EXCEPTION (LineOfExpressionLoggedAtException)

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*

3.6 Debugging

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

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

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

If true, modeling ID 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.

Chapter 4

Migration

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

Generally, you will perform migration as follows:

1. Migrate the system database.
2. If you are changing the database version, check whether the database schema is compatible with the previous schema; for example, schema of MSSql Server 2012 uses sequences, which were not available in previous versions and the schema used a table instead (in this case, drop the table LSPS_SEQ and create the sequence LSPS_SEQ with `create sequence LSPS_SEQ start with <VALUE> increment by 1000`; <VALUE> is the value from the old LSPS_SEQ table).
3. Run an application server with the LSPS Application over the migrated database and check if the running model instances are in correct state.
4. If you need to adapt running model instances, use the `Model Update` feature.
5. Manually migrate your Modules.
6. If you have a customized Application User Interface:
   (a) Generate a new `LSPS Application` and apply to it the changes from your current application. Alternatively, consider the API changes in the new application and apply them to your application.
   (b) Make sure to update the main `pom.xml` file:
   - update the parent version
     ```
     <parent>
       <groupId>com.whitestein.lsps</groupId>
       <artifactId>lsps</artifactId>
       <version>3.1.1027</version>
     </parent>
     ```
   - update the lsps version
     ```
     <properties>
       <lsps.version>3.1.1027</lsps.version>
     </properties>
     ```
     If you want to update the main pom.xml file from PDS, go to **File** > **Open File**, navigate to the root directory of the application and select file.

   (c) Rebuild the application.

In case you experience problems in any of the steps, make sure to remedy the situation before proceeding to the next step.

4.1 Migrating Modules from 2.7

- In version 2.7, queries and like could have used the _ and % wildcards. To migrate your queries and like expressions change _ to ? and % to *.

  If you want to retain the behavior set the COMPATIBILITY_VERSION column V_2_7. For example

```
UPDATE LSPS_MODULES SET COMPATIBILITY_VERSION = 'V_2_7' WHERE ID = 24000
```

  and restart the server.

  Important: Note that the mode is applied only in queries with the metadata nonStrictEscape.

4.2 Migrating Modules from 3.0

Modules created in LSPS 3.0 might require manual migration due to the following changes:

- **Characters in the name of modules and resources not allowed**

  Do not use the characters # / ? : \ * " < > | in the names of modules and resources (LSPS-7746)

- **Standard Library task types reflection records added**

  Task types of the Standard Library now generate their Activity records that have the same name as the task type. Make sure that names of records and tasks in your modules do not clash with these Activity record names (LSPS-7732).

- **Instantiation of processes with Activities with no incoming Flows**

  Previously, if a process contained several Activities without incoming Flows, the system instantiated a process for each Activity and each instance passed a token to one of the Activities with no incoming Flows. Now such a process is instantiated once and tokens are passed to all Activities with no incoming Flow at once (LSPS-7653).

- **Non-interruptible Boundary Cancel Intermediate Event removed**

  Boundary Cancel Intermediate Events are no longer interruptible. If a Process contains such an event, redesign the Process so that it uses the Boundary Cancel Intermediate Events as interruptible (LSPS-7580).

- **UIComponent no longer extends the Vaadin component**

  Since UIComponent no longer extends the Vaadin component, you need to migrate all Java custom components by removing the fireUIEvent() method and implementing the function "public AbstractComponent getWidget() { return this; }" (LSPS-7350).

- **JSF Application User Interface and Process Management Console removed**

  The JSF versions of Application User Interface and Process Management Console have been removed. Only Vaadin versions are now delivered and supported. All models that use form item based UI must be remodeled to UI Component based UI (LSPS-7097).

4.3 Database Migration

Important: Before running the script, make sure to back up your database and read the information on Migration on LSPS Forum.

To migrate the LSPS database to a newer version, do the following:

- 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 required by the new Runtime Suite.

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

- Otherwise you need to do the following:

  1. Get information on your current database version: open <YOUR_OLD_RUNTIME_SUITE>/db-migration/lsps-db-migration-<VERSION>.jar> db > migration:
     - Check the most recent java migration class in the location, for example, *V3_1_0_010__Upgrade.class*
     - 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 located in the db-migration directory in <YOUR_NEW_LSPS_RUNTIME_SUITE> 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
```

4.3.1 Migrate Script Parameters

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

databaseUrl∗ URI of the database with LSPS runtime data

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.