Database Conventions and Datasources

Datasources

LAMS uses JNDI datasources. This allows changes to database configuration (e.g. using one database vs multiple databases) without making large application changes.

LAMS has three different JNDI (non-xa) datasources.

1. **lams-ds** (JNDI name java:jdbc/lams-ds) Used for all core and services.
2. **tool-ds** (JNDI name java:jdbc/tool-ds) Used for all tools.
3. **quartz-ds** (JNDI name java:jdbc/quartz-ds) Used by Quartz (for scheduling tasks).

At the moment, all datasources point to the same database. This may change in the future.

If you are a tool developer then we ask you to put your tables in the LAMS tool datasource and follow the naming conventions given below. Do not use your own database unless it is absolutely necessary - not only does it complicate the transactional environment but it also creates more work for system administrators - they will then have your database to back up as well as the LAMS tool database.

> Do not use a local JDBC datasource to access the LAMS database in your deployed code.

JBOSS Configuration

The JBOSS configuration of the three datasources is part of the core LAMS build.

The configuration is done in the mysql-ds.xml file, stored in lams_common\conf\unix\jboss\service and lams_common\conf\windows\jboss\service. The file is copied to the server's deploy directory.

Note: if you are using a DB other than "lams" or different username/password you will need to edit this file. This file should also be modified in performance testing to changes values such as the pool size.

Null Dates
When you save a null date into MySQL, a null date is not stored in the database. Instead, a series of 0's are written to the field. To ensure that fields in the Hibernate POJOs do not contain this 0 based date when the object is read from the database, the db connection url has "zeroDateTimeBehavior=convertToNull" appended to the end of the connection.

autoreconnect

It has been suggested that we use autoReconnect=true. However the writer of the MySQL driver prefers that it is not used, and there is some doubt as to whether it is safe in a transactional mode. As LAMS 1.0 worked fine without this parameter, will not use it in 2.0 until a specific need for the value is established.

For more details:
http://www.theserverside.com/news/thread.tss?thread_id=24137#111947
http://wiki.jboss.org/wiki/Wiki.jsp?page=SetUpAMysqlDatasource

Build File

The build scripts also have the datasources defined, so that the ant builds can create tables, insert data, etc. As these scripts run in Ant, they do not use the application datasources. The settings are configured in lams_build/common.properties. Sample settings are given below. ${sharedlib} is defined as the lib directory in lams_build and ${basedir} is the base directory of a project.

```
# database related properties
db.name=lams
db.driver=com.mysql.jdbc.Driver
db.url=jdbc:mysql://localhost/$\{db.name}?characterEncoding=utf8&zeroDateTimeBehavior=
convertToNull
db.username=lams
db.password=apassword
db.driver.jar=$\{sharedlib}/mysql/mysql-connector-java-3.1.7-bin.jar
db.scripts=$\{basedir}/db/sql
```

Spring Context Files

In order to use these datasources in Spring, add the following entry to your application context file:

```
<!-- JNDI DataSource: uses tool-ds DataSource set up in app server -->
    <bean id="toolDataSource"
class="org.springframework.jndi.JndiObjectFactoryBean">
        <property name="jndiName">
            <value>java:jdbc/tool-ds</value>
        </property>
    </bean>

    <!-- Hibernate SessionFactory -->
    <bean id="crSessionFactory"
class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
        <property name="dataSource"><ref
local="toolDataSource"/></property>
        .......
    </bean>
```

JUNIT Testing

You will need to use a local JDBC datasource for junit testing as the JNDI datasources are not available within the junit environment. For accessing the content repository in unit testing, use the localApplicationContext.xml file found in the content repository jar file, rather than the usual file. This will select a local datasource for the content repository.

Database conventions

Table names

All table names may be a maximum of 30 characters long. This is for Oracle/Sybase compatibility.

A tool's tables in the database should be named tl_<signature>_<rest of name>. For example, a survey tool table is tl_lasurv10_survey_content. The Tool Signatures are kept short to allow the database table names to be short.

DDL and DML Scripts

LAMS database tables are modelled using Clay (an Eclipse plug-in) and then Clay is used to generate the SQL script.

The data model and data creation scripts are in the lams_common project.

Each tool has its own data model and table creation scripts. Each tool project has db directory, which has two subdirectories model (in which the Clay model goes) and sql (in which all sql scripts for the tool go).

When running "Generate SQL (Create Database) Script" to generate the DDL script, turn on the following options on the second dialogue box page:

- Create Table
- Create Index
- Except Indexes that Duplicate a Unique Key
- Comment on Table
- Comment on Column
- Comment on Index
- Never Use Schema Names.

The final setting stops the database name being put into the scripts. We do not want:

```
CREATE TABLE lams.tl_xxx_yyyyyyy
```

We want:

```
CREATE TABLE tl_xxx_yyyyyyy
```

This makes it much easier to change the database name! The utilities that call the scripts are responsible for setting the database.

Configuration

LAMS 2.0 uses MySQL 5. It uses JNDI datasources (defined in mysql-ds.xml) and is configured to use UTF-8 and TRANSACTION_READ_COMMITTED. We also use STRICT_TRANS_TABLES to ensure that the database behaviour is as consistent as possible with other DBMS systems.

UTF-8

All databases/tables must support UTF-8. This is required to support different languages in LAMS.

To achieve this:

1. lams db must be created to use UTF-8: CREATE DATABASE lams2 DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;
2. Any JDBC connections for testing or build scripts must be told to use UTF-8 e.g. jdbc:mysql://server:3306/lams?characterEncoding=utf8
3. The JNDI datasources (in mysql-ds.xml) must be configured for UTF-8

Transaction Isolation Read Committed.

Secondly the transaction isolation level for InnoDB should be set to READ-COMMITTED: this seems to occur out of the box for Win32 but the linux version (and I am assuming other UNIX-like OSs) defaults to REPEATABLE-READ. In order to change this for linux you need to create or edit the config file as follows:

```
[mysqld]
transaction-isolation=READ-COMMITTED
```

Note that according to the documentation you should be able to change the system variables to achieve these changes (or at least make changes that produce the same results) using the SET GLOBAL <variable>=<value> commands like we do in MySQL 4.0.
Unfortunately the SET GLOBAL does not seem to work for many of the system variables 4.1 (though it doesn't produce an error) and we don't know if it works for 5.0.

Zero Date Time

There is an issue with the JDBC driver and null dates. We have configured the JNDI datasources to use "convert to null" to ensure that any null date fields in the database appear as null in Java, rather than as the 0 date.

To achieve this:

1. Any JDBC connections for testing or build scripts must be told to use convert the dates. e.g. jdbc:mysql://server:3306/lams?characterEncoding=utf8&zeroDateTimeBehavior=convertToNull
2. The JNDI datasources (in mysql-ds.xml) must be configured to convert the dates.

Installing MySQL

UTF-8

MySQL 5 Windows version comes with 3 packages, two of which use Windows auto-installer. Select Most International option, which sets the character encoding to UTF-8.

Conf files

my.conf is not used for Windows but the conf file is: <MySQL directory>\MySQL Server 5.0\my.ini

Add the lines below to the my.ini and restart mysqld.

```
[mysqld]
transaction-isolation=READ-COMMITTED
```

Check that the entry for sql-mode in mysql configuration file (on Windows this is <mysql install directory/my.ini) includes STRICT_TRANS_TABLES. We suggest:

```
sql-mode="STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION"
```

This is a better location for the ini file, since this allows different instance of MySQL to use different conf.

Unix/Linux version conf location maybe still at: /etc/my.conf (I have not confirmed this)

If there is no conf file, you can make one by copying one of the conf files inside the MYSQL_HOME or the template, and modify it appropriately.

Service or stand-alone

Windows Service is suited for production.

```
mysqld --install (add the service)

start/stop from service control panel

mysqld --remove (remove the service)
```

Linux/Solaris has different syntax for daemon/service

Stand-alone is suited for development.

```
mysqld -u root --console (prints errors on the console)

mysqladmin -u root -p shutdown
```

Linux/Solaris has different syntax (safe-mysqld?)

Securing the server

You must set the root user password

```
mysqladmin -u root -password 'yoursecret'

To change it: mysql -u root -p

SET PASSWORD FOR root@localhost=PASSWORD('newsecret');
```

If you forget the root password, you can re-start the server by skipping the grant table and manually change it.

```
killall mysqld (or stop the Service)
mysqld -u root --console --skip-grant-tables
mysql

USE mysql;
UPDATE user SET password=password("newpassword") WHERE user="root";
-- Don't forget the WHERE clause!!!
flush privileges;
quit

mysqladmin -u root shutdown
mysqld -u root --console
```