Change request 3 log

1. Concept Location

Step #	Description	Rationale
1	We logged into the system, went to “users” tab and tried to change the admin password	To reproduce the “java.sql.SQLDataException” error
2	We analyzed the error: During the save user operation there was an attempt to retrieve a value from a database column of type 'VARCHAR', but the actual value stored in the database for that column was null.	Error understanding
3	We inspected the UserDao	We selected this class because we identified that error was a Data Exception for the User. So we knew we needed to inspect a Data Access Object which interacts with the persistent storage/ database.
4	Inspected methods of the UserDao till we encountered the saveUser method.	The error occurred during “save user” action
5	Inspected the updateUser method of UserDao.	We noticed that the saveUser called updateUser when it is an existing User ID.
6	We marked the updateUser method as located	We confirmed this method has get calls for the different columns to the database, so this method had to be modified.

Time spent (in minutes): 20

Classes and methods inspected:

- src\com\serotonin\mango\db\dao\UserDao.java
  - public void saveUser(User user)
  - void updateUser(User user)

2. Impact Analysis

Step #	Description	Rationale
1	*We checked where UserDao.saveUser() was being called.*	*To track the classes that could be impacted by the change.*
2	*First Usage:* DatabaseAccess.java *Creates a default user. Discard this.*	*This doesn’t update a user but inserts. So, no impact*
3	Second Usage: UsersDwr.java 1. *One usage is in saveUserAdmin method Impacts. This method calls the UserDao.saveUser to update the admin details including the password.* 2. *Second usage is in saveUser method Impacts. This method calls UserDao.saveUser to update the non admin user details including the password.* *Both these places were assessed for impact and no change is needed in either of these places.*	*The change that we make to the UserDao.*updateUser will apply to these usages too. It will not have a negative impact or create any bugs at these sites because our fix would only be handling the null value error.

4	Third Usage: ImportTask.java *Impacts. This imports any json task pasted in the area provided. The json could be that of a User. This imports User into the database or updates the existing User in the database if it is already present.* *No change needed here.*	*The change that we make to the UserDao.*updateUser will apply to this site too. It will not have a negative impact or create any bugs here because our fix would only be handling the null value error.

Time spent (in minutes): 10

List of Usage (No change needed in any):

DatabaseAccess.java

public void initialize

UsersDwr.java

public DwrResponseI18n saveUser

public DwrResponseI18n saveUserAdmin

ImportTask.java

private void importUser

3. Actualization

Step #	Description	Rationale

1	*We add a third argument to ejt.update:*
	new int[] {
	Types.VARCHAR, Types.VARCHAR, Types.VARCHAR, Types.VARCHAR, Types.VARCHAR, Types.VARCHAR, Types.VARCHAR, Types.INTEGER, Types.VARCHAR, Types.INTEGER }
		The third argument is an array specifying the SQL data types of the parameters in the SQL statement. This array corresponds to the parameter placeholders (?) in the SQL query defined in the first argument. This would bypass the null value exception by helping the JDBC driver handle data conversion correctly.

Time spent (in minutes): 10

4. Validation

Step #	Description	Rationale
1	*save Admin User* *Tested the scenario where admin password is changed in the application.*	*Test scenario passed. Password updated successfully*
2	*save non-admin User* *Tested the scenario where non-admin user password is changed in the application.*	*Test scenario passed. Password updated successfully*
3	*Import User* *Tested import User task under the Import/Export tab*	*Test scenario passed*

Time spent (in minutes): 5

5. Summary of the change request

Phase	**Time (minutes)**	**No. of classes inspected**	**No. of classes changed**	**No. of methods inspected**	**No. of methods changed**
Concept location	20	1	0	2	1
Impact Analysis	10	3	0	4	0
Prefactoring					
Actualization	10	1	0	1	1
Postfactoring					
Verification(Use case scenario testing)	5	0	0	0	0
Total	45	5	0	6	2

6. Conclusions

For this change, concept location was easy because the exception was a data exception, which made it easier to narrow the search to a DAO layer. Impact analysis took more time comparatively. We had to run the import task functionality and analyze each of the impact sites and experiment with the functionality to build understanding.