Data Transactions in Servoy

Data manipulations in Servoy happen inside an *in-memory* transaction. When a record is created or modified, either by user action or by developer, nothing is committed to the database immediately. The Servoy Client tracks all newly-created and modified records, including which columns have changed, their former and latter values. As records are added or modified, the amount of information stored in the In-Memory transaction accrue until they are committed or *rolled back*. The duration of this In-Memory transaction can be short or long depending on the client's configurable *Auto Save* setting.

In This Chapter

Auto Save: ON

By default, every Servoy client is started with the Auto Save setting initialized to on/true. This means that the In-Memory transaction is typically very short as changes are committed automatically as the user navigates the client session. Specific actions like clicking in a form's area, navigating to a different form, clicking a button, etc. all trigger a save event. Auto Save is ideal for situations where the user is intended to be able to make edits freely.

Auto Save: OFF

The developer may optionally set the Auto Save setting to off/false. This means that the length of the In-Memory transaction is controlled by the developer. As changes accrue, they are never committed until the developer programmatically invokes a save event. It is ideal to disable Auto Save for scenarios where the user is intended to perform edits in a controlled situation where a group of edits may be saved or rolled back all together.

The Auto Save setting can be programmatically changed throughout the duration of the client session to accommodate different modes for different editing scenarios.

See also the Database Manager's setAutoSave method in the programming reference guide.

The Anatomy of the In-Memory Transaction

Servoy provides a robust data API, giving the developer full access to the In-Memory transaction, which consists of a listing of all record objects that were added or modified. For each of these record objects, there is a listing of every column whose value was changed. For every modified column, there is a reference to the value before and after the edit. The transaction API also allows developers to distinguish between records that are newly-created and do not yet exist in the database, versus records that already exist in the database, but have outstanding edits.

See also the Database Manager's getEditedRecords and the JSRecord's getChangedData methods.

Saving Data Changes

A developer can programmatically issue a save event, causing the contents of the In-Memory transaction to be automatically translated into instructions to insert/update database tables. A developer can optionally invoke a save event for a specific record only, leaving the rest of the transaction unaffected.

If for some reason one or more records were unable to be saved (i.e. due to a back-end database violation, etc.), the transaction will also keep track of *Failed Records* and their associated errors.

See also the Database Manager's saveData and getFailedRecords methods, as well as the exception property of the JSRecord.

Rolling Back Data Changes

A developer can programmatically issue a command to *rollback* the contents of the entire In-Memory transaction, causing newly created records to be removed and modified records to be reverted to their state prior to the start of the In-Memory transaction. The developer can optionally choose to rollback changes for a specific record, leaving the rest of the transaction unaffected.

See also the Database Manager's rollbackEditedRecords method, as well as the rollbackChanges method of the JSRecord.

What about Deleting Records?

It is important to note that record deletes are not part of the In-Memory transaction. When a record is deleted, the instructions are sent to the database immediately and the delete cannot be rolled back.

See also the deleteRecord method of the JSFoundset.