Dynamic Data Binding

The Servoy platform provides a Graphical User Interface (GUI), as well as an Application Program Interface (API) which dynamically bind to database resources. This means the Servoy Application Server will dynamically generate and issue the SQL required to Read, Insert, Update and Delete database records in response to both the actions of the user and the behest of the developer.

In the simplest example, a user navigates to a form (which is bound to a specific database table) showing customer data. When the form shows, the Servoy Application Server issues a SQL *select* statement on the customer database table, retrieves the results of the query and displays them in the form.

When a user types a value into a text field (which is bound to a specific column of the database table) and clicks out, the Servoy Application Server issues a SQL *update* command to the database to modify the *selected record*. The resulting change is also broadcast to all other connected clients.

In another example a user may click a button, initiating a script written by a developer using Servoy's data API, a high-level abstraction to perform low-level database operations. The developer need only write code to interact with the API. The Servoy Application Server will translate the instructions into the raw SQL needed to perform the action.

The fundamental unit of data binding in both the GUI and the API is the Foundset object.

In This Chapter

Client Cache

A Servoy client instance keeps track of which database records are in use. This is called the Client Cache and it optimizes performance by reducing the number of queries made to the database. Records are tracked by primary key. The first time the contents of a record are accessed, the Application Server must issue a query to the database on behalf of the client. The values for all of the columns of the record object are held in memory and therefore, subsequent access of the record will not produce anymore queries to the database. The user experience is greatly enhanced as one can browse quickly between forms and cached records.
A record may fall out of the cache gracefully to conserve memory and is automatically reloaded the next time it is accessed. This happens at the discretion of the client's caching engine, which is highly optimized. Relieved of the burden of managing memory, the developer can focus on more important things.

Data Broadcasting

What happens to the cache when the contents of a record are changed by another Servoy client session?

Fortunately, the Servoy Application Server issues a Data Broadcast Event to all clients whenever a record is inserted, updated or deleted by another Servoy Client. This notification prompts each client to automatically update its cache providing the end users a shared, real-time view of the data.

In a simple example, two remote users are looking at the same *customer* record. The first user modifies the customer's name and commits the change. The second user immediately sees the change updated in his/her client session.

This functionality is provided by default for all Servoy client types. There is nothing that a developer needs to do to enable it. However, the developer may augment the default functionality by implementing the Solution's onDataBroadcast event handler and invoking specific business logic.

Updating the Client Cache

What happens when data is changed outside of any Servoy client sessions?

If a data change originates from another application, client caches may become "stale" for the affected records, meaning that the cached values are not in sync with the last values stored to the database. This is most likely to happen if an **existing** record has already been cached prior to being **updated** by another application. It may also happen if records are added or deleted. However, the duration of the problem will be shorter for inserts and deletes than for updates. This is because while foundsets may reload primary keys periodically as the user navigates the client session, the contents of a record can remain cached indefinitely.

Fortunately, Servoy's APIs provides several opportunities to programmatically update client caches. The best approach depends on the situation.

Flush All Client Caches

This approach will re-cache **all** records in **all** client caches for a specified database table. All foundsets based on the specified table will reissue their queries and all cached records will be refreshed from the database. This approach is the most comprehensive and therefore, most expensive in terms of performance. This approach is ideal to use when:

- External changes were made using the Raw SQL Plugin to update a specific database table. (This plugin bypasses the Data Binding layer and will not be reflected in the client cache.

- External changes are known to have been made on a specified table because a another application or service was invoked from Servoy client session.
- External changes may have been made to a specific table by another application, but it is not known for sure. In this situation it may be ideal to periodically update the client caches using a Headless Client , which is a server-side client session that can perform automated, scheduled operations.

For more information, see the flushAllClientsCache method in the programming reference guide.

Notify Data Change

This approach essentially sends a Data Broadcast Event to all clients. Clients are informed of changes to specific records, just as if those records were modified from within the Servoy environment. This approach is more granular than updating the entire cache for a specific table and should also yield better performance. This approach is ideal to use when:

- External changes were made using the Raw SQL Plugin to update a specific database records and the primary keys are known.
- Another application or service was invoked from Servoy client session affecting change to specific records who's primary keys are known.

This approach can also be used to allow Servoy's cache to be updated proactively from another application via a simple web service call. Any method can be exposed as a RESTful web service. Therefore, it is simple to create service that allows another application to inform Servoy of data changes. For more information see the documentation on Servoy's RESTful Web Services Plugin .

For more information, see the notifyDataChange method in the programming reference guide.

Refresh Record from Database

This approach refreshes the cache for a single record or an entire foundset in the calling client only. Therefore, unlike the previous two approaches, it does not affect the cache of all clients. This approach is ideal to use when:

- External changes may have been made which affect a record or foundset.
- It is not desirable to update the cache for other clients.

For more information see the refreshRecordsFromDatabase method in the programming reference guide.