2022.09 Whats new

Updates of components

Eclipse 2022.6

Java 17.0.4

Upgraded to angular 14.2

Rhino upgraded to 1.7.14

Httpclient updated to 5.2

- it doesn't allow urls anymore like: http://servoy.com//test those must be http://servoy.com/test (RFC3986)
- it could also result in other more stricter stuff

Api changes

"valuelist" property type is exposed at runtime (solution scripting) for most components provided by Servoy. But what it returns changed as well:

- it now returns an object that has a *name* and *dataset* property

Api was added to update an existing table filter (instead of just remove/add):

- updateTableFilterParam(String serverName, String filterName, QBSelect query)

ViewRecord has now almost all the same api as a normal table Record.

Features

A responsive container added for Css Positioned forms, so you can have a piece that is responsive.

- you can only add components in this container through the context menu (add -> components) of the container or the outline view.

Relation Editor changes, now the operators (=><) are separated from the modifier (# or ^).

Relation can now have an "in" operator, optionally with a modifier REMOVE_WHEN_NULL_MODIFIER if the IN points to an scope array variable; if that variable is null then this relation entry/item will be removed and the relation will return data without this relation item.

On a Column, Table or Server level you can configure what the sort order should be with nulls first or last and you can specify that sorting should always be done with upper/lower casing.

- This resulted in that the DBI information that we have (Server,Tables,Columns), which is normally in the column_info repository table, are now stored directly as "dbi" files in the WAR with this information. This information will not be in a .servoy file so it will only be in a WAR when you export with an active solution.

A new event is now available at the form level: "onBeforeHide".

- *Why?* It was needed because the "hide" operation of a form can be blocked either by a combination of auto-save enabled + stop editing failing or by a solution registered "onHide" handler for that form (that returns false). That denied the switch/hide in case that form was the main form or a root form in a tab of a tabpanel that is trying to change it's active tab. Problem was that - in case of deeply nested tabs (so forms nested through components like tabpanels/splitpanes and so on) it could happen that a form that is deeply nested denies hide (either due to stop editing blocking it or by return false in it's "onHide" handler) but, as it is not the root form affected by the change, the denied hide was not taken into account and the hide happened anyway. This could result in the nested form thinking it's still visible (it denied hide didn't it?!) when in fact it was hidden in the UI. That could lead to unexpected state in both solution code and internal Servoy code.
- *New way*: Things are now more predictable. In order for the solution to know correctly that a form was hidden or not, we added an "onBeforeHide" event (returns boolean) on forms and deprecated (but can still be used) the return value of "onHide" - so only the return value not the whole "onHide". "onHide" will only be called if the form actually gets hidden. Any form (even nested) can now actually deny the hide operation - if it returns false in "onBeforeHide" or the combination auto-save/stop editing denies the hide. If we would not have added "onBeforeHide", but just made nested forms "onHide" return value matter, it could still happen that a form returns true in "onHide" (so it thinks it will get hidden) but another (nested form) returns false thus blocking the whole hide operation; and the confusion would still be there.
- *How it works*: any of the forms (nested or not) that are being hidden can block the hide operation. This is a behavior change, but previous behavior was not working correctly, so it should not be a problem. The hide of a form first goes through the UI hierarchy and checks the "onBeforeHide" handlers of affected forms and auto-save/stop editing (inner most forms first), then onHide will only be called if all those allow hide - in all affected forms. This means that the execution order of onBeforeHide handlers will be reversed (first called on the leafs (most deeply nested forms) then on their parents) compared with current onHide call order. The onHide call order remains the same in case of nested forms.

Other changes

The way TiNG is build is changed, now we have a build directory per solution so when you switch there is not always a need to start a build again.

Also the detection to build or not is enhanced; for the (command line) War Export as well, if a previous run was done and the current run doesn't detect new changes (core servoy updates or package updates) it won't do a full build.

Log tables (when created by Servoy) are now by default created with a uuid pk.

Defaults are changed to be TiNG (editor/launching) but also when exporting in WAR, now TiNG is default included and NG1 is not, this can be adjusted with flags (-ng1)

Plugins are now deployed differently in a WAR, when the WAR is generated, if rmi (for SmartClient) is not enabled. In that case all the jars will end up in the generic classpath of a Web application (not anymore in WAR/plugins but WAR/WEB-INF/lib). This does mean that for plugins it's now mandatory to use the entry points: Creating Client Plugins - Servoy 2020 Documentation - Servoy Wiki

Known issues

It could be when upgrading from an older release that TiNG doesn't fully build because of old artifacts. To fix this you can delete once this dir: [your-workspace]\.metadata\.plugins\com.servoy.eclipse.ngclient.ui\target\