2021.06 Whats new

NG2 Client

More improvements to it and more components and service packages are ported over. (like google maps)

Now all the component and services have there NG2 code in there own GitHub repository (besides the AngularjJS/NG1 code) as NG2/angular library. This means that the releases from now on of such a package now contains the NG2 sources/distribution of that package.

So before you can start/use the NG2Client, you need to update the packages that you use, through the ServoyPackageManager. All component or service packages are created new releases for (that where already ported to NG2). Without a new version of a package with the NG2 code, NG2 will not run.

We added in the console view a console page: NG2 Build console, where you can see the progress of the build that needs to be done before an NG2 Client can be started. Also here you can see if there are problems creating the build, if that is the case you can always try once "Copy the NGClient2 sources" context menu of the SolutionExplorer tree and say yes once to the question to do a clean npm install.

It can be if you have components installed of packages that are not ported yet, the NG2 code will not build so the NG2Client can't be run.

Because the packages that you install now contributes a angular library now to the code of NG2, we really need to do really build it, in the previous version (2021.03) we shipped all ported components in 1 big package, but this is not the case anymore.

For source projects (so webpackages coming directly from git as webpackage projects) we still are searching for a nice way how to handle that, this is not fully supported in the first RC of 2021.06

Upgraded libs

upgraded eclipse (to 4.19) and many other java libraries.

upgraded build in tomcat to 8.5.64

upgraded shipped java to 16.0.1

upgraded chromium browser to 76, turn it on by default now on all platforms

inmem hsqldb driver updated to 2.5.1 and it by default now uses a different url: jdbc:hsqldb:mem:_sv_inmem;hsqldb.tx=mvcc
The transaction level was set to mvcc because we did encounter locks when dropping temp tables for a user.
this can be overriden with setting a servoy.propertie "server.inMemory.URL" back to "jdbc:hsqldb:mem:_sv_inmem"
see : http://www.hsqldb.org/doc/2.0/guide/sessions-chapt.html#snc_tx_mvcc for more info
All our in memory tables are specific to a single client so they are not access in a transaction by different connections at all.
So it should have minimal effect. Its just that we try to avoid locking when we are changing/create/dropping whole tables when we are shutting down clients.
But it could be that this has the reversed effect in some scenario's so then you need to place the url back.

Performance flags

we added 3 performance related properties in 2021.06 (the first was already in 2021.03):

servoy.foundset.statementBatching (default false, same behavior as in previous version of Servoy)

This tries to merge all insert statements of the same datasource into 1 batched sql statement.
This has a side effect that suddenly multiple records are inserted as if you are in a transaction (they are done in 1 commit)
And if 1 fails all will fail (be in the failed records and not saved) because we don't know which one was the problem.
In a none transaction mode until the one that failed would be inserted anyway,
in transaction mode also none would be inserted but only the one that would fail would be in the failed records.

servoy.foundset.verifyPKDatasetAgainstTableFilters (default true, same behavior as in previous version of Servoy)

If this is set to false then foundset or controller loadData(datasetOfPk) will not requery the database if there are table filters defined for that datasource.
So if you feed in a pk that shouldn't really be visible to the user that will not be filtered out anymore, so you are responsible for feeding in correct pks

servoy.foundset.optimizedNotifyChange (default true, different behavior as in a previous version of Servoy)

This should not have side effects so the default is changed to be more optimized.
But if there are problems that related foundset are not updated correctly with databroadcast (of new records) of other clients this can be set back to false.
This property optimizes the notifications of new records to all the loaded related foundsets for that datasource.
It builds up an index so we know much faster that for this inserted record we only have to notify a small set of RelatedFoundset instances instead of going over all of them to test if this new record maybe belongs to this set.

Client connections for a database datasource

Servoy added the ability to create client specific connections for a database datasource through the datasources api:

datasources.db.servername.defineClientConnection()

This returns an object that can be used to set another username/password and/or database properties. If those are set then after the create() call all access to the database of that client will go over that client connection. A client gets its own pool (with max 1 connection in the pool, more can be created but are not pooled to keep the resources to a minimum).

This does have a resource overhead because every client has its own connection so 100 clients creating a client connection will result in 100 database connections, instead of a shared pool of maybe only 10.

A sample:

```
        var connectionDef = datasources.db.example_data.defineClientConnection().username("test").password
("test").setProperty("myprop","myvalue").create();
        var foundset = datasources.db.example_data.customers.getFoundSet();
        foundset.loadAllRecords(); // this will use the above connection.
        connectionDef.destroy(); // destroys it, from now on the standard pool is used.
```

Onbeforelogin event

A Solution now has a onBeforeLogin event. This can be used to handle deeplinks before the user needs to login, it is called at the load of the solution before onOpen which waits for the user to login first.

This can be used in combination with a login form, to be able to handle already deeplinks, like auto login or authentication.

For a main solution that has a login solution its not really useful. Because then he onBeforeLogin of the login solution is checked/called not from the main.

This is an addition to that solution can be created with just a login form and not a login solution. So that there is a handle that will be executed before login, which is normally not allowed. You need to handle this with care.

The method does get an object with all the name value pairs that the deeplink has.

Other

Log4j enhancements for more keys that are usable in a logging line: [SVY-16019] Log entries for org.sablo.websocket.WebsocketEndpoint or org.sablo.BrowserConsole ont have a clientid and solutionName property set - Servoy

Boostrap tablesspanel is deprecated and replaced by a core Tabless panel (inline with other form component containers)

Improved Java 16 support for rhino

⚠ SVY-16025 - Jira project doesn't exist or you don't have permission to view it.

UserAgent

⚠ SVY-15970 - Jira project doesn't exist or you don't have permission to view it.

and keepalive

⚠ SVY-15895 - Jira project doesn't exist or you don't have permission to view it.

can now be on by the http client plugin.

TLS support for Rabbit broadcast plugin, support for just enabling tls without verification or certificates (internal networks) and support for a keystore and passphrase

⚠ SVY-16030 - Jira project doesn't exist or you don't have permission to view it.

Timezone resolve improvements for NGClient/NGClient 2, now the browser new Intl api is used to get the actual timezone its it, if that is give its not tried to resolve based on the 2 timezone offsets (winter and summer times)

API improvements/additions for SolutonModel for LayoutContainers (give no position, add as last) and components (no name, generate an auto one)

controller.focusFirstField() improvements, it now returns a boolean if it could make a call (there is a component where it could call focus on), also readonly components are skipped onDataChange event will get more information about the context of the property that is changed see Property Types - Servoy 2020 Documentation - Servoy Wiki (dataprovider section): SEvent.data will contain an object: {dataprovider : dataProviderID, scope : dataproviderScope, scopeid : variable_scope_or_datasource}.