

NGClient deployment

NGClient is one of the ways to deploy Solutions to end users.

NGClient is a browser-based application of which the UI is rendered in the browser using pure HTML, CSS and a bit of JavaScript for event handling and communication with the Servoy Application Server (HTML5, CSS3 and websockets communication is used). The business logic of the Solution that is running in the NGClient is executed on the Application Server for security reasons: the code of the business logic is not exposed in the webpage markup.

In This Chapter

- [War deployment](#)
- [Supported Browsers](#)
- [Browser Sessions](#)
- [Starting Servoy NG Clients \(giving arguments or using a deeplink\)](#)

War deployment

War deployment is the only option for an NGClient, because a lot more resources are needed then just the solution.

In the export dialog you have get the option to select the webcomponents and services that you want to include in the generated WAR file. All the components and services you use in your solution (directly in the designer or solutionmodel/scripting) should be checked and exported.

For more info about war deployment: [WAR Deployment](#)

NGClient needs a besides a browser that supports websockets also a application server that supports websockets (JSR-356 implementation)

Supported Browsers

NGClient needs HTML5, CSS3 and WebSocket support so it will run on any browser that supports these. Some supported browsers are:

- Internet Explorer 10 and beyond
- Firefox 11 and beyond
- Chrome 16 and beyond
- Safari 7 and beyond
- Opera 12 and beyond

Most of the browser supporting WebSocket also support HTML5 and CSS3. To see a full list of browser supporting the WebSocket see [WebSocket support](#).

Browser Sessions

An NGClient is bound to a Tab in the browser; the session will not be shared over multiple tabs as it happens with the WebClient. It does use the http session on the server for better security and also support for load balancing servers. But there can be multiple NGClients for 1 http session.

By default browser sessions will not timeout while the browser tab is open, no matter what the value of the the container's/tomcat's session timeout is. This is because NGClient uses a WebSocket that is a persistent connection between server and client. If you need to automatically log-out idle clients - that left their browser tab open, you can implement that based on `plugins.clientmanager` (see `JSCClientInformation.getLastAccessedTime()` - NGClient only).

Starting Servoy NG Clients (giving arguments or using a deeplink)

Servoy NGClients run in a browser and can be started by accessing a URL on the Servoy Application Server, which are plain start urls with the solution name in it (with or without arguments for the onsolution open) or direct deeplinks for calling directly a method with arguments

NG Client Start URL	What is does
{serverURL}/servoy-ngclient	Opens a page that lists all available ng-client solutions. It is available only if solution browsing is enabled; see Solution browsing .
{serverURL}/solutions/{mySolutionName}/index.html	Opens the specified solution
{serverURL}/solutions/{mySolutionName}/index.html?a={value}	Opens the specified solution. The argument value is passed into the Solutions onOpen event handler.

<code>{serverURL}/solutions/{mySolutionName}/index.html?a={value}&{name1}={value1}&{name2}={value2}</code>	The same as the previous scenario, but with additional custom keys for additional arguments
<code>{serverURL}/solutions/{mySolutionName}/index.html?m={myMethodName}</code>	Opens the specified solution and executed the specified method after the solution is opened
<code>{serverURL}/solutions/{mySolutionName}/index.html?m={myMethodName}&a={value}</code>	The same as the previous scenario, but specifying a single value to be passed as an argument into the onOpen event handler and deeplink method  Please note that if the solution is not loaded, the onSolutionOpen method is called before the deeplink method and only the deeplink method gets the arguments.
<code>{serverURL}/solutions/{mySolutionName}/index.html?m={myMethodName}&{name1}={value1}&{name2}={value2}</code>	The same as the previous scenario, but with custom keys for additional arguments



Parameters encoding

In order to make sure UTF-8 encoding is used for url parameters, you must look for value of `URIEncoding` setting from Tomcat connector. Beware that if `useBodyEncodingForURI` setting is used as well, it will have higher priority for used encoding. See Tomcat documentation for more details.

Starting with Servoy version 8.3, if 'index.html' is missing from the URL, it will be automatically appended and a redirect to the new URL will take place. Ex. `{serverURL}/solutions/{mySolutionName}` will redirect to `{serverURL}/solutions/{mySolutionName}/index.html` and `{serverURL}/solutions/{mySolutionName}?a={value}` will redirect to `{serverURL}/solutions/{mySolutionName}/index.html?a={value}`