Table of Contents

Programmers Reference, Client Side Plugins, Modify Forms

Modifying default behavior and presentation of forms

To modify the default behavior and presentation of forms you need to go to a particular object form that you want to modify and click on the Scripts property under the “Advanced” category in the list of properties of the form. You can define “initialization” script or “render” script or both (see Architecture of the client-side code).

The Javascript objects that are exposed to the initialization script for forms are the same as for queries:

- “config” object
- “markup” object
- “parser” object
- “widgets” object

Just like with queries, the parser object represents the form controller and the “markup” object represents the HTML markup of the form. However, there is a significant difference in the “config” object exposed by forms. Whereas for queries the “config” object represents some widget of the Kendo UI library, that is mainly responsible for the implementation of the query, there is no such widget for forms. A form is just an HTML code that consists of rows and columns of the Bootstrap grid system (http://getbootstrap.com/css/). Each row and column contains attributes of the form – depending on the type of the attribute they are either implemented as plain HTML or they may also include a configuration for a Kendo UI widget.

This HTML is also wrapped in a “panel” that includes HTML of toolbars around the form (if they are defined) and also the implementation of the default or custom panel header. Note that the HTML that includes toolbars and panel header is also included as part of the form markup.

So the “config” object for forms represents the configuration of a special Aware IM object called “panel”. You can find the code of this object in `AwareIM/Tomcat/webapps/AwareIM/aware_kendo/panel.js` file. At the beginning of the file there is a description of all configturation parameters supported by this object. You can change these parameters by your script..

For example, the following script will turn off the display of the header for the form, no matter what is specified in the ***AwareIM*** properties of the form:

```
config.preventHeader = true;
```

However, you are unlikely to need it because most of these parameters can be customized in ***AwareIM*** without having to use a script.

You may, however, want to modify the markup of the form – for example, you may want to modify the generated layout or styling of some attributes. You can use it through the “markup” object (or in the “render” script after the form has been drawn). The description of the HTML markup is beyond the scope of this document. If you want to study it you can just generate a form and use the browser

inspector to display the HTML of the form. You can then use your scripts to modify this markup.

You can, however, modify the widgets used by the form using the “widgets” object – you will probably modify widgets that represent attributes in their own Advanced scripts, but you can also modify other widgets used by a form, such as toolbars, for example – this has been already explained in the previous section.

The “parser” object allows you to access certain properties of the system that you may need (especially if you are modifying the markup of the form). The type of this object is `AwareApp_FormParser`. You can look up full methods of this object in the file `AwareIM/Tomcat/webapps/AwareIM/aware_kendo/parsers/formParser.js`

Some useful methods of the parser object that you can use here are:

- `getField (attributeName, sectionName)` – get the field of the form for the specified attribute and form section
- `getReferenceParser (refAttrName)` – get the “parser” of the reference attribute of the form, responsible for displaying a table, calendar etc – see Architecture of the client-side code

The second script (“render” script) that you can define in the Scripts dialog allows you to modify the form after it has been drawn. Here you can only use jQuery methods to modify the resulting document markup. The only object available for you here is the “parser” object representing the form controller.