GlideAjax

The GlideAjax class allows the execution of server-side code from the client. GlideAjax calls pass parameters to the script includes, and, using naming conventions, allows the use of these parameters.

Using GlideAjax:

1.Initialize GlideAjax with the name of the script include that you want to use.

2.When creating the script include, you must set the name field to be exactly the same as the class name.

3.When creating the script include, you must select the Client callable check box.

4.Specify the parameter sysparm_name. GlideAjax uses sysparm_name to find which function to use.

5.Any extra parameters may be passed in, all of which must begin with sysparm_. Avoid using predefined parameter names:

- **sysparm_name**
- **sysparm_function**
- **sysparm_value**
- **sysparm_type**

6.Code is then executed with the getXML() or getXMLWait() functions

Example:

Note: Client callable must be checked in the Script Include for this to work

Yellow:

This is the name of the class you create. This is usually the same as the name of the Script Include.

Magenta:

This is the name of the function to use in the script include. You can have a single script include with multiple functions that accept and return different parameters. For example, you could create a single Script Include for getting data related to users and keep adding functions to it as needed.

Green:

This is a parameter that is passed through the URL of the AJAX call. You can add more than one parameter. usually this is information you will use to make a GlideRecord call in the Script Include.

Red:
This is the function that asynchronously waits for a response. Any code that you need to wait for a response needs to go in the function referred to in the getXMLAnswer(). Code that doesn't need to wait goes directly after the getXMLAnswer() call inside the main Client Script function and won't wait for a response before executing.

Cyan:
These are the pieces of data you need from the Server. They are added to an object in the Script Include and passed back to the Client Script. You can do anything with these when they are returned. In this example they are used to set a value on the form.

Client Side (Client Script):

```
function onChange(control, oldValue, newValue, isLoading) {
    if (isLoading || newValue == '') {
        return;
    }

    var ga = new GlideAjax('asu_GetLocationData');
    ga.addParam('sysparm_name', 'getCampus');
    ga.addParam('sysparm_buildingid', g_form.getValue("u_building"));
    ga.getXMLAnswer(updateCampus);
}

function updateCampus(answer) {
    var clearvalue; // Stays Undefined
    if (answer) {
        var returneddata = JSON.parse(answer);
        g_form.setValue("campus", returneddata.sys_id, returneddata.name);
    } else {
        g_form.setValue("campus", clearvalue);
    }
}
```

Server Side (Script Include):

```
var asu_GetLocationData = Class.create();
asu_GetLocationData.prototype = Object.extendsObject(AbstractAjaxProcessor,
{
    getCampus: function () {
        var buildingid = this.getParameter('sysparm_buildingid');
        var loc = new GlideRecord('cmn_location');
        if (loc.get(buildingid)) {
            var campus = new GlideRecord('cmn_location');
            if (campus.get(loc.parent)){
                var results = {
                    "sys_id": campus.getValue("sys_id"),
                    "name": campus.getValue("name")
                };
                return JSON.stringify(results);
            }
        } else {
            return null;
        }
```

```
    }
});
```