CREATORCON
May 19 - 20, 2016 • Mandalay Bay • Las Vegas, NV

Lab Guide

19CD01 - Creating custom field inputs for ServiceNow forms

Martin Palacios

Default Login / Password:

admin / Knowledge16

itil / Knowledge16

employee / Knowledge16

This
Page
Intentionally
Left
Blank

Lab Goal

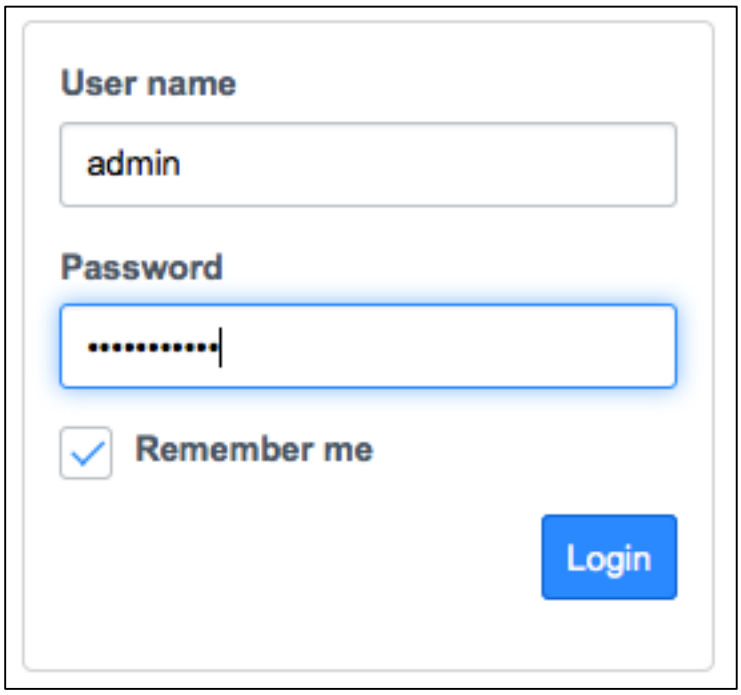
This lab explains how to create custom form elements.
The following components make up the lab:

1.1 Create Contact Type Formatter
1.2 Create Dynamic Contact Type Formatter
1.3 Create Percent Complete Knob

Lab 1.1 Create Contact Type Formatter

Navigate to the unique instance URL provided to you.

Log on with provided credentials.



Create a UI Macro

1. Navigate to **System UI > UI Macros**.

2. Click **New**

3. Enter the following information:

   Name: **incident_contact_selector**

4. Copy and paste the code below in the XML field.

```
<?xml version="1.0" encoding="UTF-8"?>
<j:jelly xmlns:j="jelly:core" xmlns:g="glide" xmlns:g2="null" xmlns:j2="null" trim="false">
  <!-- Contact Selector Container begin -->
  <ul class="contact_selector">
    <li id="contact_email" class="contact_option" onClick="selectOption(this,'email')">
      <img src="contact_selector_email.png" />
    </li>
    <li id="contact_phone" class="contact_option" onClick="selectOption(this,'phone')">
      <img src="contact_selector_phone.png" />
    </li>
    <li id="contact_self-service" class="contact_option" onClick="selectOption(this,'self-service')">
      <img src="contact_selector_self-service.png" />
    </li>
    <li id="contact_walk-in" class="contact_option" onClick="selectOption(this,'walk-in')">
      <img src="contact_selector_walk-in.png" />
    </li>
  </ul>
  <!-- Contact Selector Container End -->
  <style>.contact_selector{
        overflow:auto;
        text-align:center;
        padding:0;
        }
        .contact_option{
        width:auto;
        display:inline-block;
        cursor:pointer;
        margin:0 5px;
        background:lightblue;
        border-radius: 5px;
        opacity: .5;
        }
        .contact_option:hover,.contact_option.selected{
        opacity:1;
        }</style>
  <script>//Function called when clicking on an option
```

```
        function selectOption(obj,val){
                $j(".contact_option").removeClass("selected");
                $j(obj).addClass("selected");
                g_form.setValue('contact_type',val)
                }
        //On load read the value from contact type and select the correct contact option
                $j(document).ready(function(){
                $j("#contact_"+g_form.getValue("contact_type")).addClass("selected");
                });</script>
</j:jelly>
```

5. Click **Submit**.

Create a UI Formatter

1. Navigate to **System UI > Formatters**.
2. Click **New**.
3. Enter the following information:

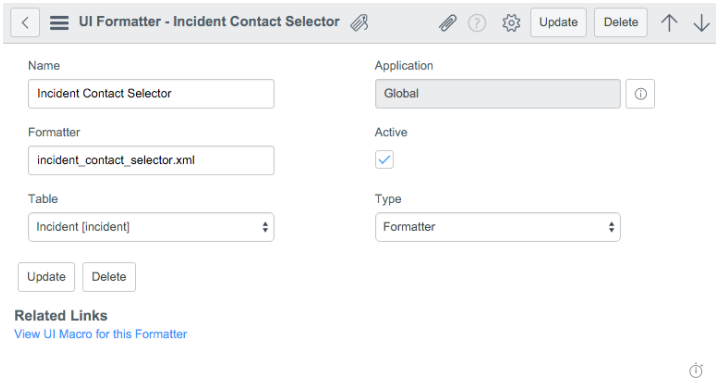
   Name: **Incident Contact Selector**

   Formatter: **incident_contact_selector.xml**

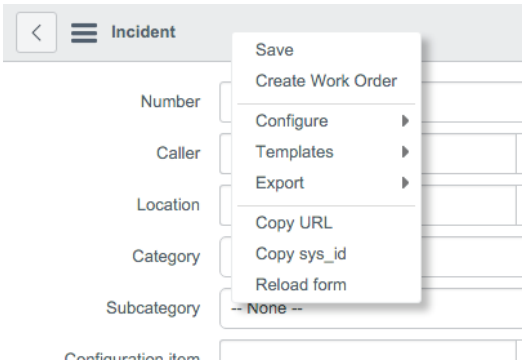
   Table: **Incident**

   Active: **True**

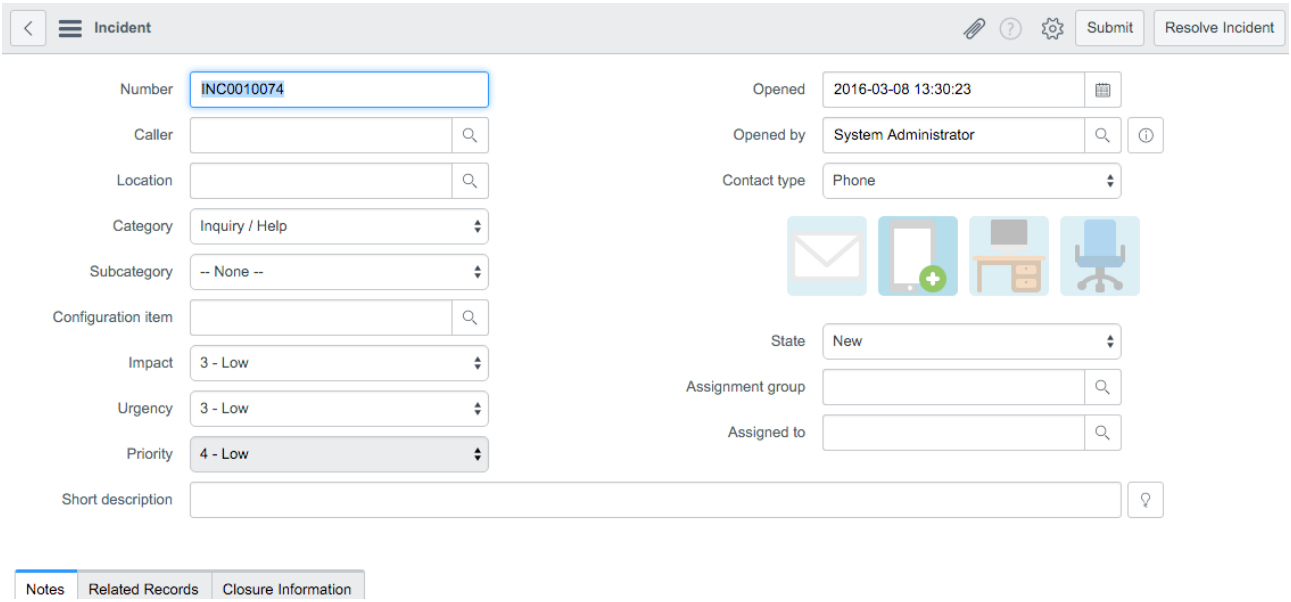
   Type: **Formatter**
4. Click **Submit**.



Add Formatter to Incident Form



1. Navigate to **Incident > Create New**
2. Right Click on header and click **Configure > Form layout**
3. Add **Incident Contact Selector** to the form.
4. Click Save



Progress Report

1. Navigate to **Lab Management > Report Lab Progress**.
2. Click **I am Done!**

Lab Goal

Lab 1.2 Create Dynamic Contact Type Formatter

This lab explains how to create a custom form element with dynamic content.

1. Navigate to **System UI > UI Macros**.
2. Click **New**.
3. Enter the following information:

   Name: **incident_contact_selector_dynamic**
4. Copy and paste the code below in the XML field.

```
<?xml version="1.0" encoding="UTF-8"?>
<j:jelly xmlns:j="jelly:core" xmlns:g="glide" xmlns:g2="null" xmlns:j2="null" trim="false">
  <ul class="contact_selector">
    <g:evaluate>var contact_options = new GlideRecord('sys_choice');
                contact_options.addQuery('table','task');
                contact_options.addQuery('element','contact_type');
                contact_options.orderBy('sequence');
                contact_options.query()</g:evaluate>
    <j:while test="${contact_options.next()}">
      <li id="contact_${contact_options.value}" class="contact_option" onClick="selectOption(this,'${contact_options.value}')">
        <img src="contact_selector_${contact_options.value}.png" />
      </li>
    </j:while>
  </ul>
  <style>
      .contact_selector{
                overflow:auto;
                text-align:center;
                padding:0;
                }
                .contact_option{
                width:auto;
                display:inline-block;
                cursor:pointer;
                margin:0 5px;
                background: lightblue;
      border-radius: 5px;
      opacity: .5;
                }
                .contact_option:hover,.contact_option.selected{
                opacity:1;
                }</style>
  <script>function selectOption(obj,val){
```

```
                $j(".contact_option").removeClass("selected");
                $j(obj).addClass("selected");
                g_form.setValue('contact_type',val)
                }
                $j(document).ready(function(){
                $j("#contact_"+g_form.getValue("contact_type")).addClass("selected");
                });</script>
</j:jelly>
```

5. Click **Submit**.

Create a UI Formatter

1. Navigate to **System UI > Formatters**.
2. Click **New**.
3. Enter the following information:

   Name: **Incident Contact Selector Dynamic**

   Formatter: **incident_contact_selector_dynamic.xml**

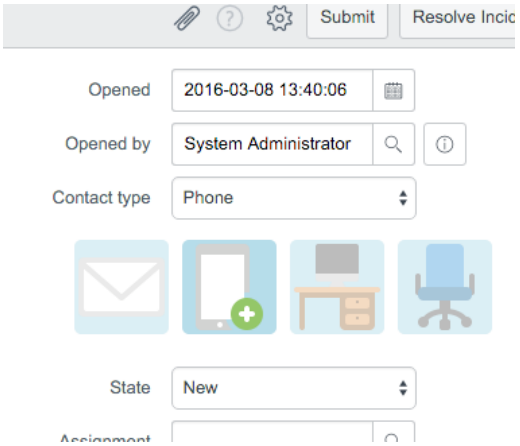
   Table: **Incident**

   Active: **True**

   Type: **Formatter**
4. Click **Submit**.

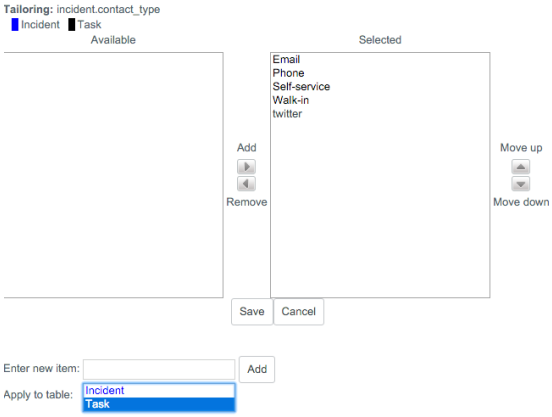
Add Formatter to Incident Form

1. Navigate to **Incident > Create New**
2. Right Click on header and click **Configure > Form layout**
3. Add **Incident Contact Selector Dynamic** to the form.
4. Remove **Incident Contact Selector** from the form.
5. Click Save



Add Choice to Contact Type

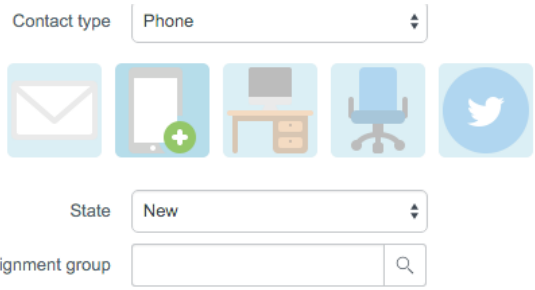
1. Navigate to Incident > Create New
2. Right Click on Contact type
3. Select Configure Choices
4. Enter twitter
5. Select **task** from the apply to table options
6. Click **Add**
7. Click **Save**



Clear Instance Cache

1. Access the following URL:

   https://[your_instance].service-now.com/cache.do
2. Click the back button of your browser
3. Navigate to Incident > Create New



Challenge

Modify the code above to take into account if choices are removed (inactivated, not deleted) from the choice list

Progress Report

1. Navigate to **Lab Management > Report Lab Progress**.
2. Click **I am Done!**

Lab Goal

This lab explains how to create a knob using an external Javascript library

Lab 1.3 Create Percent Complete Knob

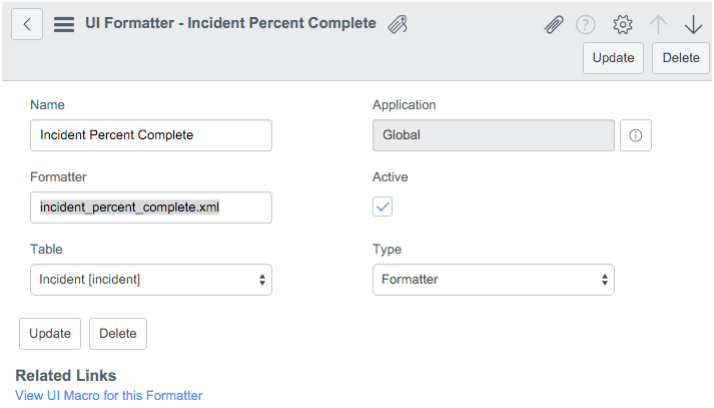
1. Navigate to **System UI > UI Macros**.
2. Click on **New**.
3. On the macro form enter the following information:

   Name: **incident_percent_complete**
4. Copy and paste the code below in the XML field.

```
<?xml version="1.0" encoding="UTF-8"?>
<j:jelly xmlns:j="jelly:core" xmlns:g="glide" xmlns:g2="null" xmlns:j2="null" trim="false">
  <g:requires name="jquery.knob.jsdbx" params="cache=$[jvar_stamp]" />
  <div class="knob_container" style="text-align:center">
    <input type="text" value="0" class="dial" />
  </div>
  <script>$j(function() {
                $j(".dial").knob({
                        fgColor:"#66CC66",
                        angleOffset:-125,
                        angleArc:250,
                        width:90,
                        height:90,
                        'release' : function (v) { g_form.setValue('u_percent_complete',v)}
                });
                $j(".dial").val(g_form.getValue('u_percent_complete')).trigger('change');
   });</script>
</j:jelly>
```

5. Click **Submit**.

Create a UI Formatter



1. Navigate to **System UI > Formatters**.
2. Click **New**.
3. Enter the following information:

   Name: **Incident Percent Selector**

   Formatter: **incident_percent_complete.xml**

   Table: **Incident**

   Active: **True**

   Type: **Formatter**
4. Click **Submit**.

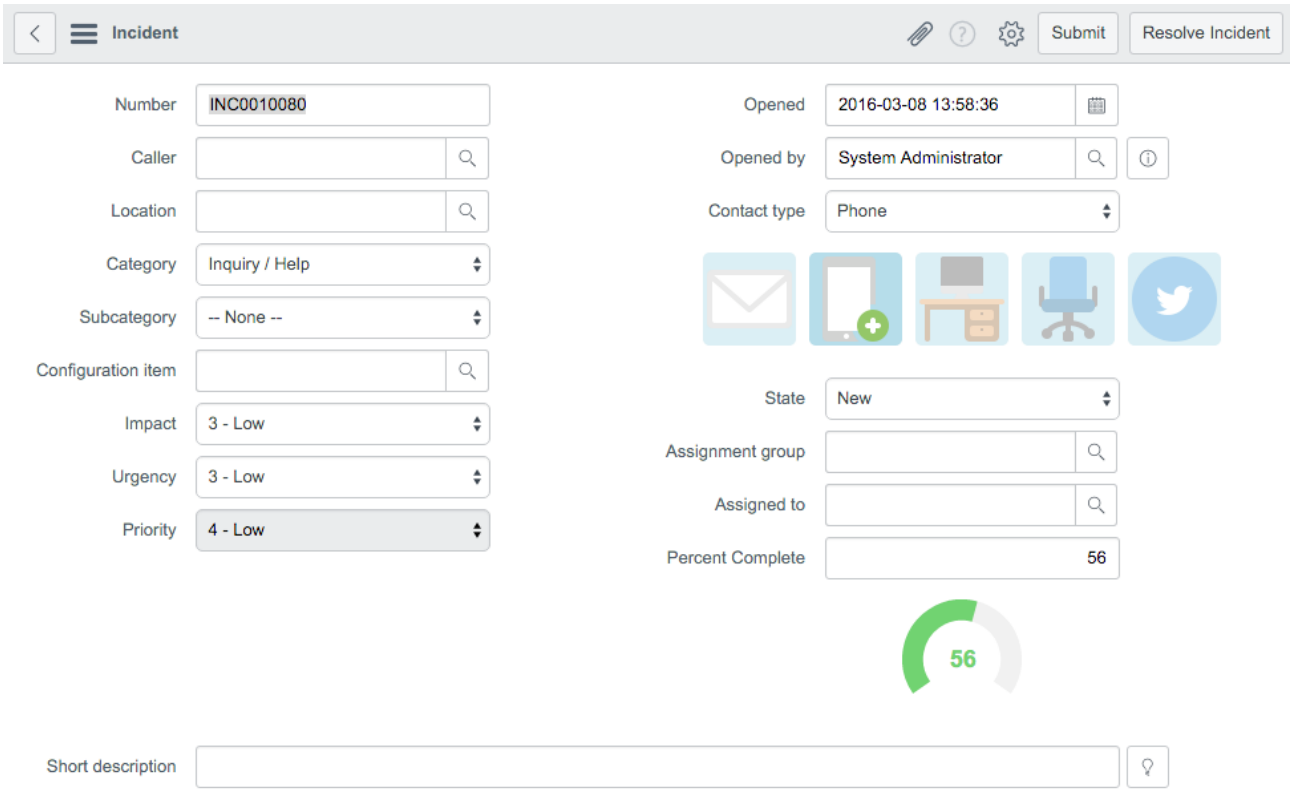
Create Percent Complete Field

1. Navigate to **Incident > Create New**
2. Right Click on header and click **Configure > Form layout**
3. Create a new field called **Percent Complete** of type **Integer**.
4. Click **Add**

   Notice it has been added to **Selected** section of the Slush Bucket.
5. Click **Save**

Add Formatter to Incident Form

1. Navigate to **Incident > Create New**
2. Right Click on header and click **Configure > Form layout**
3. Add **Incident Percent Selector** to the form.
4. Click **Save**



Progress Report

1. Navigate to **Lab Management** > **Report Lab Progress**.
2. Click **I am Done!**

Lab Goal

This lab explains how to create an assigned to selector using angular JS

Lab 1.4 Assigned to Selector using AngularJS

Create Angular Assigned To UI Script

1. Navigate to **System UI** > **UI Scripts**.
2. Click **New**
3. On the UI Script form enter the following information:

   Name: **angular_assigned_to**

4. Copy and paste the code below in the XML field.

```
// Create application definition
angular.module('assignedTo', ['sn.glideAjax']);
angular.module('assignedTo').controller('AssignedToCtrl', function($scope, GlideAjax) {
	// Stores the list of user in the assignment group
	$scope.assignedTo = [];
	// On load gets the list of users for the current assignment group
	getAssignedTo();
	// Defines get function which executes when a user is clicked on
	$scope.get = function(group) {
		getAssignedTo()
	};

	//Keeps track of the currently selected user
	$scope.selected = g_form.getValue('assigned_to');

	// Function to perform the assigned to selection
	$scope.selectAssignedTo = function(sys_id,index){
		$scope.selected = sys_id;
		g_form.setValue('assigned_to',sys_id);
	};
	/** Returns the list of users that belong to a specific group **/
	function getAssignedTo() {
		var ga = new GlideAjax('AngularGroupMembers');
		ga.addParam('sysparm_name', 'get');
		ga.addParam('group', g_form.getValue("assignment_group"));
		ga.getJSON(function(response) {
			if (!response)
				return;
			$scope.assignedTo = response;
		});
```

```
    }
});
```

5. Click **Save**

Create glideAjax connector for Angular

1. Navigate to **System UI** > **UI Scripts**.
2. Click **New**
3. On the UI Script form enter the following information:

   Name: **service.sn-glideAjax**

4. Copy and paste the code below in the XML field.

```
angular.module('sn.glideAjax', []);
angular.module('sn.glideAjax').service('GlideAjax', function($rootScope) {
    var glideAjax = window.GlideAjax;
    glideAjax.prototype.getJSON = function(callback) {
        this.getXMLAnswer(function(answer) {
            var answerJSON = JSON.parse(answer);
            callback(answerJSON);
            $rootScope.$apply();
            });
    };

    return glideAjax;
});
```

5. Click **Save**

Create Angular Framework UI Script

1. Navigate to **System UI** > **UI Scripts**.
2. Click **New**
3. On the UI Script form enter the following information:

   Name: **Angular_1_3_framework**

4. Copy and paste the code below in the XML field.

```
document.writeln('<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.js"></script>');
```

5. Click **Save**

Create AngularGroupMembers Script Include

1. Navigate to **System UI** > **Script Includes**.
2. Click **New**
3. On the Script Include form enter the following information:

   Name: **AngularGroupMembers**
   Client Callable: **true**

4. Copy and paste the code below in the XML field.

```
var AngularGroupMembers = Class.create();
AngularGroupMembers.prototype = Object.extendsObject(global.AbstractAjaxProcessor, {
	get: function() {
		var group = this.getParameter('group');
		var list = [];
		if(group !=''){
			var assignedTos = new GlideRecord('sys_user_grmember');
			assignedTos.addQuery('group',group);
			assignedTos.query();
			while (assignedTos.next()) {
				list.push({first:
assignedTos.user.first_name+'',last:assignedTos.user.last_name+'',id:assignedTos.user.sys_id+''});
			}
		}
		return new global.JSON().encodeArray(list);
	},
	type: 'AngularGroupMembers'
});
```

5. Click **Save**

Create angular_assigned_to UI Macro

1. Navigate to **System UI** > **UI Macros**.
2. Click **New**
3. On the Script Include form enter the following information:

   Name: **angular_assigned_to**

4. Copy and paste the code below in the XML field.

```
<?xml version="1.0" encoding="utf-8" ?>
<j:jelly trim="false" xmlns:j="jelly:core" xmlns:g="glide" xmlns:j2="null" xmlns:g2="null">
<script language="javascript" src="Angular_1_3_framework.jsdbx" />
        <script src="angular_assigned_to.jsdbx?v=${gs.nowDateTime()}"></script>
        <script language="javascript" src="service.sn-glideAjax.jsdbx" />
        <style>
        .assignedTo{
   display: inline-block;
   cursor: pointer;
   margin: 0 5px;
   background: lightblue;
   border-radius: 5px;
        padding:5px;
        text-align:center;
        font-weight:bold;
        opacity:.5;
        }
        .assignedTo:hover,.assignedTo.selected{
        opacity:1;
        }
        </style>
        <div ng-app="assignedTo">
  <div id="AssignedToCtrl" ng-controller="AssignedToCtrl">
          <div class="assignedTo" ng-repeat="assignedToItem in assignedTo" ng-
click="selectAssignedTo(assignedToItem.id,$index)" ng-class="{selected: assignedToItem.id == selected}" >
        <div>{{assignedToItem.first}} </div>
        <div>{{assignedToItem.last}}</div>
</div>
                    </div>
        </div>
</j:jelly>
```

5. Click **Save**

Create a UI Formatter

1. Navigate to **System UI** > **Formatters**.
2. Click **New**.
3. Enter the following information:

   Name: **Angular Assigned To**
   Formatter: **angular_assigned_to.xml**
   Table: **Incident**
   Active: **True**
   Type**: Formatter**
4. Click **Submit**.

Add Formatter to Incident Form

1. Navigate to **Incident > Create New**
2. Right Click on header and click **Configure > Form layout**
3. Add **Angular Assigned To** to the form.
4. Click **Save**

Challenge: Modify the code to also display each user's image.

Progress Report

1. Navigate to **Lab Management** > **Report Lab Progress**.
2. Click **I am Done!**

Lab Goal

As part of this lab you will create a carousel to display an incident's work notes.

The code below will display a carousel with static information. Feel free to modify it or write your own carousel.

For the displaying of the work notes you can achieve this either using jelly or AngularJS.

Lab 1.5 Challenge: Work Notes Carousel

```
<?xml version="1.0" encoding="utf-8"?>
<j:jelly trim="false" xmlns:j="jelly:core" xmlns:g="glide" xmlns:j2="null" xmlns:g2="null">
  <div class="container">
    <div id="myCarousel" class="carousel slide" data-ride="carousel">
      <!-- Indicators -->
      <ol class="carousel-indicators">
        <li data-target="#myCarousel" data-slide-to="0" class="active"></li>
        <li data-target="#myCarousel" data-slide-to="1"></li>
        <li data-target="#myCarousel" data-slide-to="2"></li>
        <li data-target="#myCarousel" data-slide-to="3"></li>
      </ol>
      <!-- Wrapper for slides -->
      <div class="carousel-inner" role="listbox">
        <div class="item active" style="height:100px;text-align:center">
    Note 1
  </div>
        <div class="item" style="height:100px;text-align:center">
    Note 2
  </div>
        <div class="item" style="height:100px;text-align:center">
    Note 3
  </div>
        <div class="item" style="height:100px;text-align:center">
    Note 4
  </div>
      </div>
      <!-- Left and right controls -->
      <a class="left carousel-control" href="#myCarousel" role="button" data-slide="prev">
        <span class="glyphicon glyphicon-chevron-left" aria-hidden="true"></span>
        <span class="sr-only">Previous</span>
      </a>
      <a class="right carousel-control" href="#myCarousel" role="button" data-slide="next">
```

```
          <span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>
          <span class="sr-only">Next</span>
        </a>
      </div>
    </div>
</j:jelly>
```

Progress Report

1. Navigate to **Lab Management** > **Report Lab Progress**.
2. Click **I am Done!**