Lab Guide

Service Portal:

Creating Custom Widgets

Will Leingang

Yago Ferrer

Default Login / Password:

admin / Knowledge16

itil / Knowledge16

employee / Knowledge16

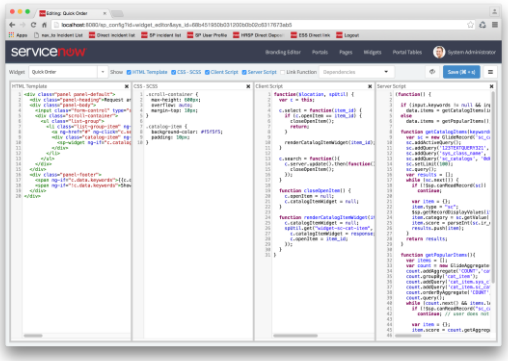
This
Page
Intentionally
Left
Blank

Lab Goal

Create a simple widget and test page

Lab 1
Building a new widget

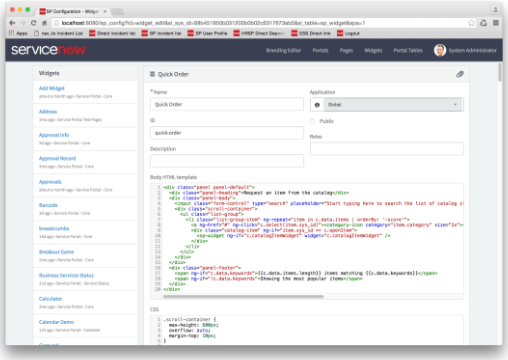
First things first. Before you can apply all of the exciting and powerful concepts you're about to learn you first need to know where to put all of that code. You will want to familiarize yourself with the widget editor. The 2 primary ways to edit a widget in service portal are with the following tools:



4 column editor

URL: /sp_config?id=widget_editor

Similar to an IDE, this editor is a full page app that lets you easily choose what parts of a widget you want to edit (Template, Client Script, Server Script…) and hides the rest.



Full widget record editor

URL: /sp_config?id=widget_edit

This page lets you edit all of the fields on the sp_widget record.

Use this editor if you need to edit angular providers, dependencies, or change the widget name.

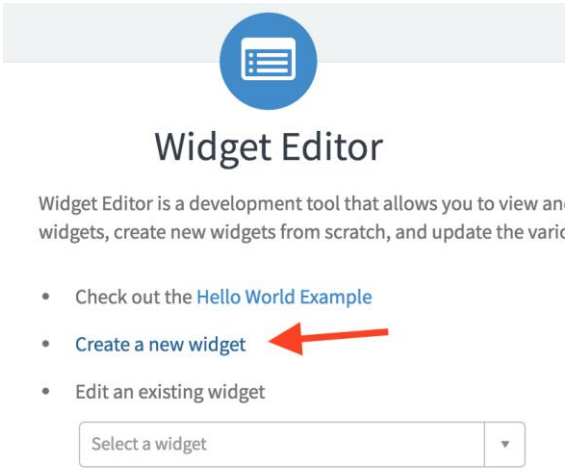
Both editors offer syntax highlighting, auto-completion and code hinting for client and server side script fields.

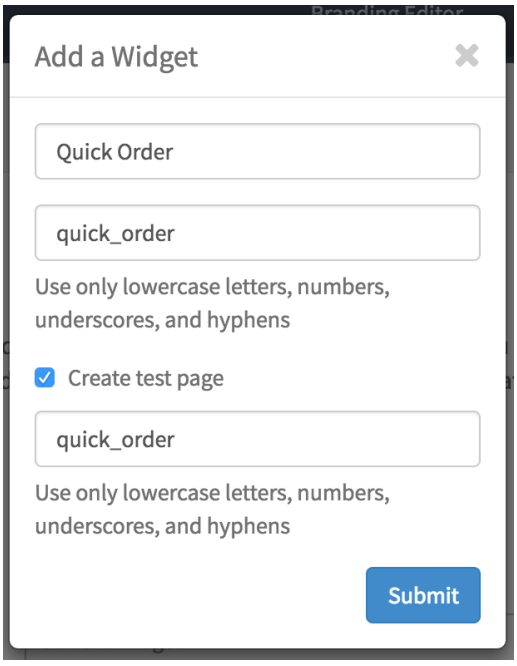
®2016 ServiceNow, Inc. All rights reserved.

Add a widget

1. Get started by going to the widget editor: /sp_config?id=widget_editor
2. Click on “Create a new widget.”
3. Add a widget with the following values:

   Widget Name: **Quick Order**
   (**The rest of the values are automatically filled in**)
   Widget ID: **quick_order**
   Create test page: **True**
   Page ID: **quick_order**





By creating a test page at the same time you automatically created the following records:

- **sp_page**
- **sp_container**
- **sp_row**
- **sp_column**
- **sp_instance**
- **sp_widget**

(Use the page editor to explore this hierarchy for any page here: /sp_config?id=page_edit)

Edit the HTML Template

Using the widget editor, put this html in the HTML Template field:

```
<div class="panel panel-primary">
 <div class="panel-heading">Request an item from the catalog</div>
 <div class="panel-body">
   My catalog results
 </div>
</div>
```

Leave all of the other widget fields alone for now.

©2016 ServiceNow, Inc. All rights reserved.

Preview

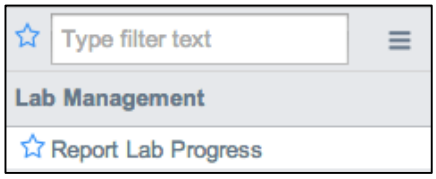
Now preview your page in a new browser tab (or window) here: /sp?id=quick_order



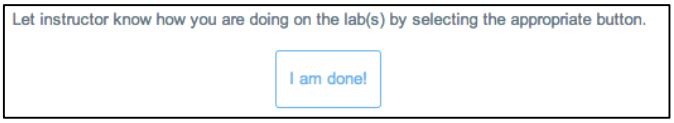
Your page should look like this, and you're done with Lab 1.

Progress Report

1. Navigate to **Lab Management > Report Lab Progress**.



Click **I am done!**



©2016 ServiceNow, Inc. All rights reserved.

Lab Goal

Lab 2
Client/Server IO

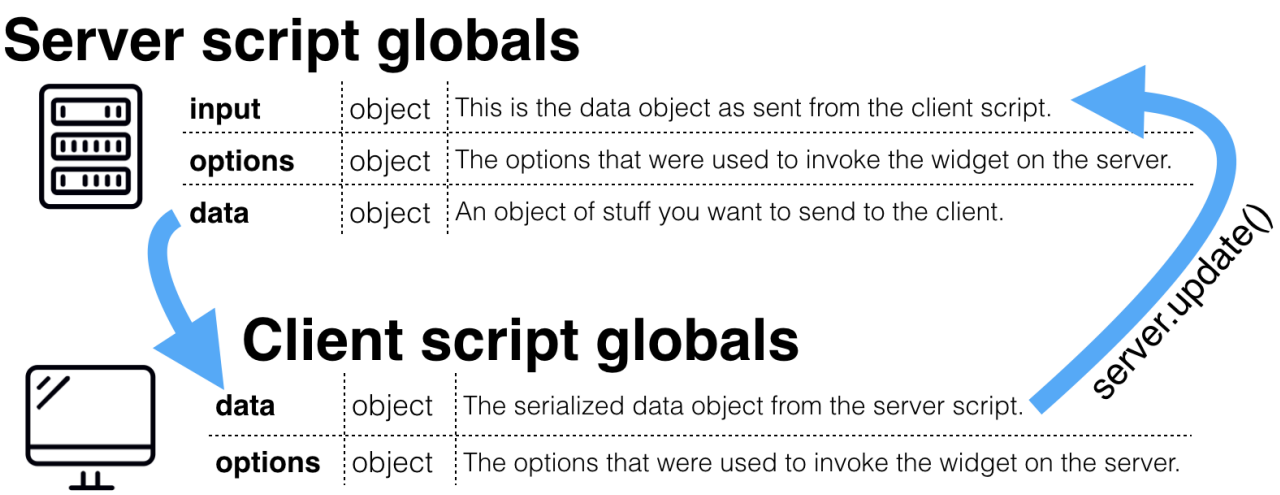
Let a user search for catalog items

In this exercise you will learn how to pass the data model between the client and server. Using the server script block you will query the sc_cat_item table and based on a keyword search return matching items.

Widgets run on both the client and server so you can get data from the database, show it to the user, and save changes with very little code!

A high level overview

The following graphic shows what variables are globally available in each scripting environment and how they are initialized.



Server Script

1. An empty **data** object is initialized.
2. Use the **input** or **options** objects to get any data that was sent from the client controller or used to initialize the widget from the server.
3. After the server script executes, the **data** object is json serialized and sent to the client controller.

©2016 ServiceNow, Inc. All rights reserved.

Client Script

1. Everything from the server's data object can be accessed via **c.data** or $**scope.data**. (You will learn more about c and $scope later)
2. If you change a value in the data model, use **server.update()** to post the entire data object to the server script as **input**.
   Note: After calling **server.update()** the client script's **data** object is automatically overwritten with the server's data object.
3. Use **c.options** or **scope.options** to see what values were used to invoke the widget. This object is readonly and making changes will have not be visible in the server script.

Server Script

Use the following server script to perform a keyword search on the sc_cat_item table. Replace the entire contents of the server script field with this code:

```
(function() {
    if (input.keywords != null && input.keywords != '')
          data.items = getCatalogItems(input.keywords);

    function getCatalogItems(keywords) {
          var sc = new GlideRecord('sc_cat_item');
          sc.addActiveQuery();
          sc.addQuery('123TEXTQUERY321', keywords);
          sc.addQuery('sys_class_name', 'NOT IN',
'sc_cat_item_wizard,sc_cat_item_content');
          sc.addQuery('sc_catalogs', '0d08b13c3330100c8b837659bba8fb4');
          sc.setLimit(100);
          sc.orderByDesc("ir_query_score");
          sc.query();
          var results = [];
          while (sc.next()) {
                if (!$sp.canReadRecord(sc))
                      continue;

                var item = {};
                $sp.getRecordDisplayValues(item, sc, 'name,price,sys_id');
                item.category = sc.getValue('category');
                results.push(item);
          }
          return results;
    }
})();
```

©2016 ServiceNow, Inc. All rights reserved.

HTML Template

This html adds a search field and displays the results of the query. Replace the entire HTML Template with this code:

```
<div class="panel panel-primary">
 <div class="panel-heading">Request an item from the catalog</div>
 <div class="panel-body">
   <input class="form-control" type="search" placeholder="Start typing here to
search the list of catalog items" ng-model="c.data.keywords" ng-
change="c.server.update()" ng-model-options="{debounce: 250}" />
   <ul class="list-group result-container">
     <li class="list-group-item" ng-repeat="item in c.data.items">
       <a href>{{item.name}}</a><span class="pull-right">{{item.price}}</span>
     </li>
   </ul>
 </div>
</div>
```

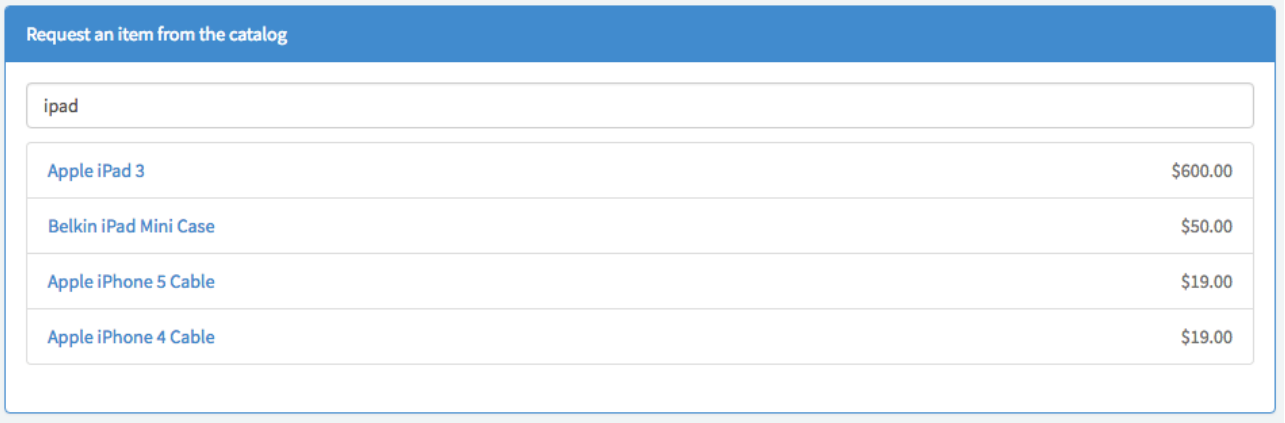
SCSS

Add this to your widget's SCSS code block:

```
.result-container {
 margin-top: 10px;
}
```

Result

As you type in the search box you will see matching catalog items. Try searching for "ipad":



©2016 ServiceNow, Inc. All rights reserved.

A deeper look

Before you continue to the next exercise, take a minute to examine these code changes and familiarize yourself with the following concepts:

Ng-model, ng-model-options, ng-change

```
<input class="form-control" type="search" placeholder="Start typing here to search the list of catalog items" ng-model="c.data.keywords" ng-change="c.server.update()" ng-model-options="{debounce: 250}" />
```

The search field is using 3 different angular directives:

ng-model	Automatically read and write value changes to the model variable **c.data.keywords**
ng-model-options	Configure how ng-model works, in this case only update the model after a user has stopped typing for 250 milliseconds
ng-change	After the model value changes, execute **c.server.update()**

c.server.update()

By default widgets are automatically created with an anonymous client controller function and the controller_as value of “c”. So c = the instance of the client script controller.

Calling c.server.update() will post the data object to the server script. After the server script is executed the data object is automatically updated with the new values from the server’s generated data object.

Ng-repeat

```
<li class="list-group-item" ng-repeat="item in c.data.items">
   <a ng-href="#">{{item.name}}</a><span class="pull-right">{{item.price}}</span>
</li>
```

If you’ve used angular before then you are already familiar with the ngRepeat directive but just in case you’re not here’s what’s going on:

1. **ng-repeat** creates a template from the element it’s applied to, including child elements
2. For each **item in c.data.items** an instance of that template is created and the expressions **{{item.name}}** and **{{item.price}}** are replaced with the values from item
3. There is no 3.

©2016 ServiceNow, Inc. All rights reserved.

©2016 ServiceNow, Inc. All rights reserved.

Lab Goal

Lab 3
The art of the empty state

Show the user a list of popular items before they search.

You might not even realize it but handling empty states has become a kind of art lately. Maybe you could get away with saying “No results” in the past but there’s no excuse for it today. It’s so big in fact that there are entire websites dedicated to showcasing empty state handling: http://emptystat.es/

This lab is all about giving the users cues and suggestions. The next steps will guide you through adding a list of popular items to the default state.



Server Script

Replace your widget’s server script block with this code. Changes are highlighted:

©2016 ServiceNow, Inc. All rights reserved.

```
(function() {
    if (input.keywords != null && input.keywords != '')
          data.items = getCatalogItems(input.keywords);
    else data.items = getPopularItems();

    function getCatalogItems(keywords) {
          var sc = new GlideRecord('sc_cat_item');
          sc.addActiveQuery();
          sc.addQuery('123TEXTQUERY321', keywords);
          sc.addQuery('sys_class_name', 'NOT IN',
'sc_cat_item_wizard,sc_cat_item_content');
          sc.addQuery('sc_catalogs', '0d08b13c3330100c8b837659bba8fb4');
          sc.setLimit(100);
          sc.orderByDesc("ir_query_score");
          sc.query();
          var results = [];
          while (sc.next()) {
                if (!$sp.canReadRecord(sc))
                      continue;

                var item = {};
                $sp.getRecordDisplayValues(item, sc, 'name,price,sys_id');
                item.category = sc.getValue('category');
                results.push(item);
          }
          return results;
    }

    function getPopularItems() {
          var items = [];
          var count = new GlideAggregate('sc_req_item');
          count.addAggregate('COUNT','cat_item');
          count.groupBy('cat_item');
          count.addQuery('cat_item.sys_class_name', 'NOT IN',
'sc_cat_item_guide,sc_cat_item_wizard,sc_cat_item_content');
          count.addQuery('cat_item.sc_catalogs', '0d08b13c3330100c8b837659bba8fb4');
          count.orderByAggregate('COUNT', 'cat_item');
          count.query();
          while (count.next() && items.length < 9) {
                if (!$sp.canReadRecord("sc_cat_item",
count.cat_item.sys_id.getDisplayValue()))
                      continue; // user does not have permission to see this item

                var item = {};
                item.name = count.cat_item.name.getDisplayValue();
                item.category = count.cat_item.category.toString();
                item.price = count.cat_item.price.getDisplayValue();
                item.sys_id = count.cat_item.sys_id.getDisplayValue();
                items.push(item);
          }
          return items;
    }
```

©2016 ServiceNow, Inc. All rights reserved.

```
})();
```

HTML Template

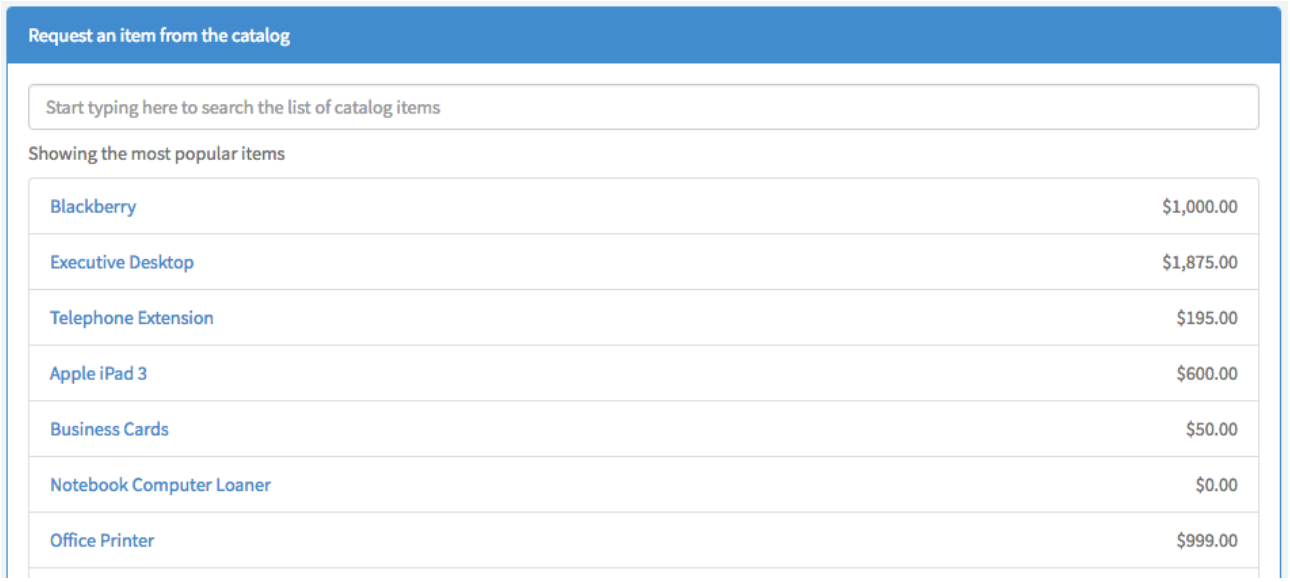
Replace the entire contents of your HTML Template with this code. The actual changes are highlighted:

```
<div class="panel panel-primary">
  <div class="panel-heading">Request an item from the catalog</div>
  <div class="panel-body">
    <input class="form-control" type="search" placeholder="Start typing here to
search the list of catalog items" ng-model="c.data.keywords" ng-
change="c.server.update()" ng-model-options="{debounce: 250}" />
    <h5 ng-if="!c.data.keywords">Showing the most popular items</h5>
    <ul class="list-group result-container">
      <li class="list-group-item" ng-repeat="item in c.data.items">
        <a href>{{item.name}}</a><span class="pull-right">{{item.price}}</span>
      </li>
    </ul>
  </div>
  <div class="panel-footer" ng-if="c.data.keywords">
    <ng-pluralize count="c.data.items.length"
                  when="{'0': 'No items found for ',
                      '1': 'One item matching ',
                      'other': 'Found {} items matching '}">
    </ng-pluralize>
    {{c.data.keywords}}
  </div>
</div>
```

Result

Now when a user sees your widget for the first time they won't think something is broken.

©2016 ServiceNow, Inc. All rights reserved.



©2016 ServiceNow, Inc. All rights reserved.

Lab Goal

Lab 4
Embed an existing widget

Clicking an item will let a user order that item without leaving the page.

In this lab you will learn how to reuse widgets by embedding one inside of another.

This lab will explain and guide you through:

- Exploring all available widgets
- Using client script to fetch a widget model
- Using <sp-widget /> to embed a widget

Explore all widgets

A great way to quickly see every Service Portal widget and explore its source code is to go to /sp_config?id=widget_edit

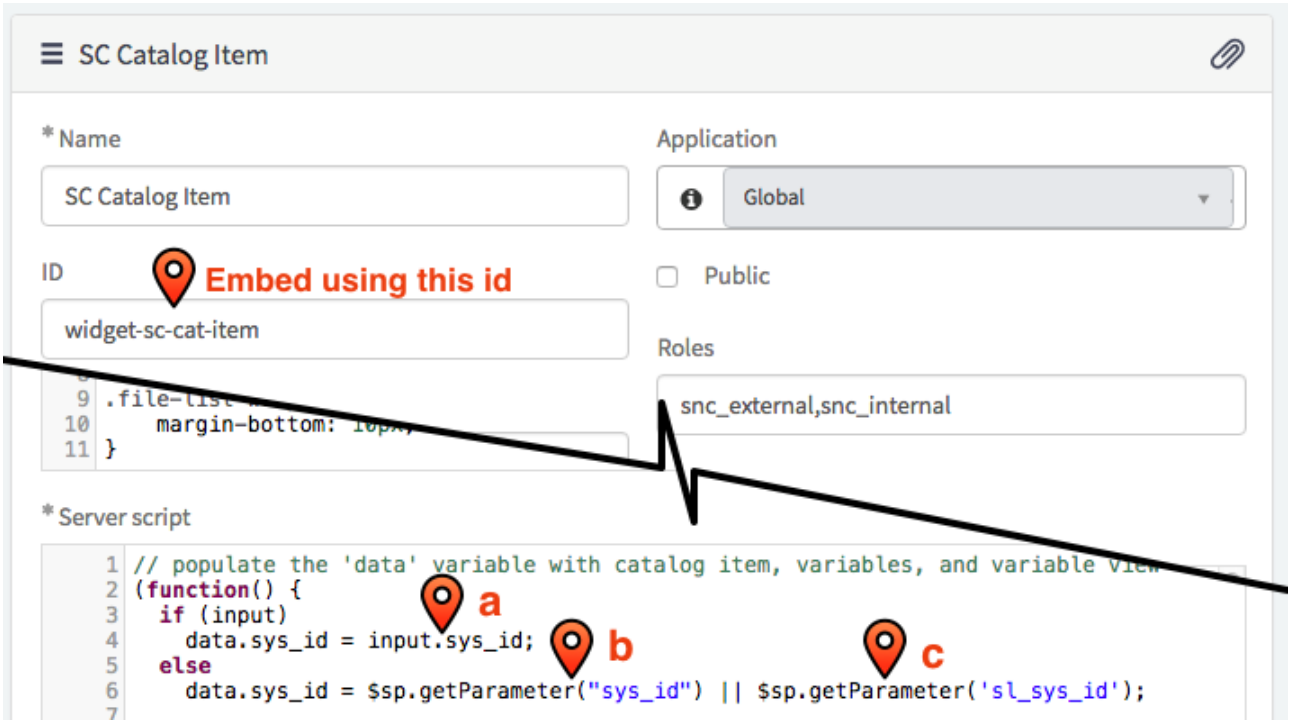
1. Go to the widget list and select “SC Catalog Item”.
   (You will probably have to scroll back up to the top of the page after clicking on the link)

2. The ID field contains the id you will use to embed the widget.

3. Scroll to the Server Script field and look at the top few lines.
   You may remember from earlier how posted **data** from the client script becomes **input** in the server script. This widget looks for a catalog item sys_id in different ways:

   a. input.sys_id — The posted data object
   b. $sp.getParameter(“sys_id”) — The page querystring param: sys_id
   c. $sp.getParameter(“sl_sys_id”) — The page querystring param: sl_sys_id

©2016 ServiceNow, Inc. All rights reserved.



So if you want to use this widget you need to pass in a catalog item sys_id… but how?

spUtil.get()

spUtil.get() is a client scripting api for getting a widget from the server. You can use it to get the "SC Catalog item" widget by ID.

This code snippet shows you just which variables to use

```
spUtil.get("widget-sc-cat-item", {sys_id: item_id}).then(function(response) {
    // this callback executes when the promise is resolved
// response is the widget model
    c.myCatItemWidget = response;
});
```

But a widget model alone is like peanut butter without bread… just an uninteresting blob. You need a way to render it!

©2016 ServiceNow, Inc. All rights reserved.

<sp-widget />

The spWidget directive takes a widget model and renders the compiled widget exactly where you put it in your HTML Template.

```
<sp-widget widget="c.myCatItemWidget"></sp-widget>
```

Okay, time to wire this stuff up inside your widget.

Double check you're editing the Quick Order widget.

Client Script

Replace the entire client script block with this code:

```
function($location, spUtil) {
  var c = this;

    c.select = function(item_id) {
          if (c.openItem == item_id) {
                c.openItem = null;
                return;
          }

          renderCatalogItemWidget(item_id);
    }

    function renderCatalogItemWidget(item_id) {
          c.catalogItemWidget = null;
          spUtil.get("widget-sc-cat-item", {sys_id:
item_id}).then(function(response){
                c.catalogItemWidget = response;
                c.openItem = item_id;
          });
    }
}
```

©2016 ServiceNow, Inc. All rights reserved.

HTML Template

Replace the entire HTML Template with this code. The specific additions from this lab are highlighted.

```
<div class="panel panel-primary">
  <div class="panel-heading">Request an item from the catalog</div>
  <div class="panel-body">
    <input class="form-control" type="search" placeholder="Start typing here to
search the list of catalog items" ng-model="c.data.keywords" ng-
change="c.server.update()" ng-model-options="{debounce: 250}" />
    <h5 ng-if="!c.data.keywords">Showing the most popular items</h5>
    <ul class="list-group result-container">
      <li class="list-group-item" ng-repeat="item in c.data.items">
        <a href ng-click="c.select(item.sys_id)">{{item.name}}</a><span
class="pull-right">{{item.price}}</span>
        <div class="catalog-item" ng-if="item.sys_id == c.openItem">
          <sp-widget ng-if="c.catalogItemWidget" widget="c.catalogItemWidget" />
        </div>
      </li>
    </ul>
  </div>
  <div class="panel-footer" ng-if="c.data.keywords">
    <ng-pluralize count="c.data.items.length"
                  when="{'0': 'No items found for ',
                      '1': 'One item matching ',
                      'other': 'Found {} items matching '}">
    </ng-pluralize>
    {{c.data.keywords}}
  </div>
</div>
```

SCSS

Replace your widget's SCSS code block with this:

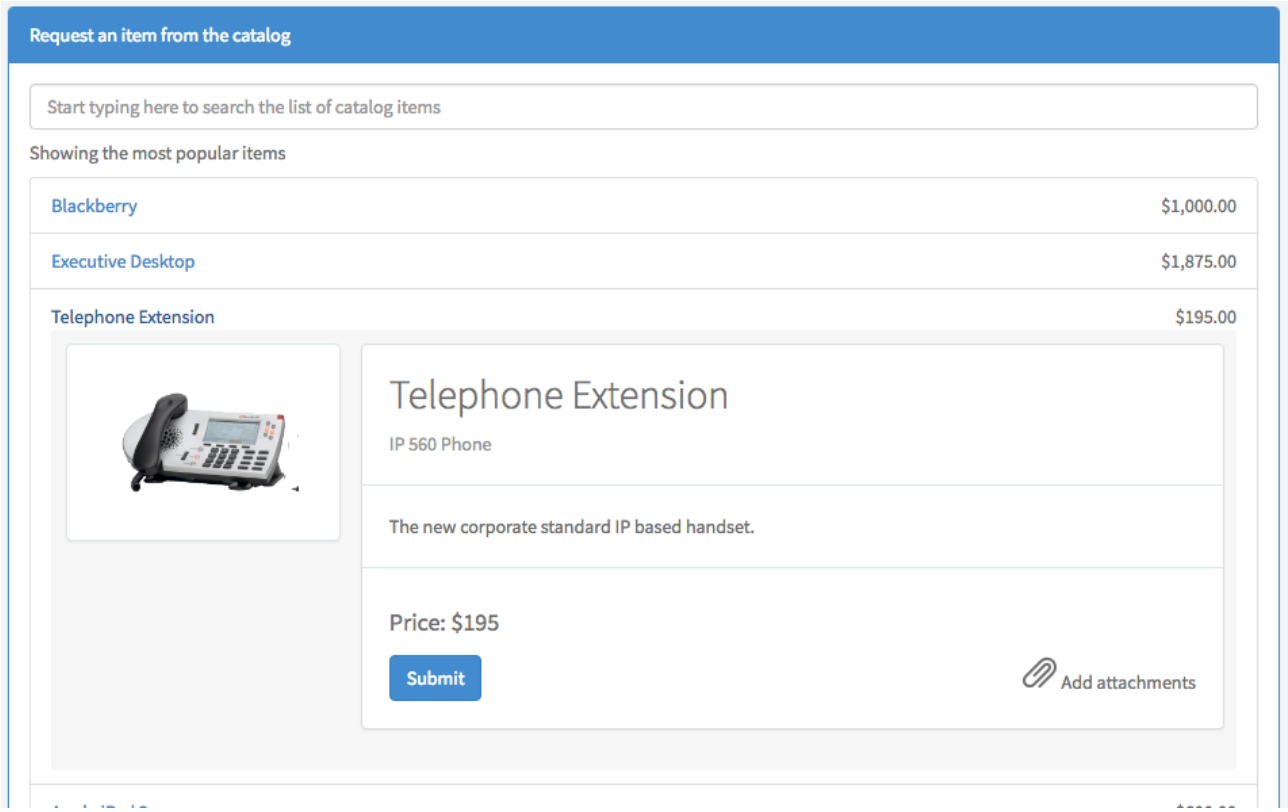
```
.result-container {
  margin-top: 10px;
}

.catalog-item {
  background-color: #f5f5f5;
  padding: 10px;
  @include border-top-radius($panel-border-radius);
  @include border-bottom-radius($panel-border-radius);
}
```

©2016 ServiceNow, Inc. All rights reserved.

Result

Now when you click on any of the catalog items, the “SC Catalog Item” widget is shown inline. A user can now order that item without even leaving the page!



A deeper look

<sp-widget /> directive

For maximum code reuse you can embed any widget inside of your widget’s HTML template using the sp-widget directive. The <sp-widget/> directive requires a complete widget model which you can construct manually or using $sp.getWidget() on the server script. Think of sp-widget as a widget factory.

©2016 ServiceNow, Inc. All rights reserved.

Lab Goal

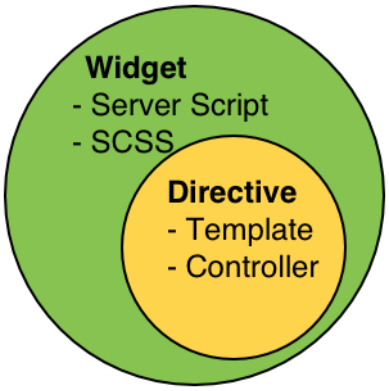
Lab 5
Reuse code with Angular Providers

Create a directive that shows a category icon next to each result

In this lab you will learn about angular providers and how to use them to share code between widgets. Angular providers let you build angular directives and services that can be automatically injected into your client script controller. The code you write in a provider differs from a typical angular directive or service because you must create them anonymously, without appending them to a specific module.

Why not just use <sp-widget />?

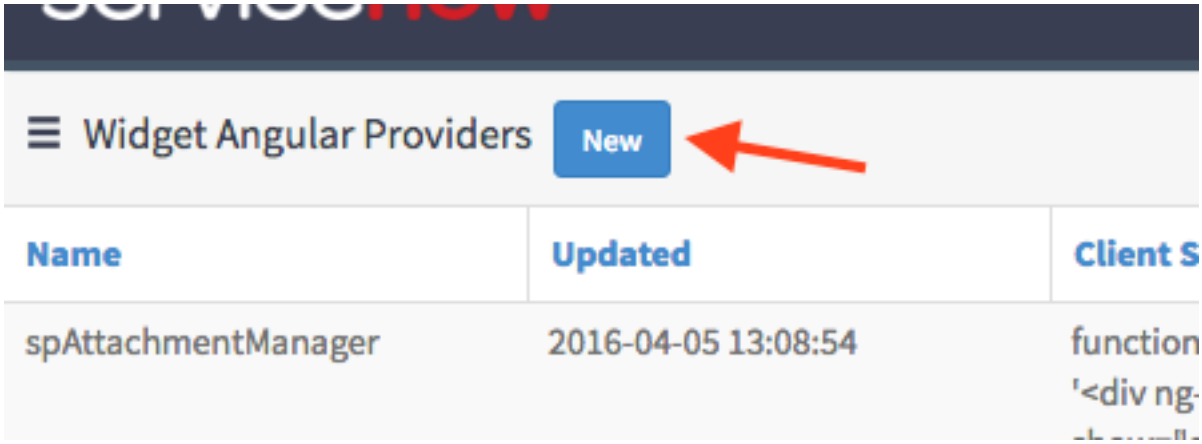
While it's true that you could use <sp-widget /> to create reusable components, it's also a fully capable application runtime and a little overweight for sharing small components.



Creating a directive

Go to the list of angular providers here:
/sp_config?id=lf&table=sp_angular_provider

Click New to create a directive.



©2016 ServiceNow, Inc. All rights reserved.

Use these values for the new record:

Type: **Directive**

Name: **categoryIcon**

Client Script:

```
function() {
  return {
    template: '<span class="fa fa-stack fa-lg"><i class="fa fa-circle fa-stack-
2x"></i><i class="fa fa-{{::icon}} fa-stack-1x fa-inverse"></i></span>',
    restrict: 'E',
    replace: true,
    scope: {
      category: '='
    },
    link: function(scope, element) {

var _iconMap = {
          "b06546f23731300054b6a3549dbe5dd8": "tablet", /* Tablets */
          "15706fc0a0a0aa7007fc21e1ab70c2f": "question", /* Can we help you? */
          "d68eb4d637b1300054b6a3549dbe5db2": "mobile-phone", /* Mobiles */
          "109cdff8c6112276003b17991a09ad65": "print", /* Office and Print */
          "5d643c6a3771300054b6a3549dbe5db0": "print", /* Printers */
          "2c0b59874f7b4200086eeed18110c71f": "plug", /* Peripherals */
          "2809952237b1300054b6a3549dbe5dd4": "desktop" /* Software */
    };

    scope.icon = _iconMap[scope.category] || "shopping-cart";
    }
  }
}
```

Click Save and you just created an angular directive!

This directive renders a font awesome icon depending on the category that's passed in. For example:

	<category-icon category="d68eb4d637b1300054b6a3549dbe5db2" />
	<category-icon category="d68eb4d637b1300054b6a3549dbe5db2" />

©2016 ServiceNow, Inc. All rights reserved.

Add the angular provider relationship

Before you can you use the new categoryIcon directive you have to associate it with your sp_widget record. For performance reasons when a widget is loaded onto a page it loads only the angular providers it actually uses.

1. Go to /sp_config?id=widget_edit
2. Open the “Quick Order” widget and scroll to the bottom of the form to the Related Lists section.
3. Click on “Angular Providers”
4. In the list that loads, click New to associate an existing Angular Provider.
5. Use these values:

   Angular Provider: **categoryIcon**
   Widget: **Quick Order**

6. Save

Use the directive

Go back to the widget editor (/sp_config?id=widget_editor) and open “Quick Order”. Now that the angular provider is associated with record you can use it in your HTML Template.

Replace your entire HTML Template with this. The changes are highlighted:

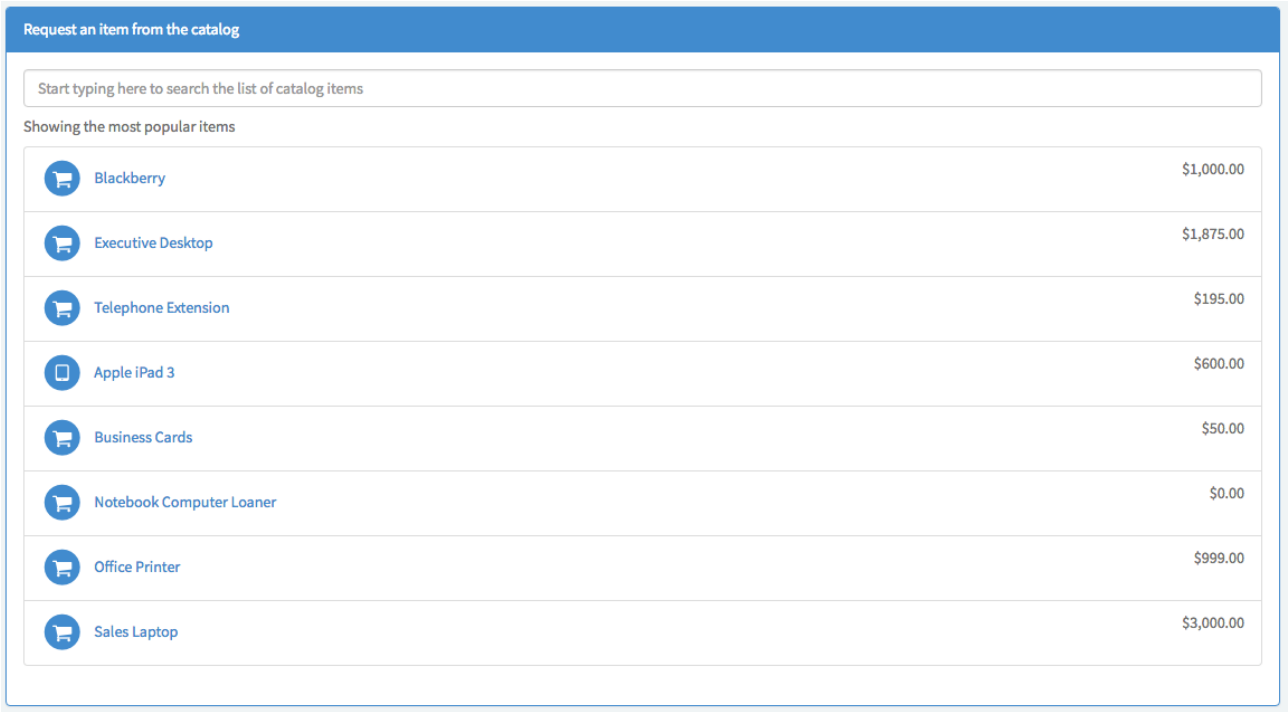
©2016 ServiceNow, Inc. All rights reserved.

```
<div class="panel panel-primary">
  <div class="panel-heading">Request an item from the catalog</div>
  <div class="panel-body">
    <input class="form-control" type="search" placeholder="Start typing here to
search the list of catalog items" ng-model="c.data.keywords" ng-
change="c.server.update()" ng-model-options="{debounce: 250}" />
    <h5 ng-if="!c.data.keywords">Showing the most popular items</h5>
    <ul class="list-group result-container">
      <li class="list-group-item" ng-repeat="item in c.data.items">
        <a href ng-click="c.select(item.sys_id)"><category-icon
category="item.category" style="margin-right: 10px"></category-
icon>{{item.name}}</a><span class="pull-right">{{item.price}}</span>
        <div class="catalog-item" ng-if="item.sys_id == c.openItem">
            <sp-widget ng-if="c.catalogItemWidget" widget="c.catalogItemWidget" />
          </div>
      </li>
    </ul>
  </div>
  <div class="panel-footer" ng-if="c.data.keywords">
    <ng-pluralize count="c.data.items.length"
                  when="{'0': 'No items found for ',
                      '1': 'One item matching ',
                      'other': 'Found {} items matching '}">
    </ng-pluralize>
    {{c.data.keywords}}
  </div>
</div>
```

The only scope attribute on the categoryIcon directive is **category**, and **item.category** contains the appropriate sys_id. See the results for yourself...

Result

©2016 ServiceNow, Inc. All rights reserved.



©2016 ServiceNow, Inc. All rights reserved.