Application Development Workshop

This

Page

Intentionally

Left

Blank

SERVICENOW APPLICATION DEVELOPMENT WORKSHOP

Table of Contents

Setup

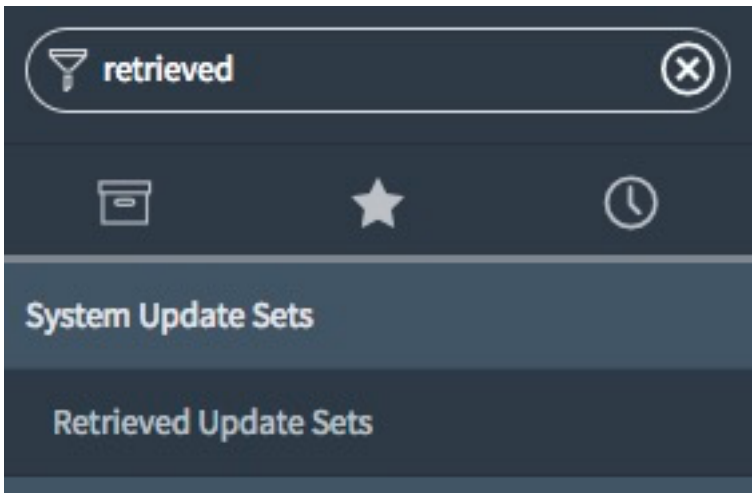
Setup

If you are working in a pre-configured lab instance, all plugins are activated and the the sample data is already loaded for you and you can skip to the next chapter.

Load sample data

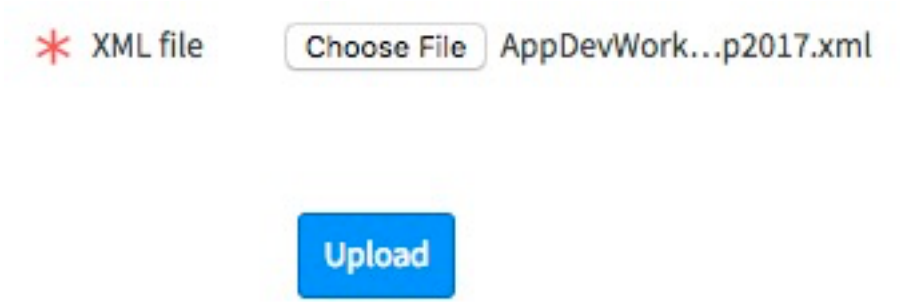
This workshop relies on a specific set of sample data. Follow these steps to load the data into your instance.

1. In the **Filter navigator** box, type “retrieved”

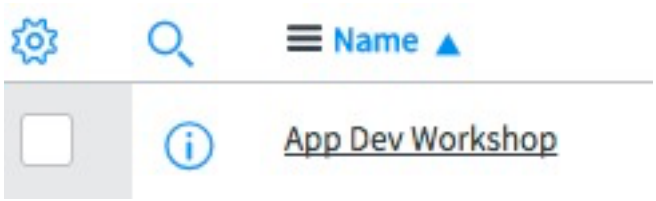


2. Click **Import Update Set from XML**

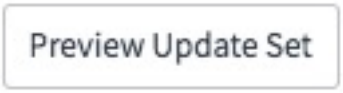
3. Select the **AppDevWorkshop2017.xml** file provided by your instructor and click **Upload**



4. Select the **App Dev Workshop** update set record



5. Click **Preview Update Set** button



6. Click **Commit Update Set** button

Required sample data is now loaded.

Start the Application

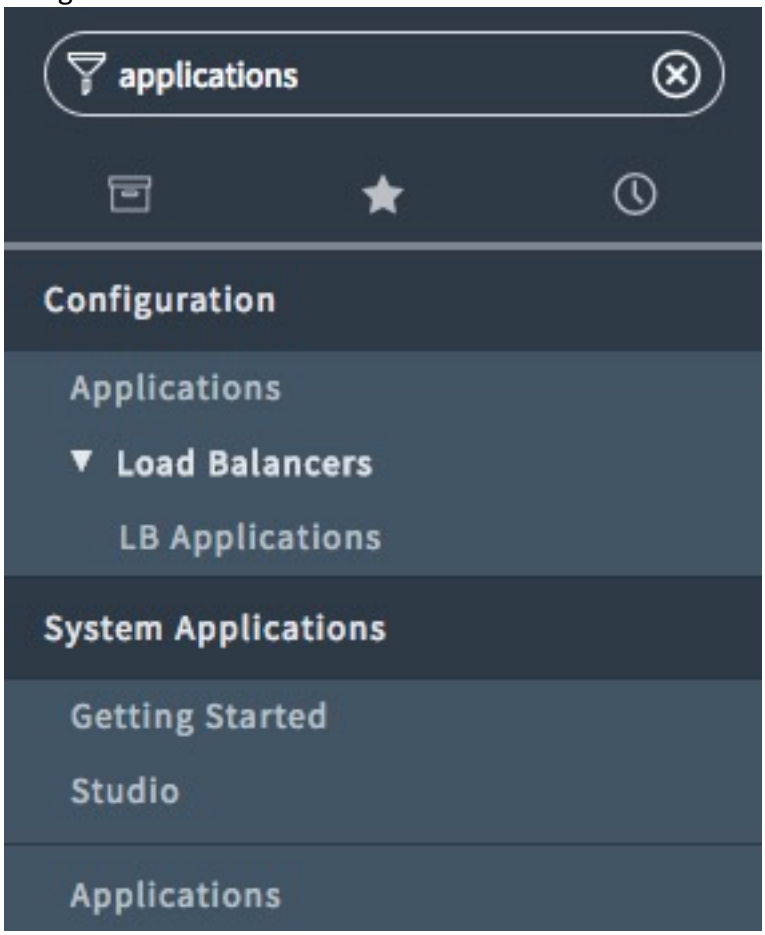
This lab explains how to create the initial application.

Lab 1
Start the Application

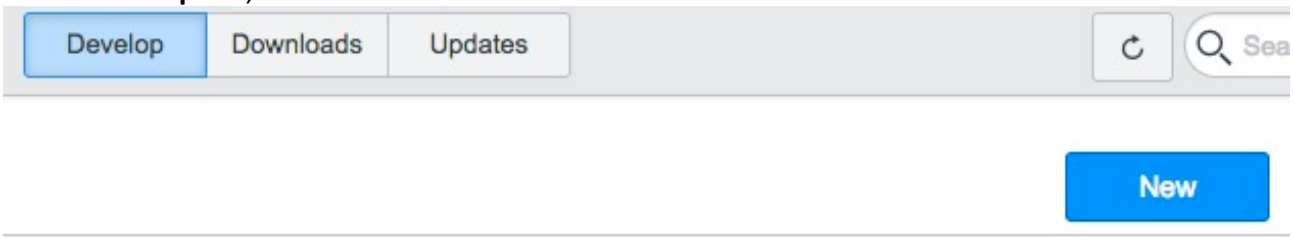
Choose How to Start the Application

In this section, you decide how to start the application and give the application a name.

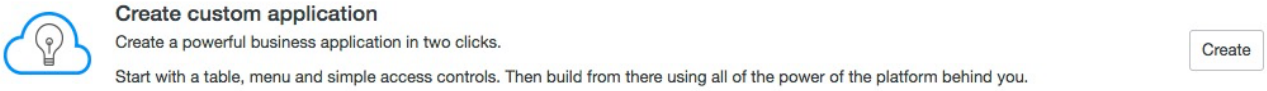
1. Navigate to **System Applications > Applications** by typing **applications** in the application navigator filter box.



2. In the **Develop** tab, click **New**.



3. In the **Get started building application** page, next to **Create custom application**, click **Create**.



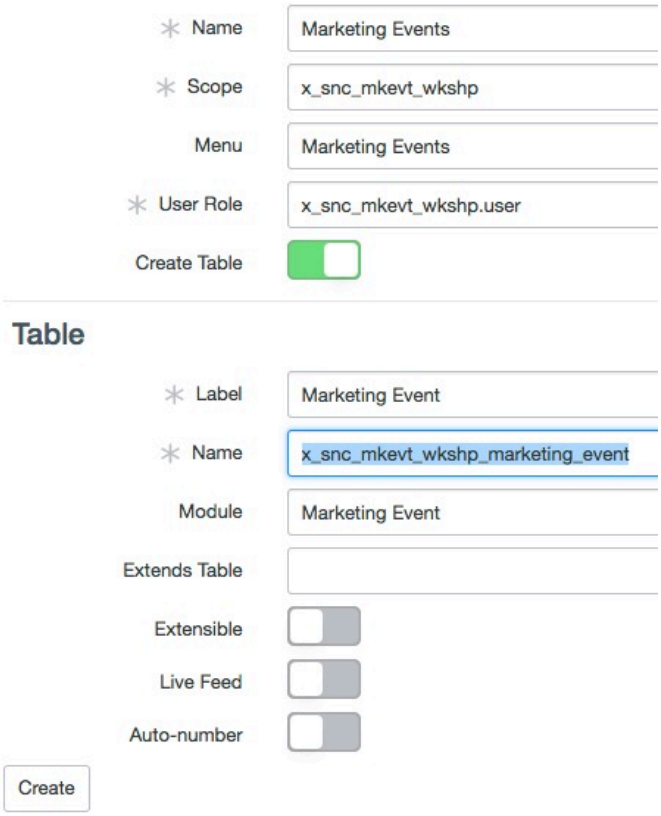
4. In the **Create Application** form, enter the following information (ServiceNow automatically fills in some fields based on the application name):

In this field…	Enter…
Name	Marketing Events
Create Table	On

5. In the **Table** section, enter the following:

In this field…	Enter…
Label	Marketing Event

Note: Your application scope will vary from the screenshots depending on if you are using a lab instance, personal developer instance or your company's dev instance.



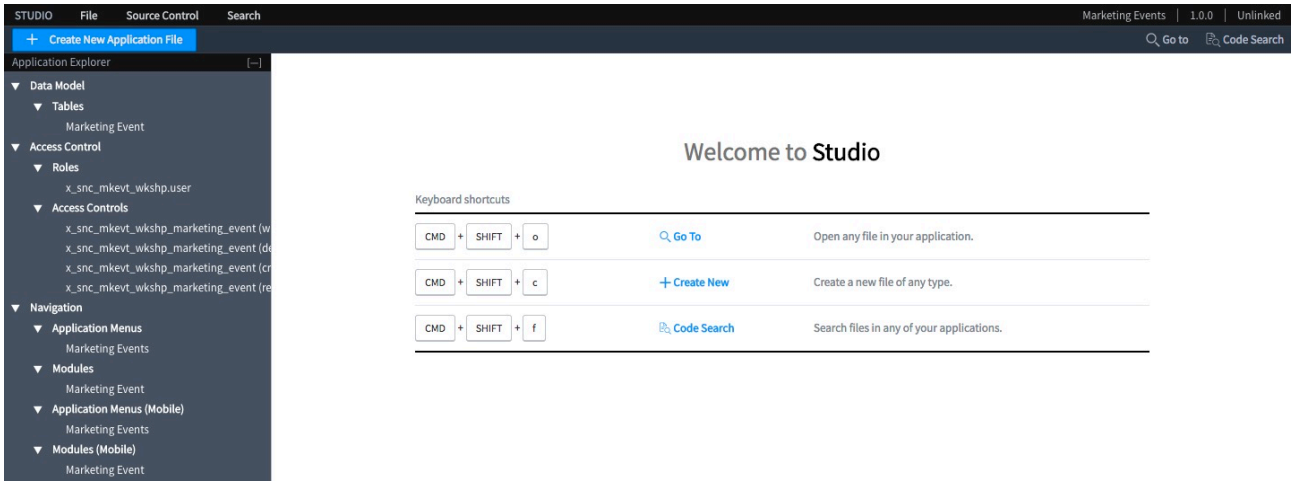
6. Click **Create**.

7. In the **Confirm Application** dialog, click **OK**.

8. Click **Edit App** to open your newly created app in Studio

Verify Results

ServiceNow has created a new application named Marketing Events.



You created a new application, but it does not do much yet. The next lab explains how to start marking it functional.

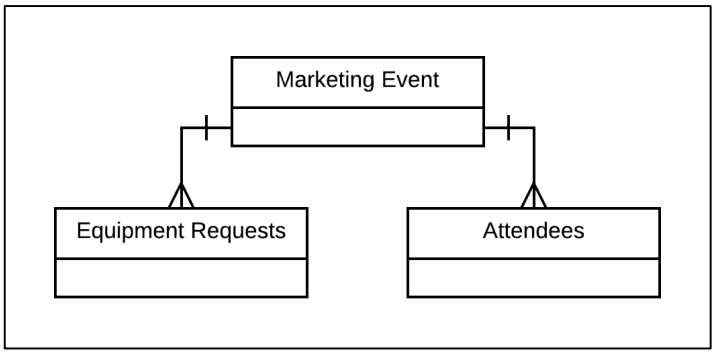
Create the Database Tables

Lab 2
Data Model

At the heart of every application is a database to store information. ServiceNow provides a relational database model for business objects. You can store the attributes of a business object and the relationships between them.

The application needs three objects:

- Marketing Event
  - Has a one-to-many relationship to Equipment Request and one-to-many relationship to Attendee.
- Equipment Request
- Attendee

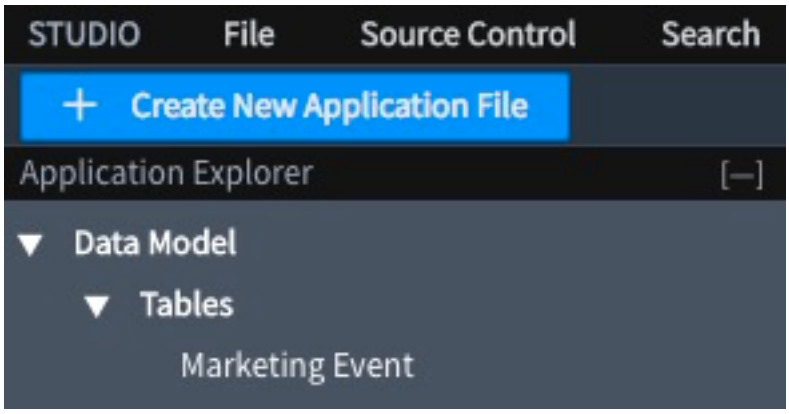


This lab explains how to build the Marketing Event and Equipment Request tables. You build the Attendee table later.

Add Fields to the Marketing Event Table

Creating the application created the Marketing Event table, but the table now only has system-generated fields. In this section, you add the fields required by the business.

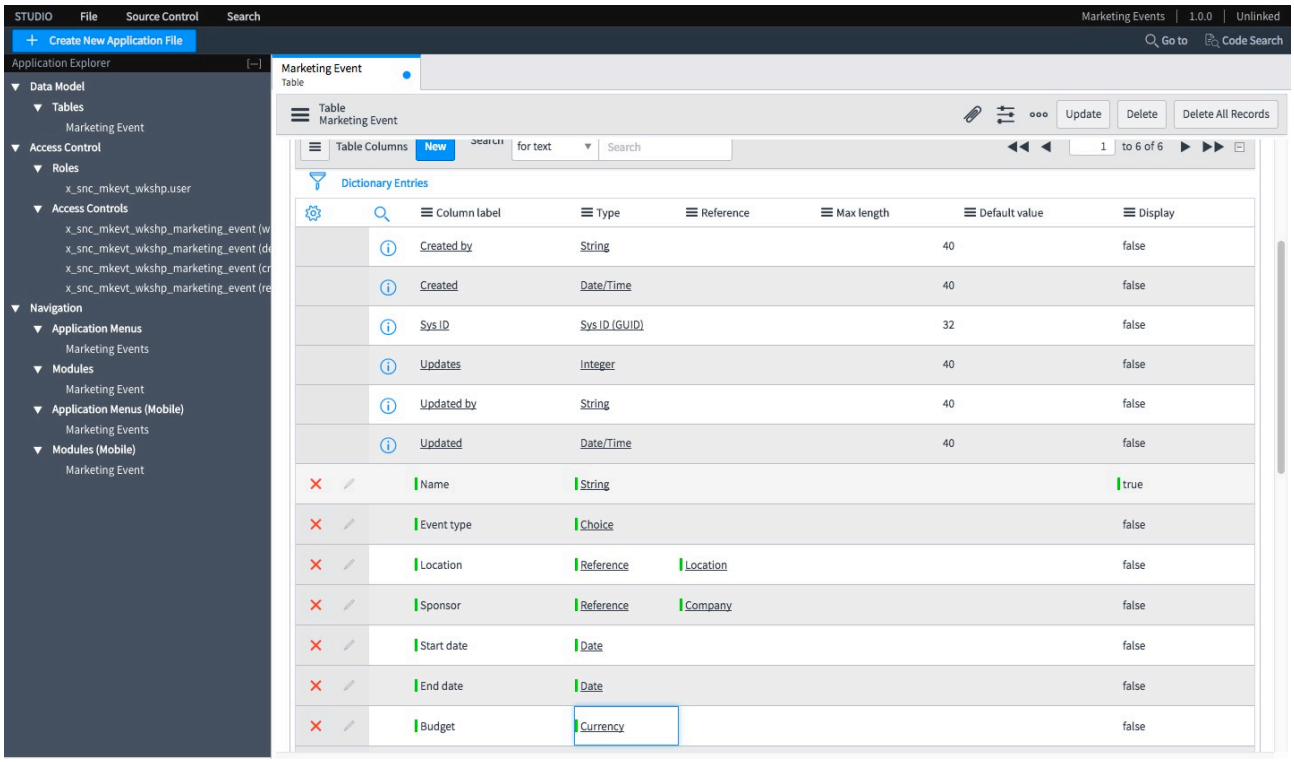
1. In the **Studio,** open the **Marketing Event** table.



2. In the **Columns** section, double-click **insert a new row**.

3. Complete the **Dictionary Entries** embedded list with the following values:

Column Label	Type	Reference	Display
Name	String		True
Event type	Choice		False
Location	Reference	Location [cmn_location]	False
Sponsor	Reference	Company [core_company]	False
Start date	Date		False
End date	Date		False
Budget	Currency		False

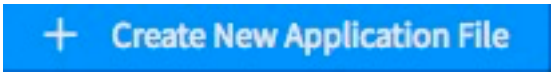


Setting **Display** to true for **Name** causes ServiceNow to display the user-friendly value for the event name rather than displaying a unique system generated ID whenever another record references a Marketing Event.

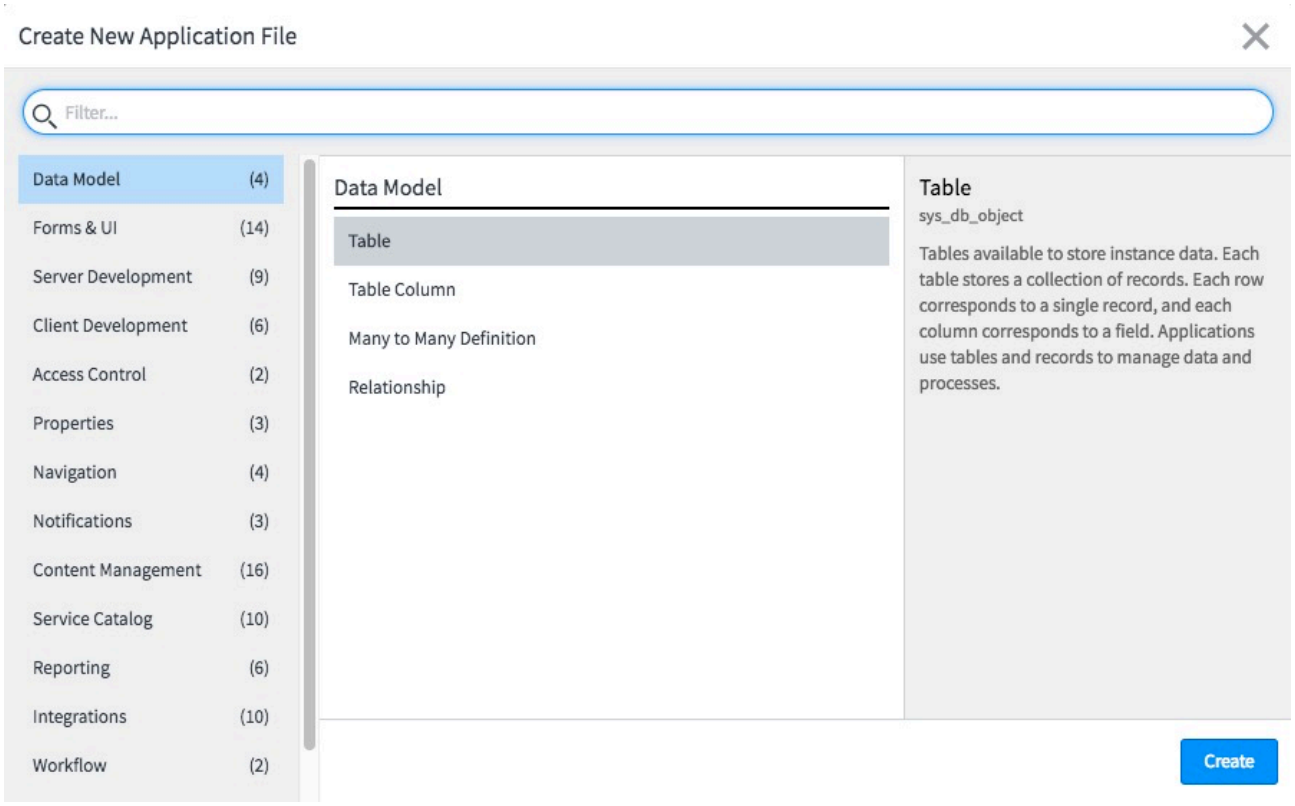
4. Click **Update**. You have created a first table with seven custom columns.

Create the Equipment Request Table

Create the Equipment Request table in a similar manner.

1. In the upper left hand corner of Studio, click 

2. Select **Data Model > Table** and click **Create**



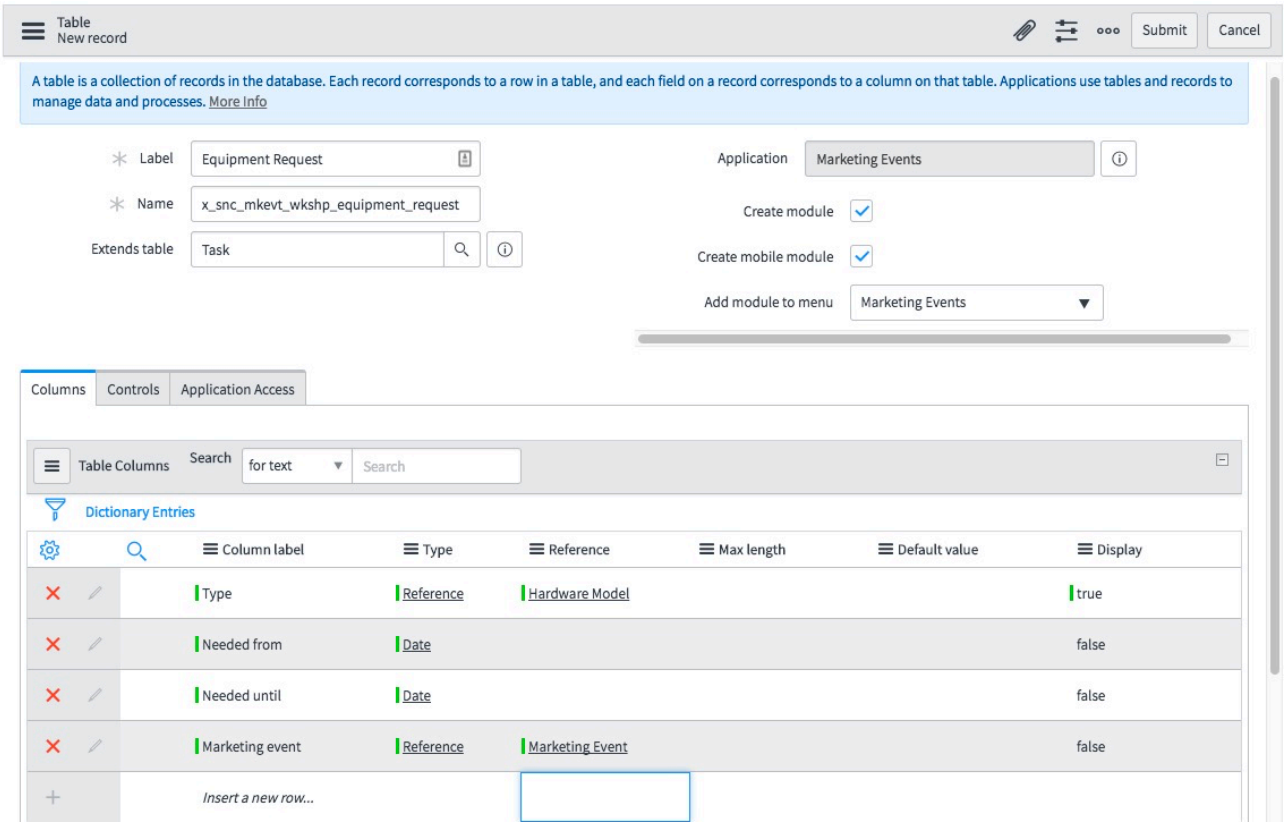
3. In the **Label** field, type **Equipment Request**. ServiceNow automatically fills in the **Name** field based on the label and application name.

4. In **Extends table** field, enter **Task** and select **Task** from the auto-complete drop-down list. Extending an existing table is a powerful facility that allows easy reuse of existing functionality. In this case, Task provides the underlying functionality for an object to participate in a workflow. Since the Equipment table extends task, it can also participate in a workflow without any additional effort. You build the workflow in a later lab.

5. Fill in the **Dictionary Entries** embedded list with the following values:

Column Label	Type	Reference	Display
Type	Reference	Hardware Model [cmdb_hardware_product_model]	True
Needed from	Date		False
Needed until	Date		False
Marketing event	Reference	Marketing Event [x_<YOUR_SCOPE_HERE>_marketing_events_event] *	False

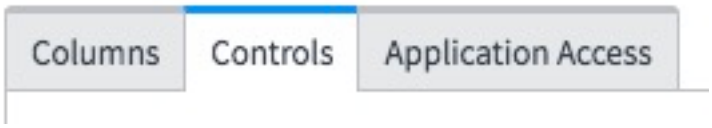
* Note: the scope prefix (x_) on tables you create on your instance will vary from what you see in this guide.



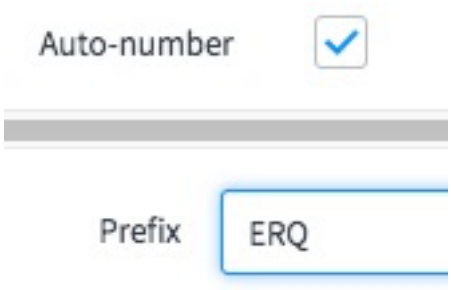
Note: The **Marketing Event** field is a **Reference** type field to the **Marketing Event** table. A reference field creates a relationship between two tables. In this case, a particular piece of

equipment is related to a particular marketing event. It allows you to display information from another table and is like a foreign key in a relational database.

5. Navigate to the **Controls** section



6. Check **Auto-number**
7. Change the **Prefix** to **ERQ**

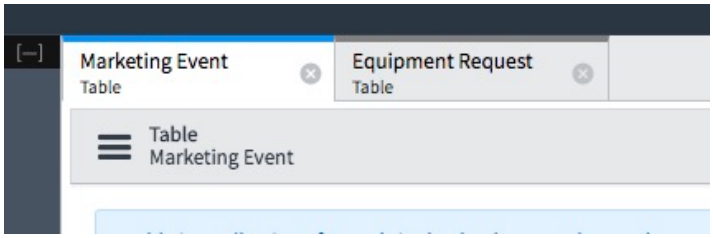


8. Click **Submit**. Now there are two tables in the application.

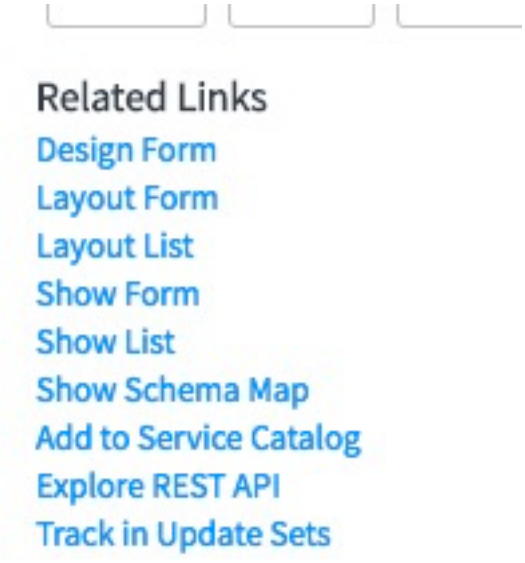
Verify Results

Get another perspective on data model using the schema map.

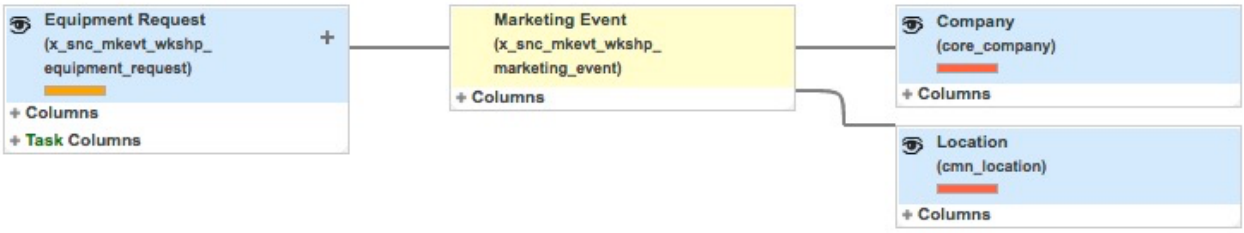
1. Click back to the **Marketing Event** table.



2. In the Related Links, click Show **Schema Map**.



This shows the schema map (Entity Relationship diagram) for the application.



If you do not see the relationship as depicted above, it means that the reference field between the two tables is not properly set. Go back to step 4 in the previous section.

Use the schema map to drill down and look at the application's tables and fields. Notice that there are a few standard system fields added such as Sys ID and Updated. For the Equipment Request table, note the fields included from extending the Task table. The application now has two tables. In the next lab, you modify the user interface so that you can enter data into these tables.

Optimize the User Interface

Lab 3 User Interface

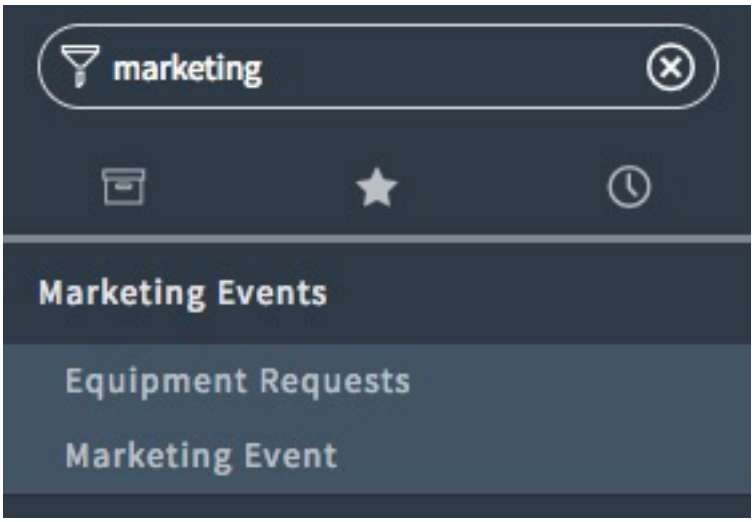
Users need to interact with the application to enter details about a marketing event or to request equipment. Fortunately, ServiceNow automatically generates most of the user interface (UI) features you need. By default, ServiceNow automatically provides a list view of all records in a table and a form to create new records. The current task is to customize the automatically generated UI.

This lab explains how to modify the user interface for users to interact with the application.

Inspect what's been generated

When you create tables, lists and forms are generated as well.

1. Switch windows from Studio back to the main instance window and reload your browser
2. Type "**marketing**" into the **Filter navigator**



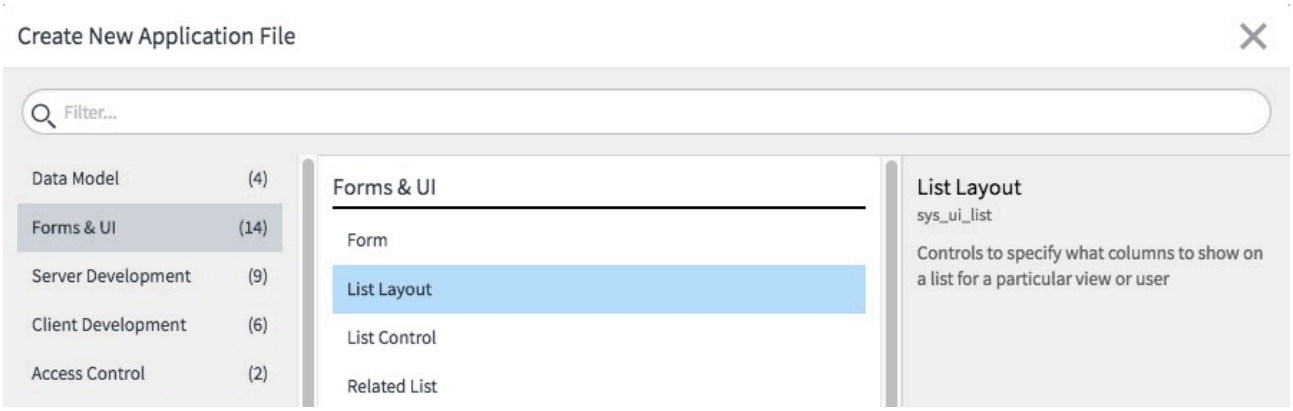
3. Explore the Marketing Event and Equipment Requests. Make note of the values and behaviors when you select values on the **Event Type** field on **Marketing Event** and **Type** on **Equipment Request**.

You'll find that while a good start, the layout and order can be improved. In the next section, you will change the layouts to make the app easier to use.

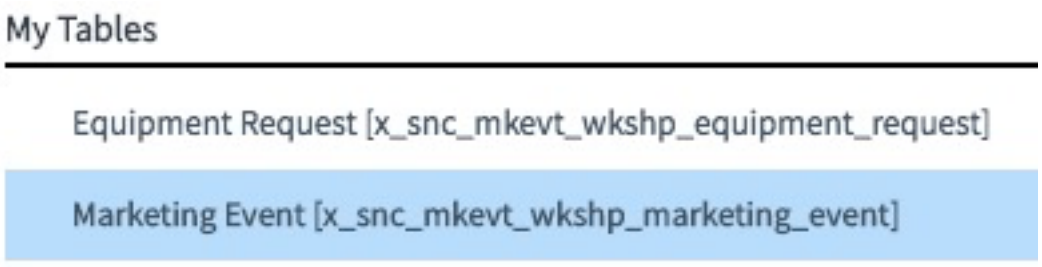
Modify the Marketing Event List Layout

In this section, you change how the list columns are presented to users looking at marketing events.

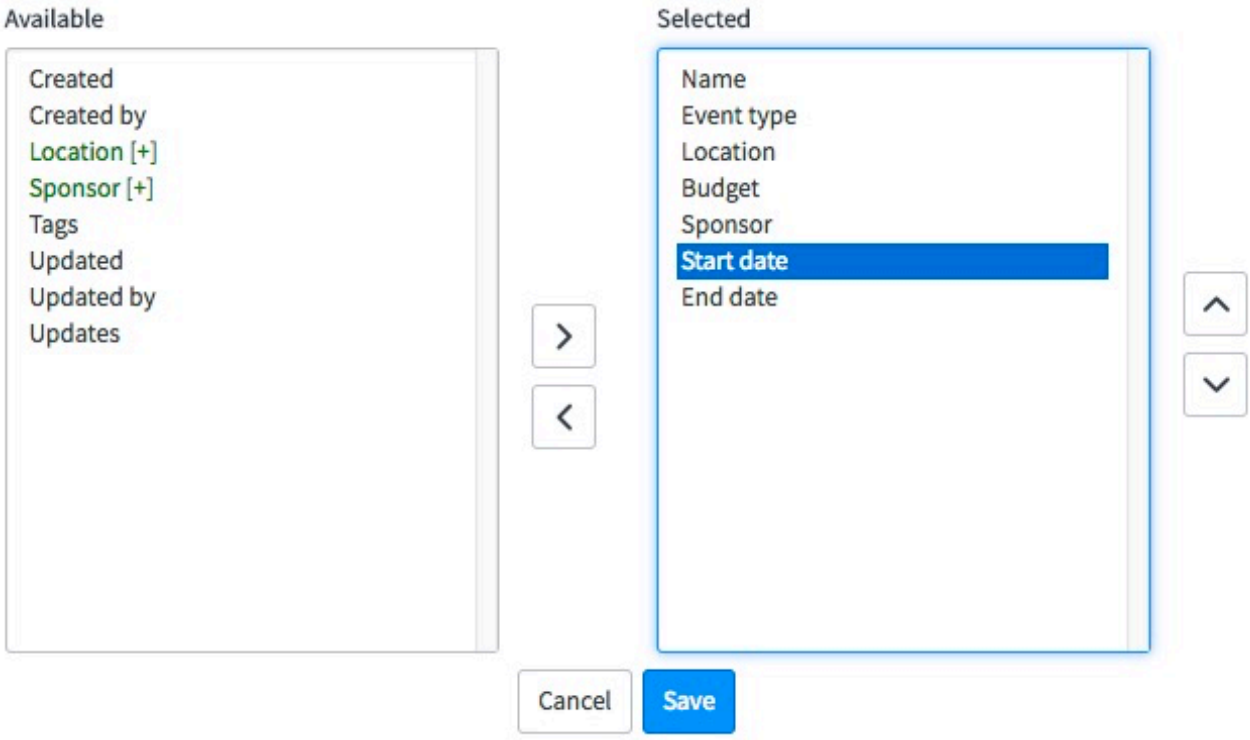
1. Switch back to the Studio browser window and reload
2. **Create New Application File** and select **Forms & UI > List Layout**



3. Select **My Tables > Marketing Event** and click **Create**



4. Arrange the column headers by selecting a column and clicking the **Up** or **Down** arrow. Arrange the column headers in the following order: **Name, Event Type, Location**, **Budget**, **Sponsor**, **Start Date**, and **End Date**.



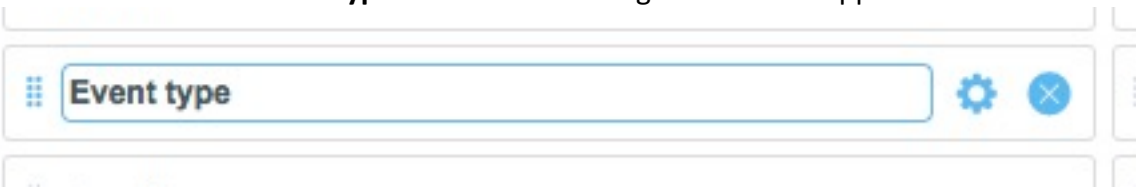
5. Click **Save**. The list uses the new order of column headings.

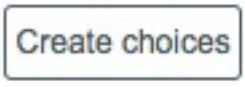
Modify the Marketing Event Form

Now that the list is formatted better, turn your attention to the form layout.

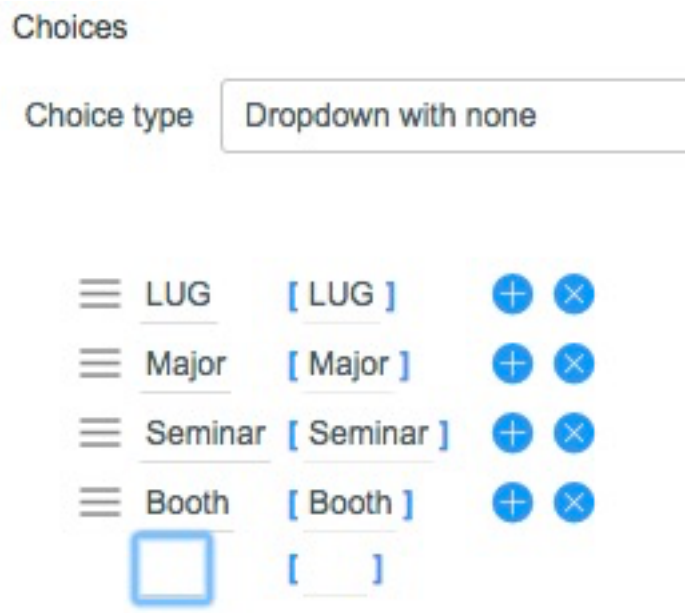
1. If you see **Marketing Event** under **Forms & UI > Forms**, skip to step 4
2. **Create New Application File** and select **Forms & UI > Form**
3. Select **My Tables > Marketing Event** and click **Create**
4. Configure the form as follows by dragging the field from **Fields** list on the left and dropping them on the **Marketing Event** form, as specified below

5. **When exploring**the **Marketing Event** form, you may have noticed how the **Event type** field is a drop-down field containing no values. Remember that you made this into a **Choice** field. Hover your mouse over the **Event type** field and click the gear icon that appears



6. Click the 

button
7. Add the following values for event types: **LUG, Major, Seminar,** and **Booth**.



8. Close the Properties window

Note: Providing users with a choice list prevents data entry errors and typos that a free-form text field allows. A fixed list also prevents users from creating multiple variations of the same value.

9. Click **Save** when done.

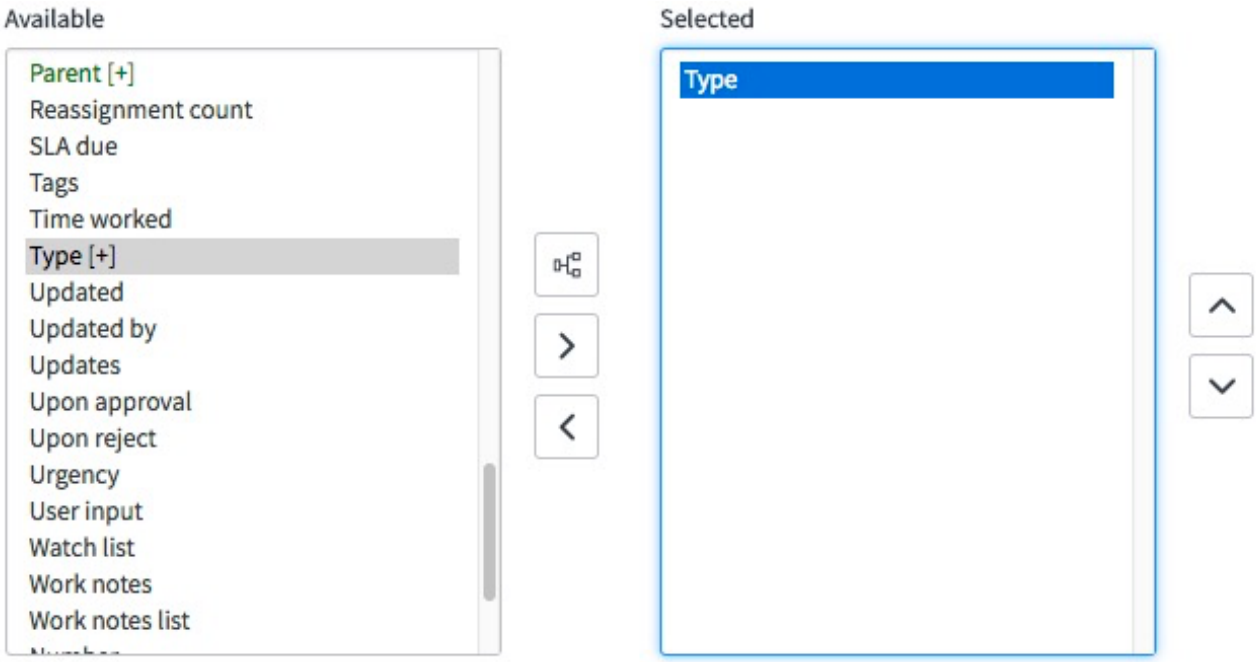
Modify the Equipment Request List Layout

Now customize the Equipment Request list and form views following a similar process.

1. **Create New Application File** and select **Forms & UI > List Layout**
2. Select **My Tables > Equipment Request** and click **Create**
3. Since the Equipment Request table extends from the Task table, you will see a lot of fields in the available fields list and some fields from the Task table in the **Selected** list.
4. Remove all the default fields in the **Selected** list by double-clicking on the field names. Alternatively, hold down the **SHIFT** key and select all these fields and then press the **<** (remove) button.

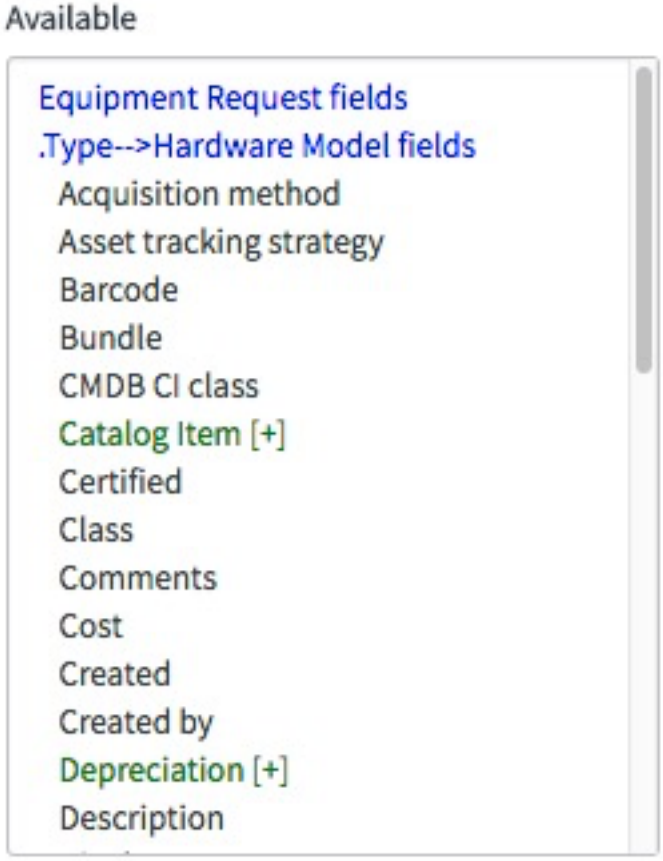
Note: You can safely remove fields from a list or form view. Removing fields with configuration does not change the actual table structure.

5. In the **Available** list, double-click on the **Type** field to add them to the Selected list.

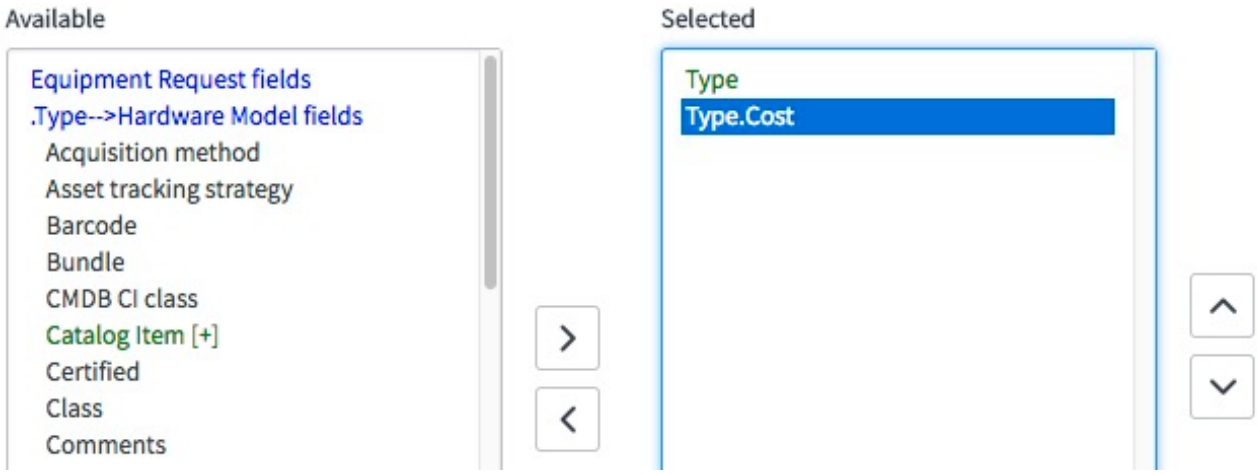


6. Display in the Equipment List a field from the related Hardware Model table. To see fields from this related table, select the **Type** field from the **Available** fields list. ServiceNow displays tables in green text with a [+] next to the table name.

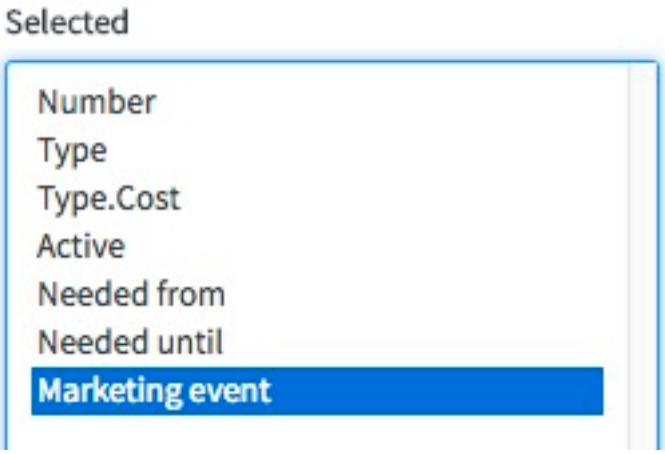
   Click on  to show a list of the fields in the related table.



7. Double-click the **Cost** field to add it to the Selected list.



8. In the **Available** list, click **Equipment Request fields** to return to the fields directly on the Equipment Request table.
9. Add and order the **Needed from**, **Needed until**, and **Marketing event**.



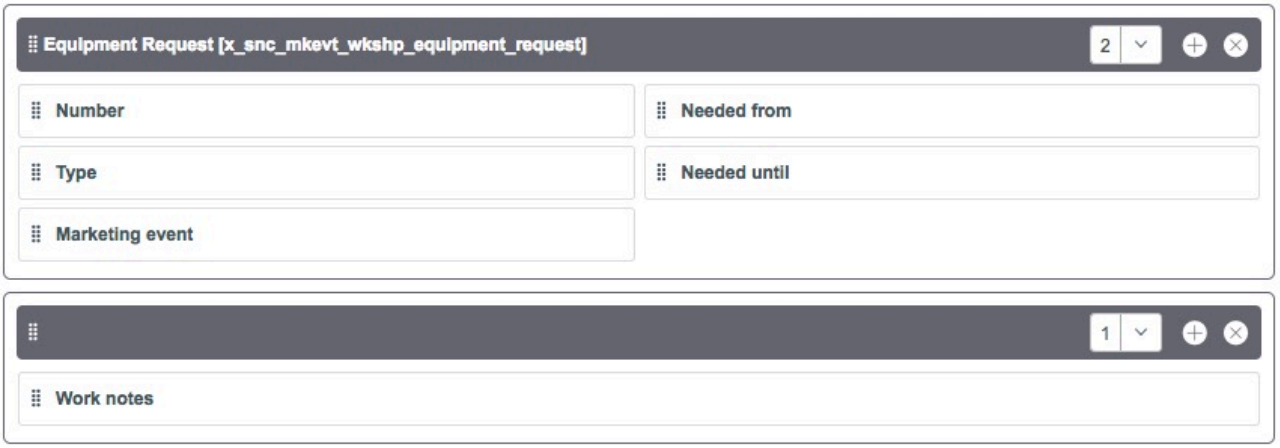
10. Click **Save**.

Modify the Equipment Request Form

Now configure the Equipment Request form.

1. If you see **Equipment Request** under **Forms & UI > Forms**, skip to step 4
2. **Create New Application File** and select **Forms & UI > Form**
3. Select **My Tables > Equipment Request** and click **Create**
4. Remove all the default fields in the form except **Number** and **Work notes** by selecting each field and then clicking the **X**.

5. Configure the form as follows:

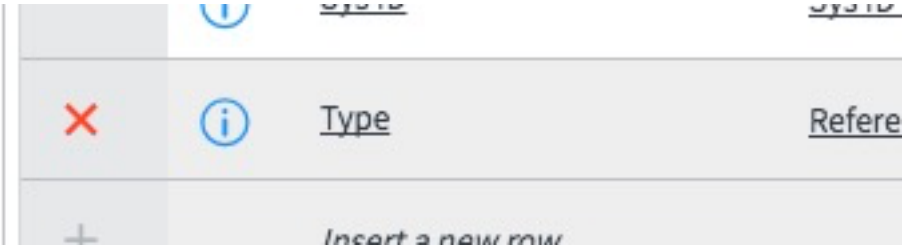


6. Click **Save**.

Create a Reference Qualifier

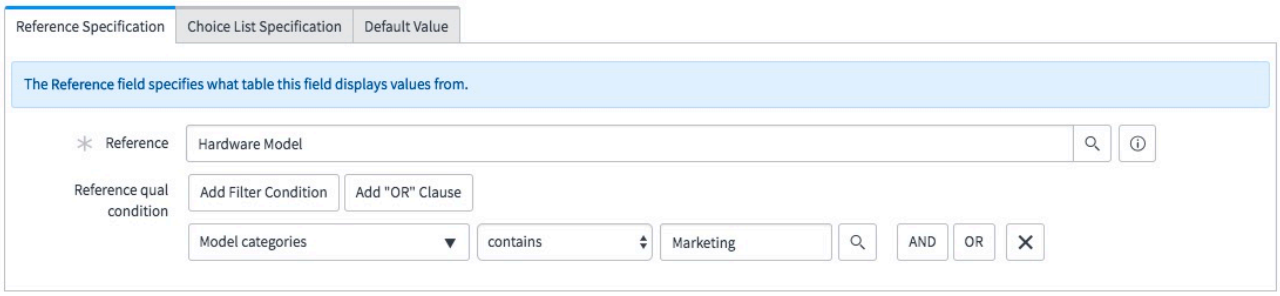
The form looks good. You can optimize the user experience even more by making reference lists more relevant. This reference list filtering can be done with a reference qualifier.

1. Open the **Equipment Request** table.
2. In the **Columns** section, scroll to **Type** and click on the column label.



Hint: it will be on the last page at the end

3. In the **Reference Specification** section, in the **Reference qual condition** field, set the filter condition to **Model categories contains Marketing** as shown below.



4. Click **Update**.

Now the application has two tables, forms to add new records, and lists to view all records. In the next lab, take your application for a spin!

Test Drive the Application

Lab 4
Test Drive

This lab explains how to verify what you have done in the previous labs.

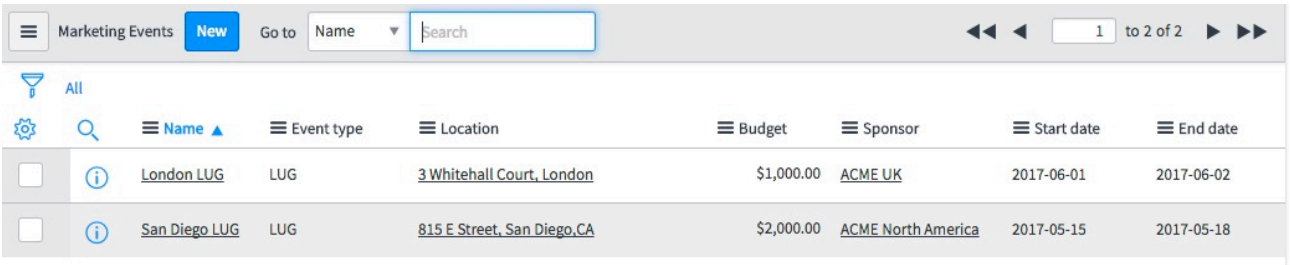
Enter Marketing Events

It is time to start enjoying the fruits of your labor. Add a few marketing event records.

1. Switch back to the main ServiceNow window and navigate to **Marketing Events > Events**.

2. Create marketing event records using the following information.

Name	Type	Sponsor	Budget	Location	Start date	End date
San Diego LUG	LUG	ACME North America	$2000.00	815 E Street, San Diego,CA	2017-05-15	2017-05-18
London LUG	LUG	ACME UK	$1000.00	3 Whitehall Court, London	2017-06-01	2017-06-02



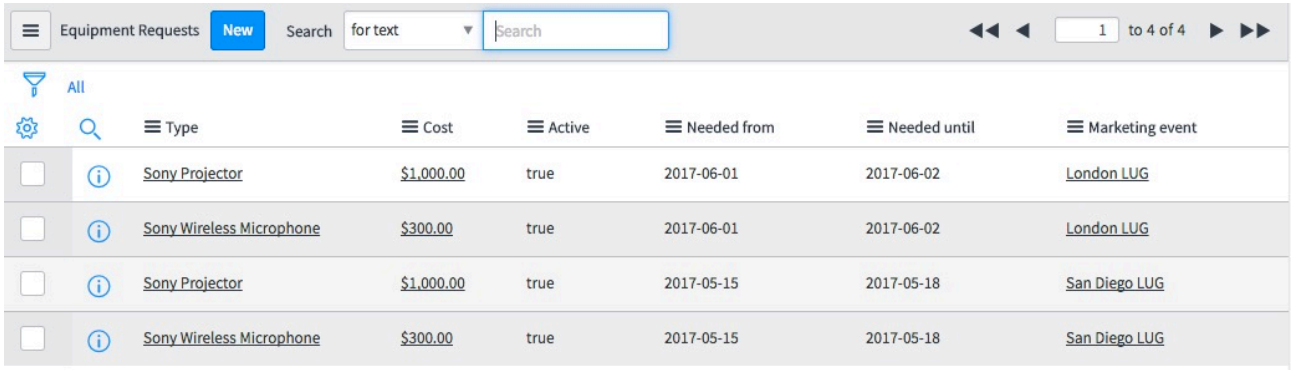
Now add some corresponding equipment request records.

Enter Equipment Requests

1. Navigate to **Marketing Events > Equipment Requests.**

2. Create equipment request records using the following information.

Type	Marketing Event	Needed from	Needed Until
Sony Projector	London LUG	2017-06-01	2017-06-02
Sony Wireless Microphone	London LUG	2017-06-01	2017-06-02
Sony Projector	San Diego LUG	2017-05-15	2017-05-18
Sony Wireless Microphone	San Diego LUG	2017-05-15	2017-05-18

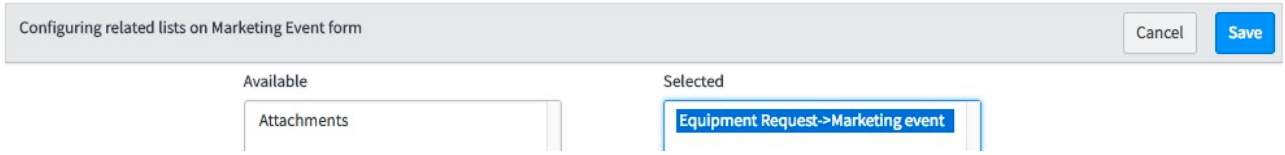


Note: You can sort by the column headings in the list view as well as edit any field by double clicking. Give it a try.

Add a Related List

Make one small enhancement to the application. When you look at a particular marketing event, you want to be able to see the equipment requested for that event on the same page. Since the Equipment Request table is related to Marketing Event table, you can do this easily.

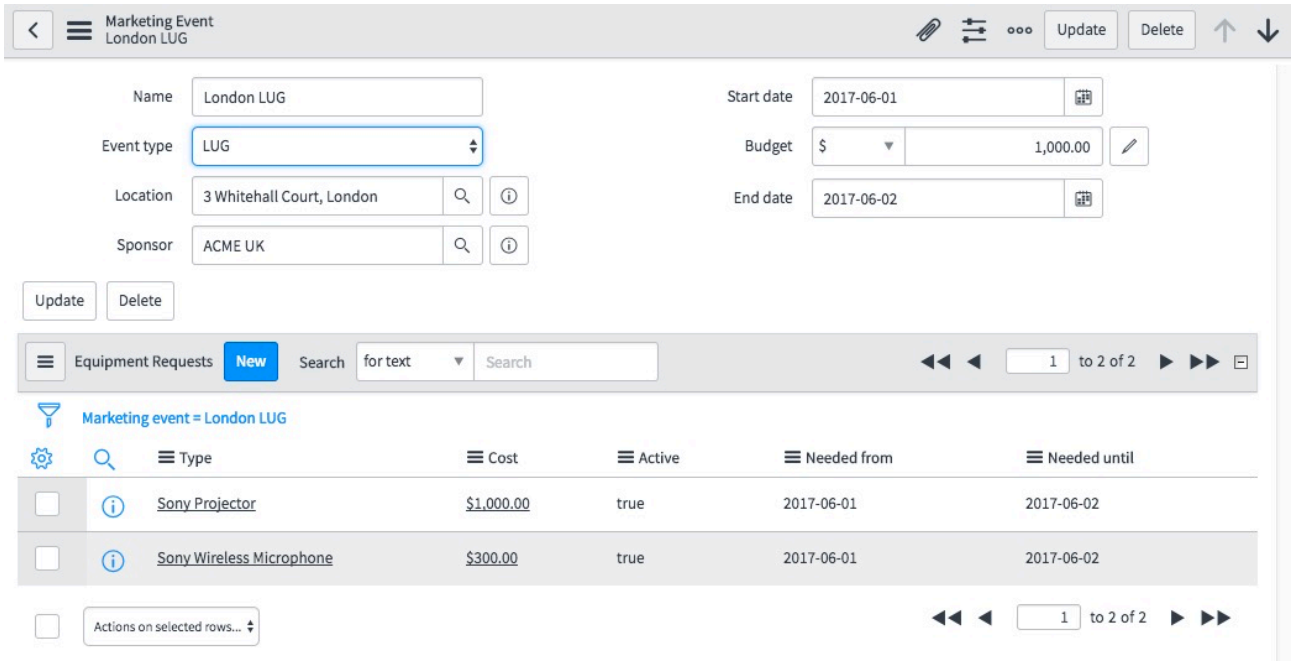
1. Back in **Studio**, **Create New Application File** and select **Forms & UI** > **Related List**.
2. Select the **Marketing Event** table and click **Create**.
3. From the **Available** list, select **Equipment Request->Marketing event** and click  .



4. Click **Save**.

5. Switch back to the main ServiceNow window and navigate to **Marketing Events** > **Marketing Event**

6. Click on the **London LUG** record. Notice that the related equipment requests for the event now show up in the same view.



You now have a functioning application hosted securely in the cloud. This is already a huge improvement over ad-hoc spreadsheet sharing. But you are not done with the application yet.

Extend the Data Model

Lab 5
Data Model Extension

In this lab you will:

- Create a new attendee table and link it to the marketing event table
- Create two new fields in the marketing event table:
  - Expected number of attendees
  - Actual registration

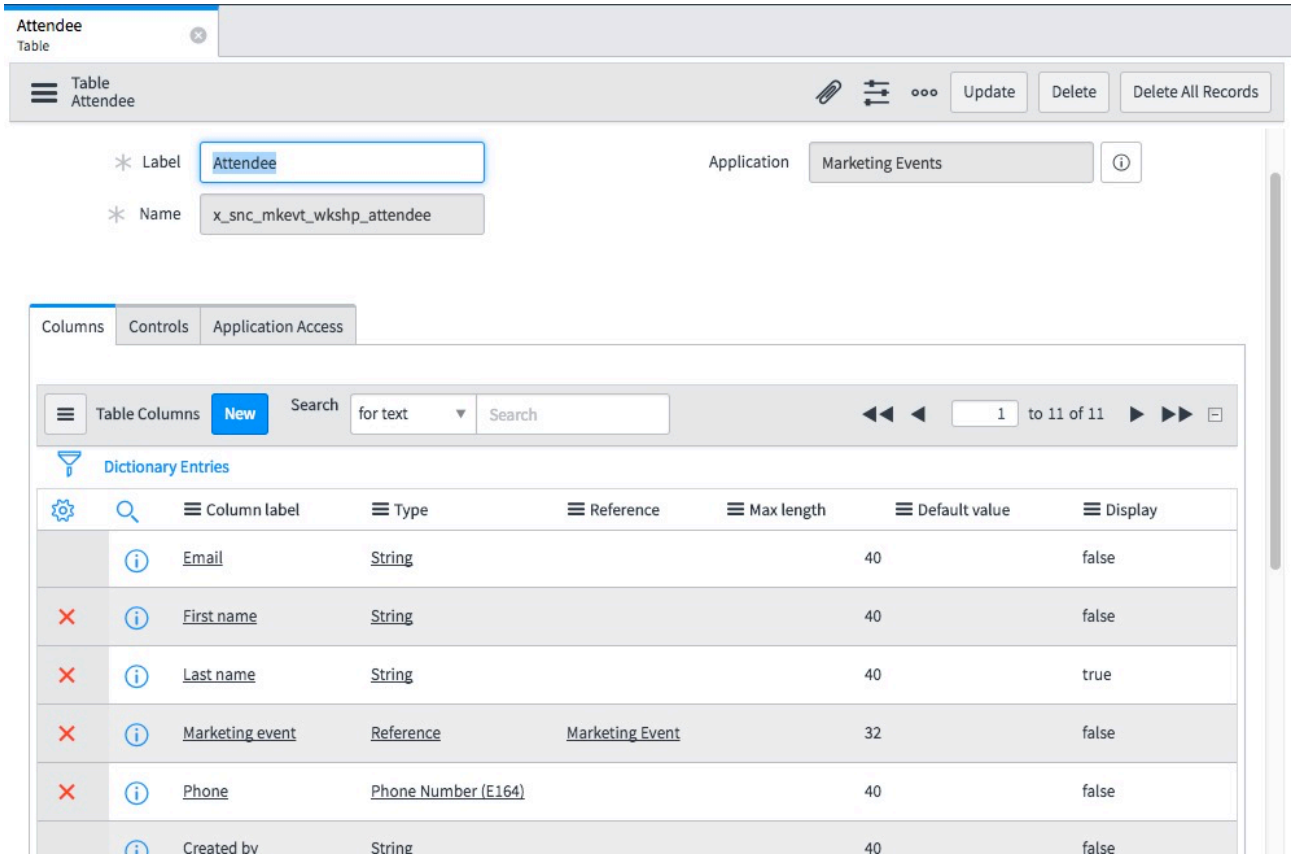
The process of creating a new table is the same as that outlined in **Create the Database Tables** lab.

Create the Attendee Table

1. In **Studio**, create a new table labeled **Attendee** with the following fields.

Column label	Type	Reference	Display
First name	String		False
Last name	String		True
Email	String		False
Phone	Phone Number (E164)		False
Marketing event	Reference	Marketing event [x_<YOUR_SCOPE_HERE>_marketing_events_event] *	False

* Note: the scope prefix (x_) on tables you create on your instance will vary from what you see in this guide.



2. Click **Submit**.

Modify the Marketing Events Table

1. Add these fields to the **Marketing Event** table.

Column label	Type	Display
Number of attendees expected	Integer	False
Number of attendees registered	Integer	False

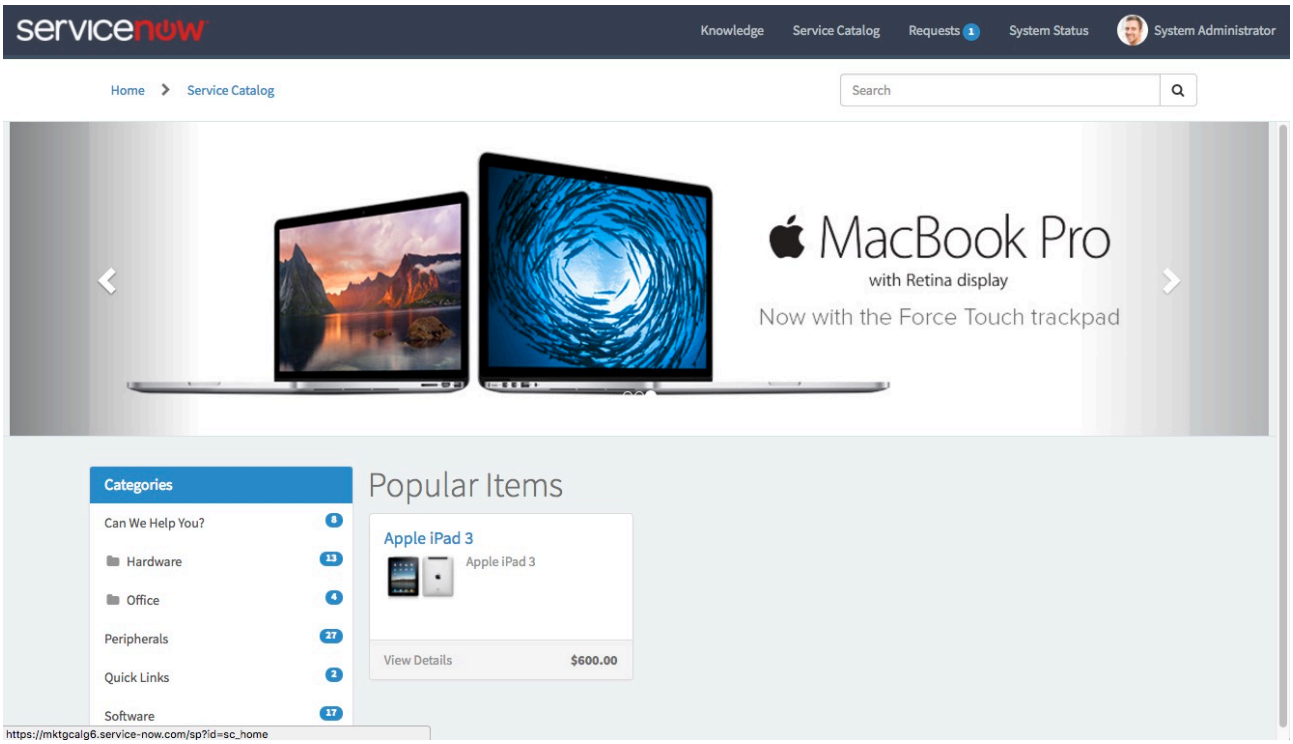
2. Click **Update**.

Make Requests Through the Service Catalog

Lab 6 Service Catalog

Service Catalog is a term that has its origins in IT Service Management. A Service Catalog is a representation of the Services offered to an organization by a service provider (IT) to their customers (the Business). Using the ServiceNow Service Automation Platform's built-in Service Catalog ensures that users have a single interface for all of their request-based needs without having to navigate multiple applications and interfaces. Multiple, independent catalogs can be managed while still giving requesters a single entry point and interface.

Here is a sample Service Catalog.



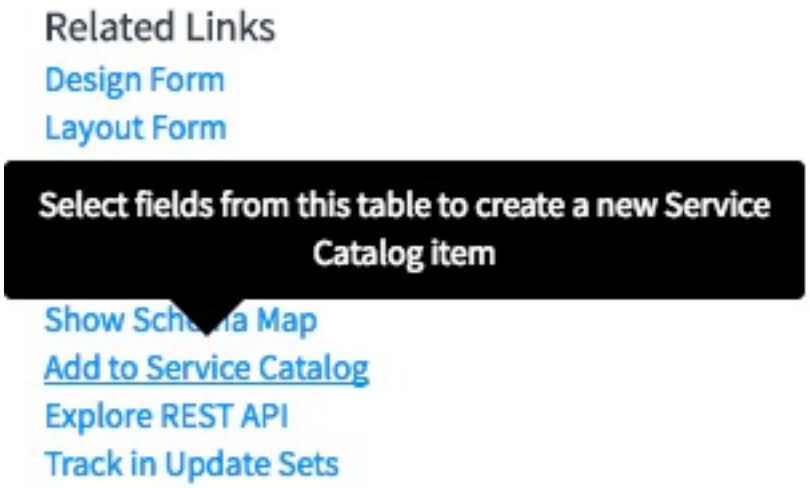
This lab explains how to create a Service Catalog for marketing event requests.

To implement the request, you create a Record Producer. A Record Producer creates an input form within the service catalog to capture requests and then uses the request data to create a new record in a

destination table. For this exercise, you want a Record Producer that creates new records in the Marketing Event table when someone uses the Service Catalog to request a new event.

Create a Record Producer

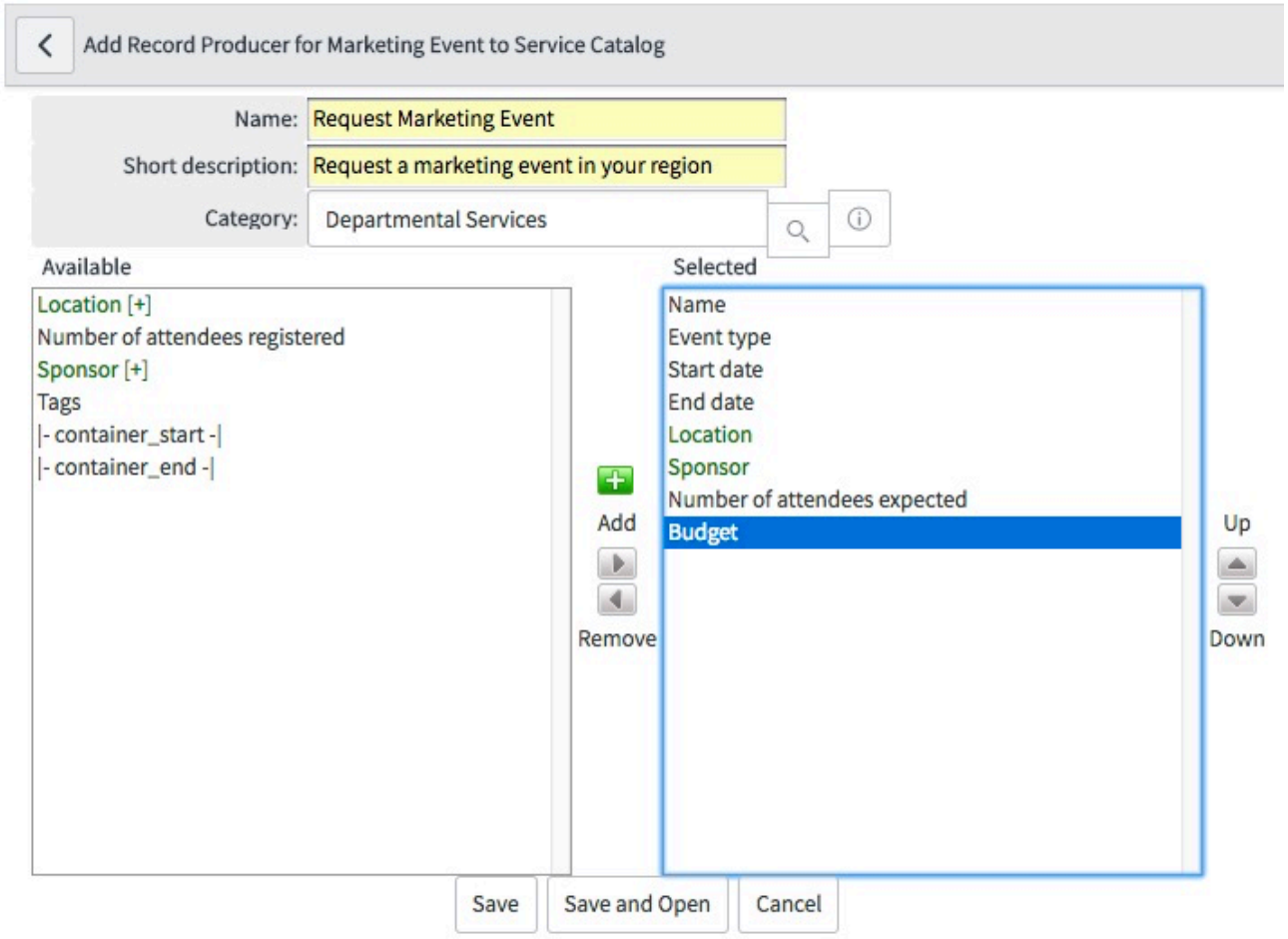
1. In **Studio**, open the **Marketing Event** table definition.
2. In the **Related Links**, click **Add to Service Catalog**.



3. Fill in the **Record Producer** form with the following values:

In this field…	Enter…
Name	Request Marketing Event
Short Description	Request a marketing event in your region
Category	Departmental Services

4. Add fields to the record producer by moving fields from the **Available** side to the **Selected** side in the order they are shown below.

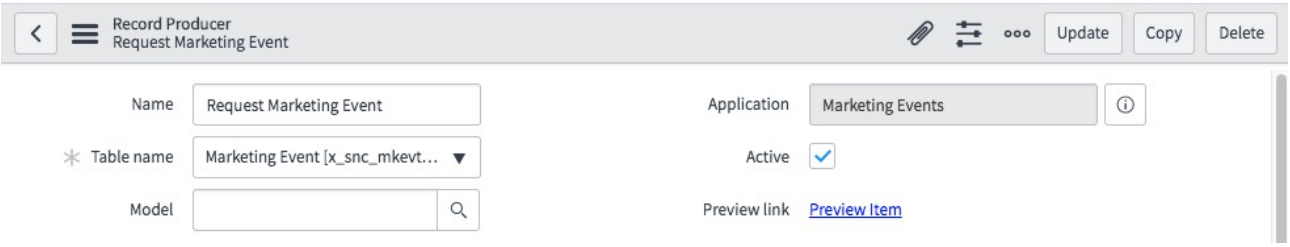


5. Click **Save** and **Open**.

Verify Results

Now that you created the variables, the Record Producer displays fields for users to fill out.

1. Click the **Preview Item** link. This provides a preview of record producer.



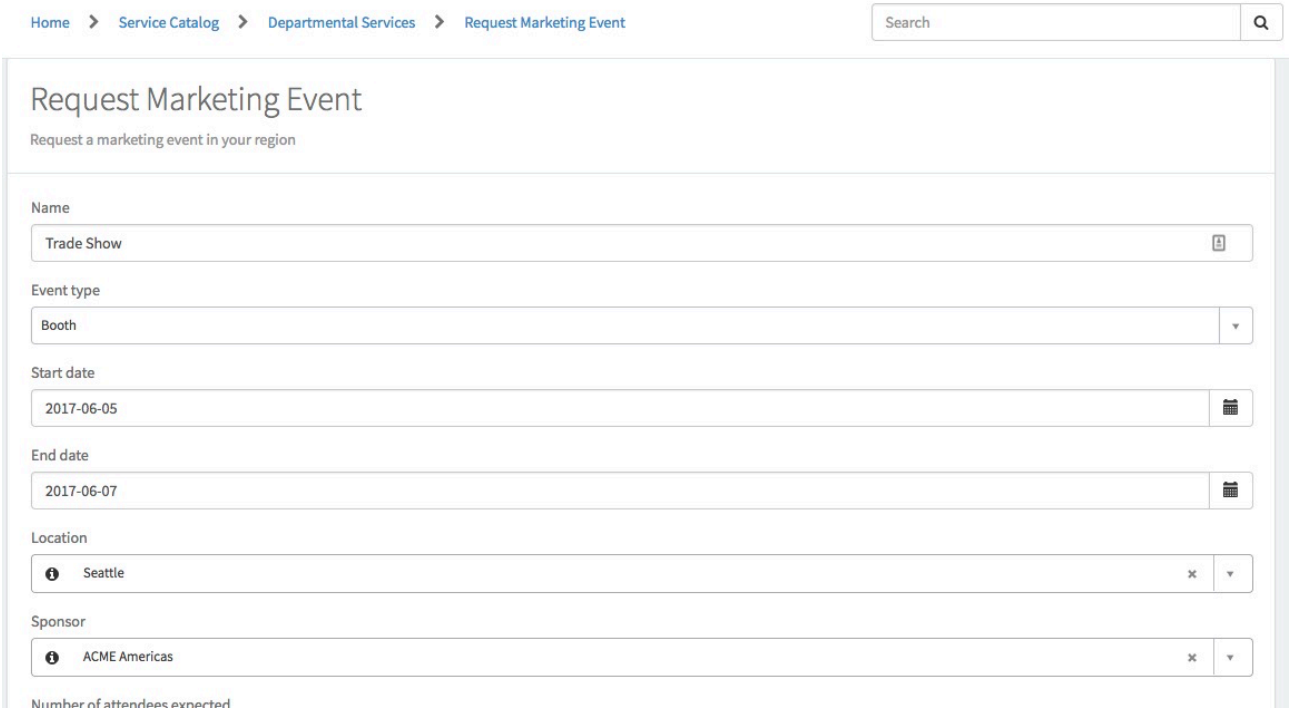
2. Test it out. Enter some data and click **Submit**. You are redirected to the newly created record.

3. Close the pop-up window.

4. Next try it from the Service Portal. In the main window, navigate to **Service Portal > Service Portal Home**.

5. Navigate to **Order Something > Departmental Services**.

6. Now select the **Request Marketing Event** service from the catalog. Add an entry.



7. Click **Submit**.

8. Verify that is has been added by switching back to the main window and navigating to **Marketing Events > Events**.

Congratulations! You created a new Service Catalog item for users to submit Marketing Event requests.

Automate with Workflows and Approvals

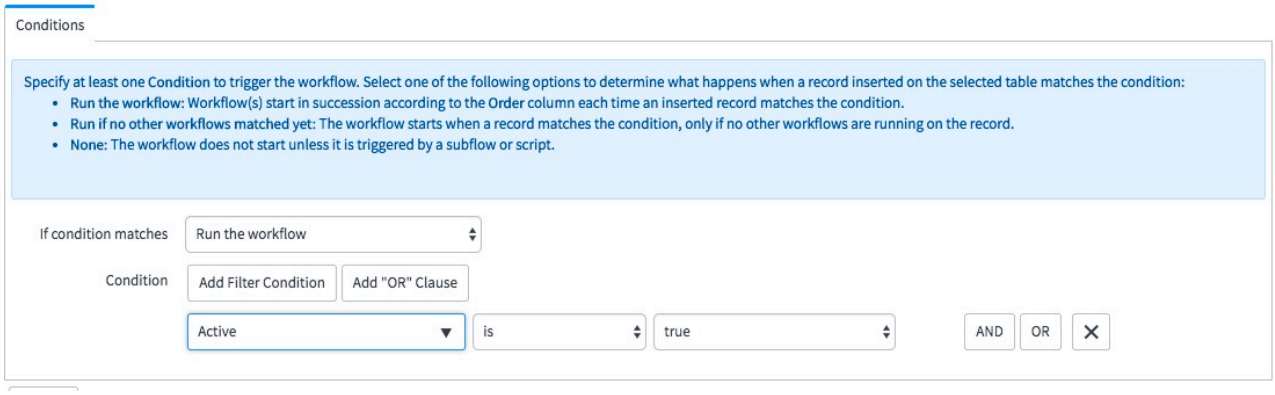
Lab 7
Workflow

Too often business processes are implemented by a combination of email, ad-hoc conversations and other inefficient, manual methods. Workflows can automate this, resulting in increased efficiency and reduced errors.

Start the Workflow

In this section, create the workflow container.

1. In **Studio**, **Create New Application File** and select **Workflow > Workflow**. A new browser window opens with the workflow editor.
2. In the Name field, type **Marketing Event Equipment Request**.
3. In the **Table** field, select **Equipment Request.**
4. You want the workflow to run only for active requests. Set the **Condition** to **Active is true**.

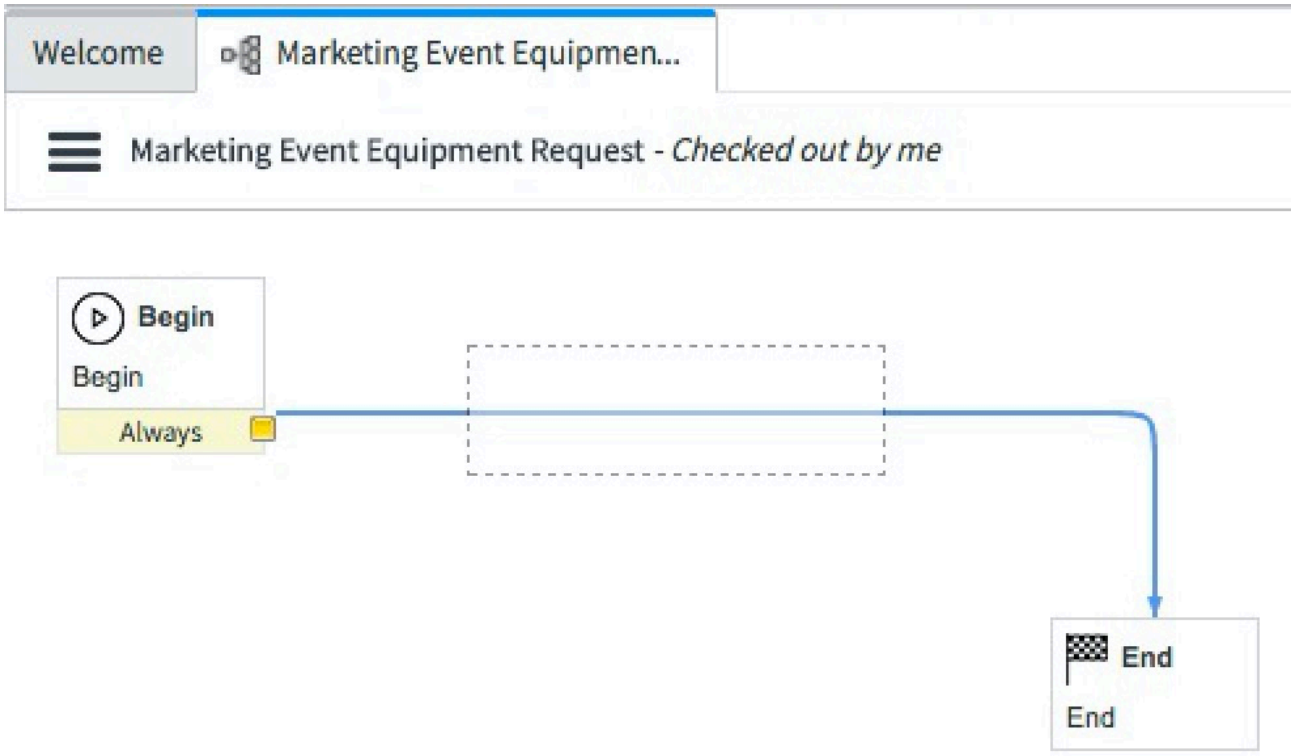


5. Click **Submit**. An empty workflow is shown. On the right, in the **Core** tab, is the palette of tasks and other activities you can add to the workflow.

Add an If Activity

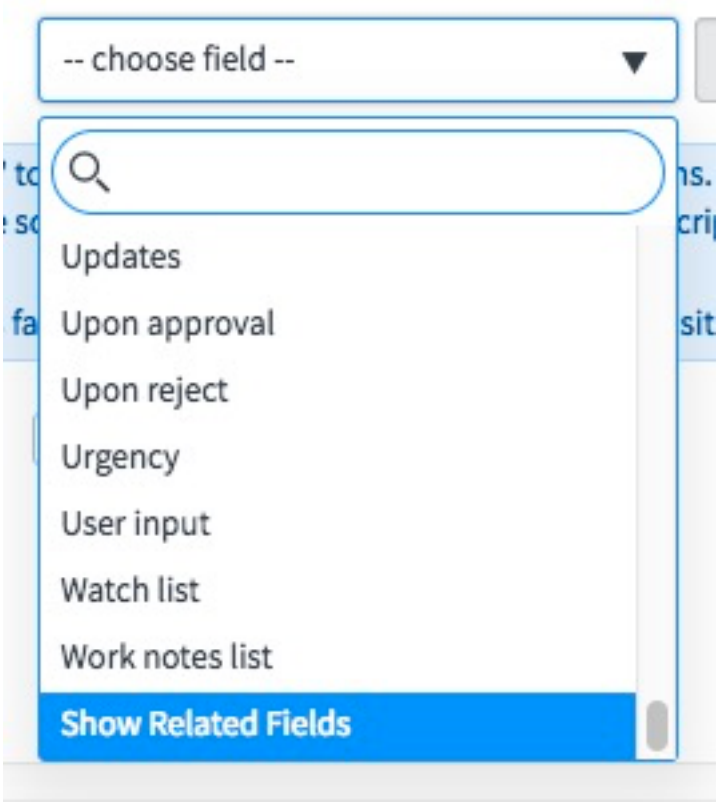
In this section, you add an activity to check if approval is required based on the equipment cost.

1. In the **Core** tab, expand the **Conditions** folder. Drag the **If** activity between the **Begin** and **End** activities. The connecting line will turn blue. Drop it there.



2. In the **New Activity: If** dialog **Name** field, enter **Over $500?**.

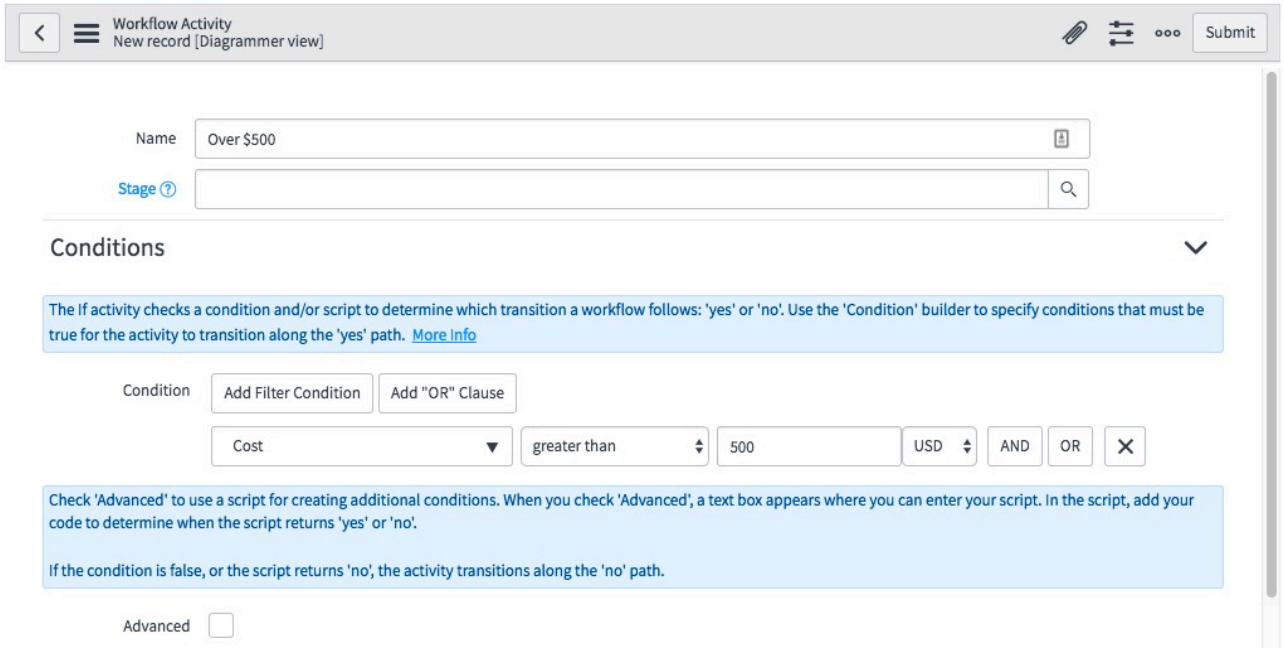
3. For the **Condition**, you will use the **Cost** field from the related **Hardware Model** table. To access the fields on the related table, from the **field** list, select **Show Related Fields**.



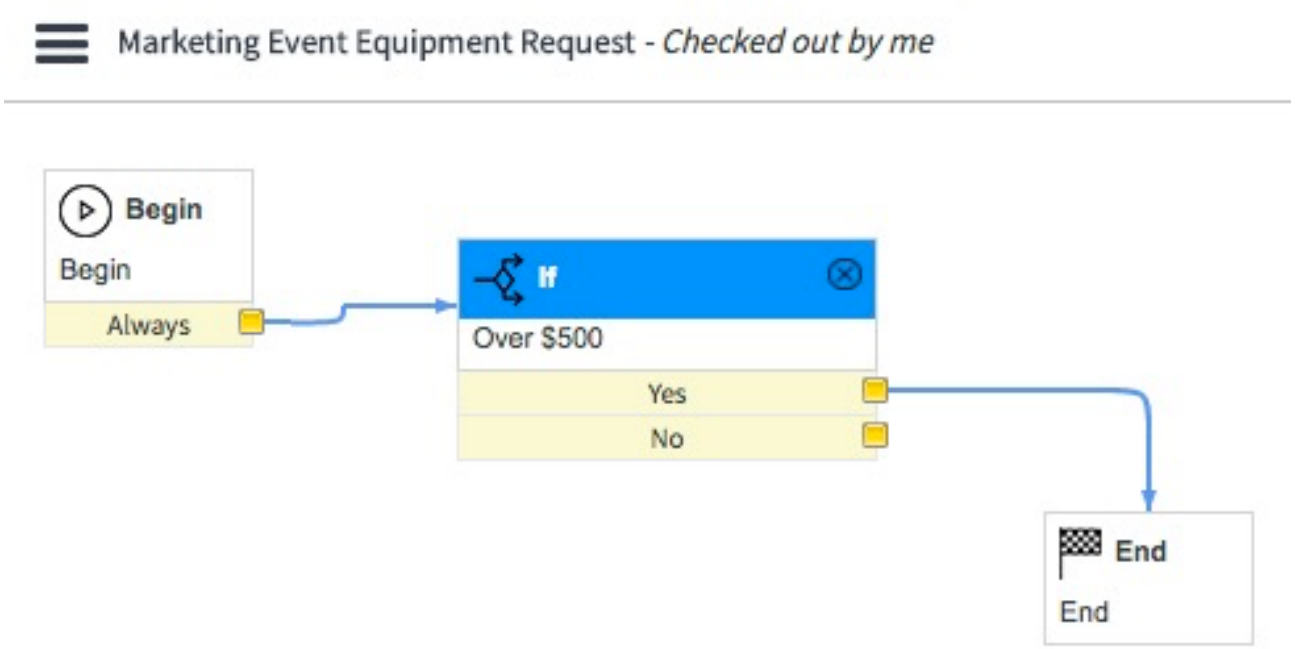
4. Select **Type** → **Hardware Model fields**.

5. Select **Cost**.

6. Complete the condition to read **Cost greater than 500 USD**. The completed form is shown below.



7. Click **Submit**.



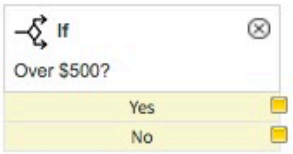
Add an Approval - User Activity

You want to track the status of the workflow in a field on the record itself so that the user who submitted the request can quickly know whether their request was approved or not. For this you use a field from the task table called Approval (remember that Equipment extends Task so you have access to all the fields in the task table). The Approval field can have four possible values as shown below—this works nicely for what you want to do.

- Not Yet Requested
- Requested
- Approved
- Rejected

In this section, you add manager approval.

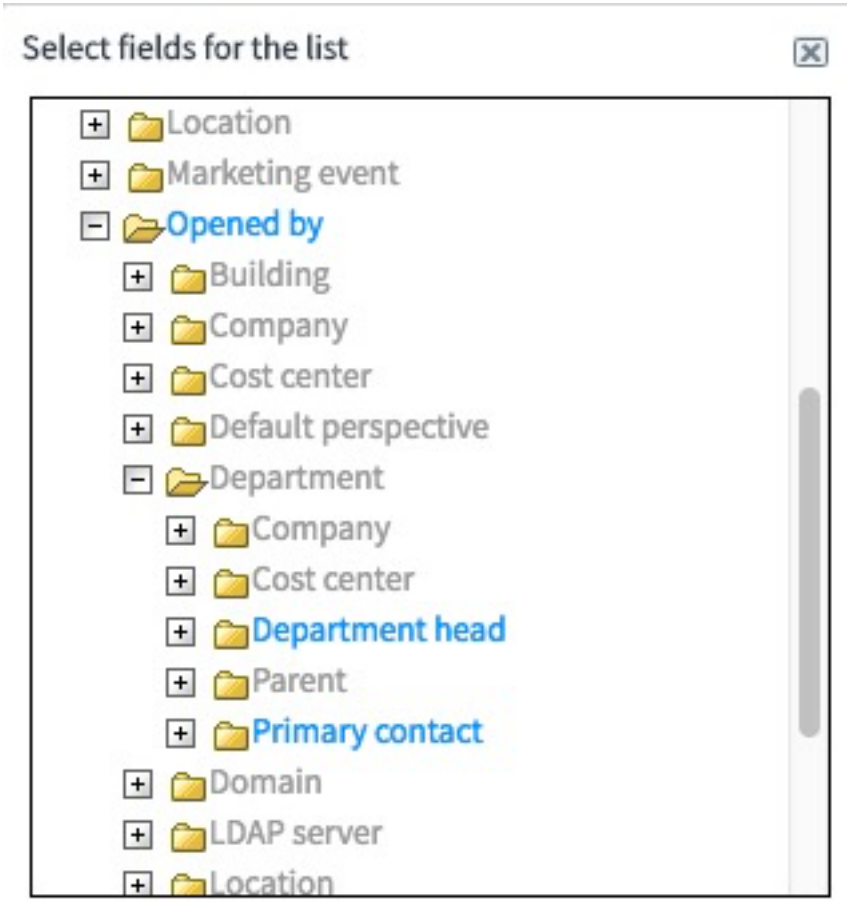
1. To prepare for the workflow for the new activity, click the line that connects the **If** activity and the **End** activity. Press **Delete**. Do the same for the line between **Begin** and **If**. Move the **End** activity down and to the right.



2. In the **Core** tab, expand the **Approvals** folder. Drag and drop the **Approval-User** activity to the right of the **If** activity.
3. In the **New Activity: Approval – User** dialog, in the **Name** field, type **Manager Approval**.
4. You can send the request to the requester's department manager. Click on  next to **Users.**

5. Click  .

6. In the **Select fields for the list** dialog, expand **Opened by**, expand **Department**, and click **Department head**.



7. Click  . You should see the variable **${opened_by.department.dept_head}** in the **Users** field.

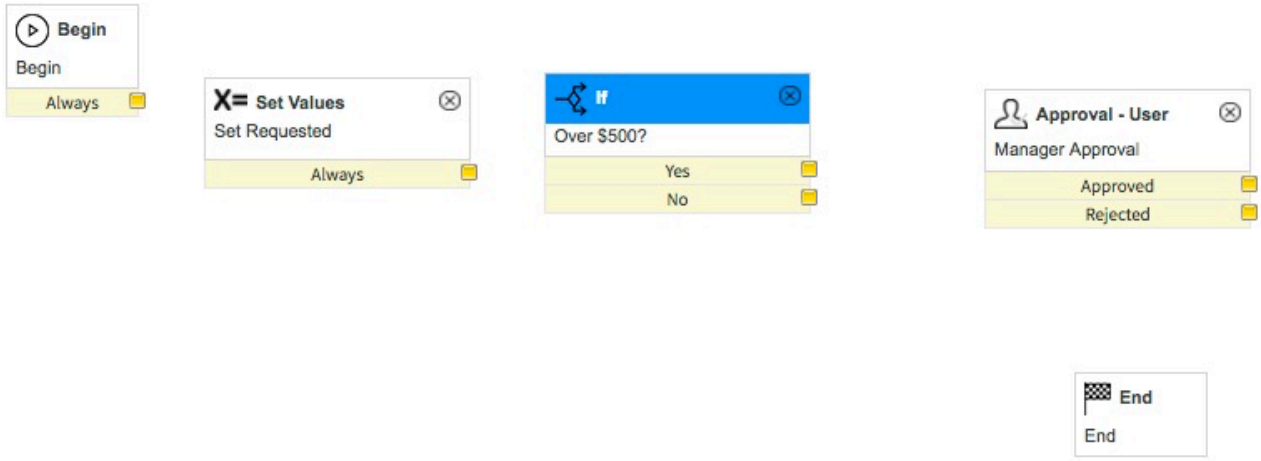
8. Click **Submit**.

Add Set Values

Now update the approval values as the request moves through the workflow

1. Expand the **Utilities** folder. Drag and drop the **Set Values** activity between **Begin** and **If**.
2. In the **Name** field, type **Set Requested**.
3. In the **Values** section, set **Approval** to **Requested**.

4. Click **Submit**. The workflow should now look like the example.



5. Add two more **Set Values** activities, one setting **Approval** to **Approved** and another setting **Approval** to **Rejected**.

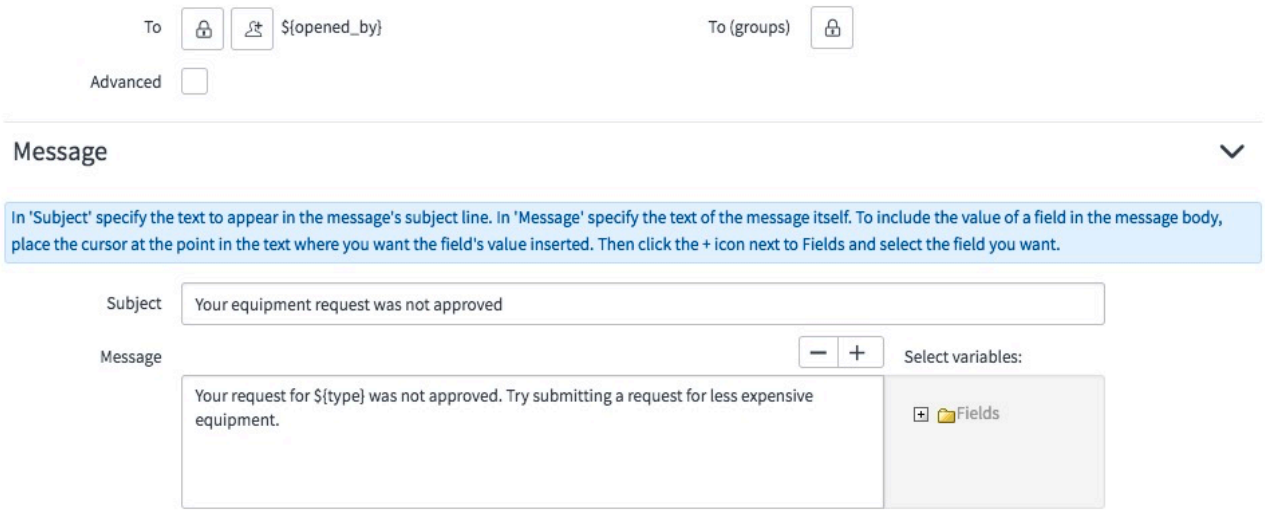
Add a Notification Activity

It would be nice for the requester to be notified if the request is rejected. Use a Notification activity for requester notification.

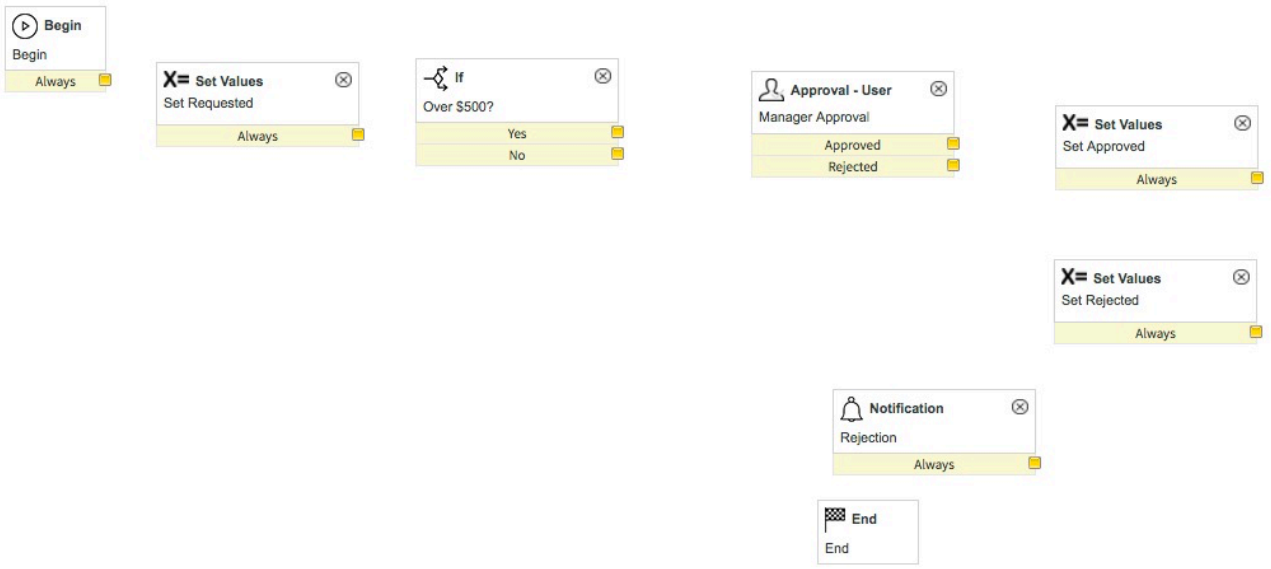
1. In the **Core** tab, expand the **Notifications** folder, drag the **Notification** activity to the canvas and drop it above the **End** activity.
2. In the **New Activity: Notification** dialog, in the **Name** field, type **Rejection**.
3. Unlock the **To** field. Using the **Select fields dialog box** as you did before, select **Opened by**.

4. In the **Subject** field, type **Your equipment request was not approved**.

5. In the **Message** field, type **Your request for ${type} was not approved. Try submitting a request for less expensive equipment**.



6. Click **Submit**. Your workflow should look like the example.



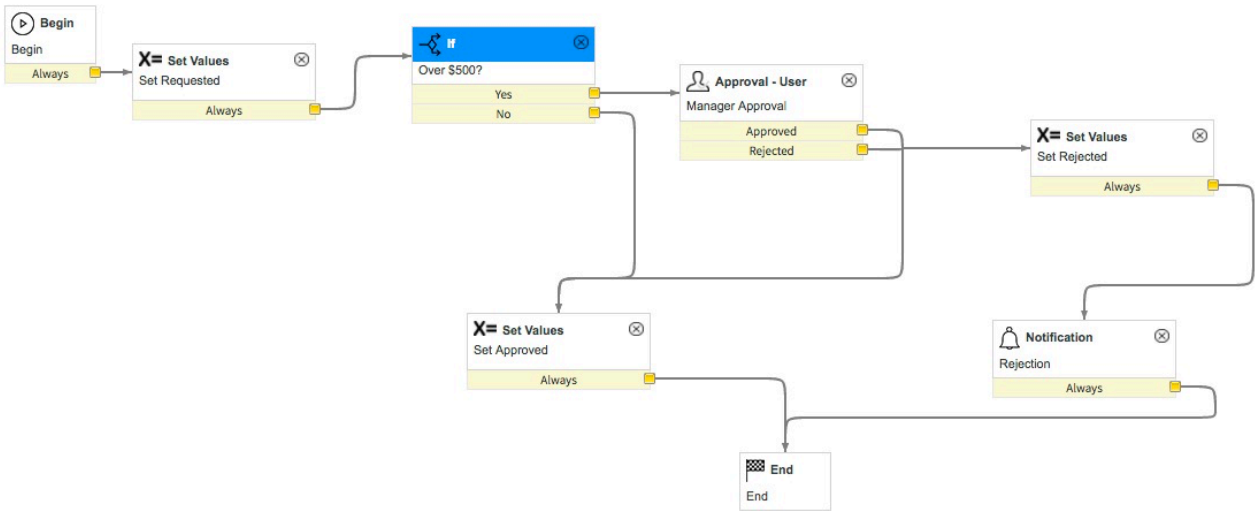
Connect the Activities

Now that all of the activities are in place, connect activity results to the appropriate next step in the process.

1. To connect activities, drag the square next to the activity result to the target activity. Alternatively, you can right-click on the activity result and select **Link to…** to use a dialog to make the connections. Use this table to make the connections:

From activity…	Drag result…	To activity…
Begin	Always	Set Requested
Set Requested	Always	If
If	Yes	Approval - User
If	No	Set Approved
Approval – User	Approved	Set Approved
Approval – User	Rejected	Set Rejected
Set Approved	Always	End
Set Rejected	Always	Notification
Notification	Always	End

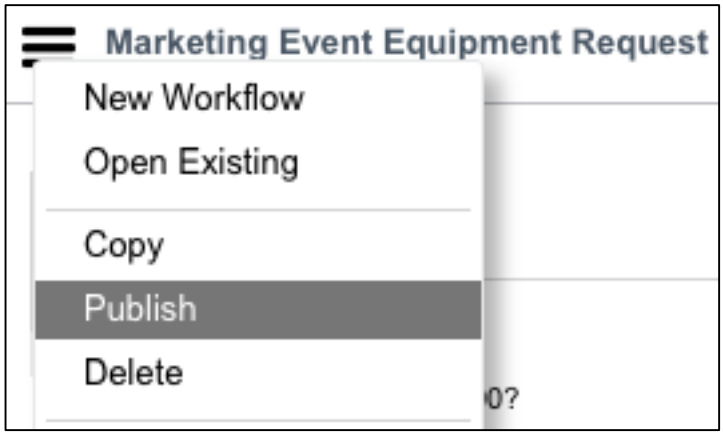
2. Rearrange the activities to make the process flow more readable.



Publish and Test the Workflow

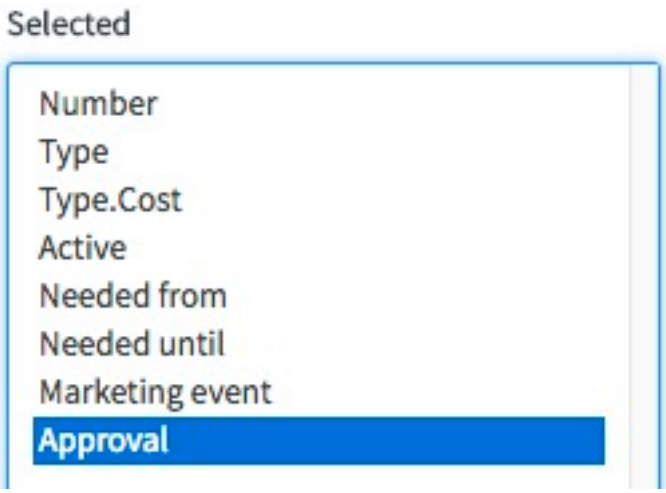
The workflow is complete and ready to be activated. In this section, you will publish the workflow.

1. In the **Workflow Actions** menu, select **Publish**. The status at top should now show **Published**.

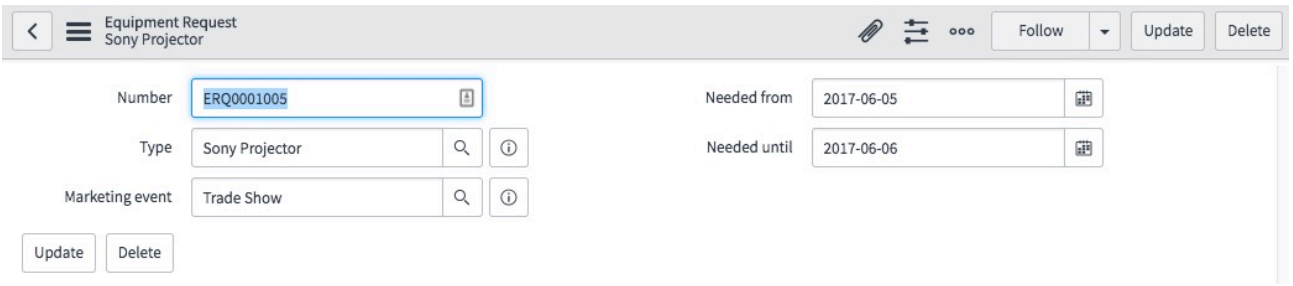


2. Before you test the workflow, make a simple addition to the Equipment list view. In Studio, open **Forms & UI > List Layouts > Equipment Request**

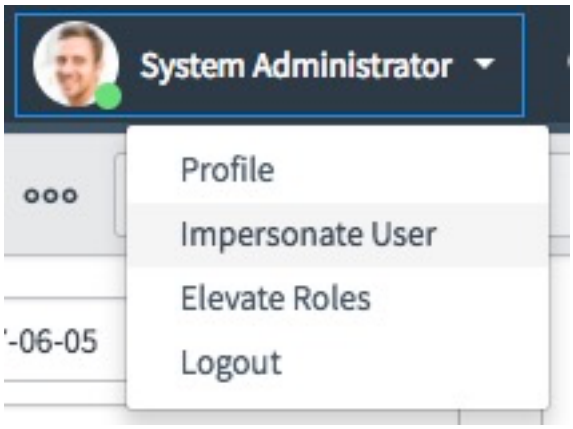
3. Configure the **Equipment Requests** list view and add the **Approval** field.



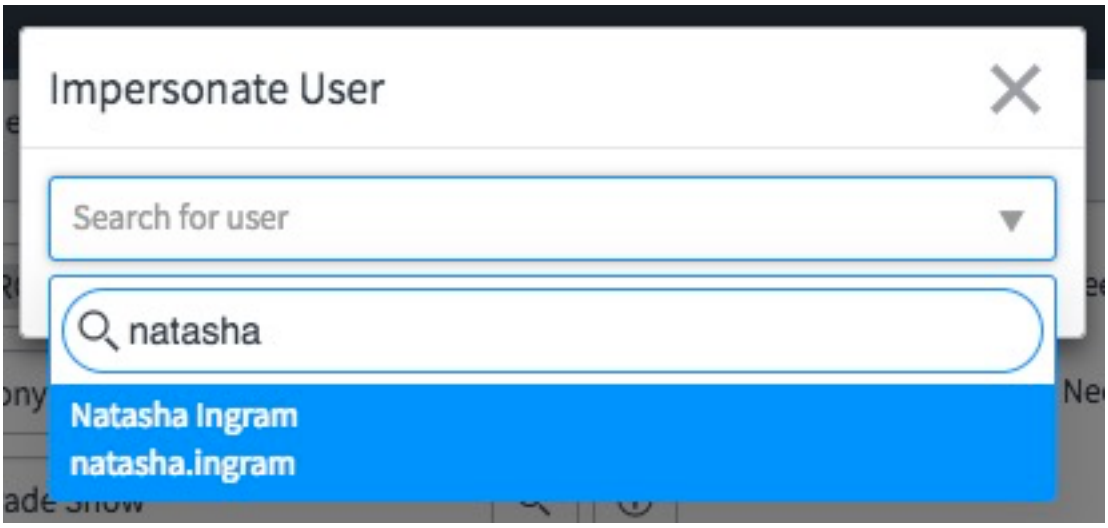
4. Now test the approval process. In the main application window, enter a new equipment request for an equipment **Type** that costs more than **$500** (see sample below) and click **Submit**.



5. To see if someone got the approval request, you first need to log in as Natasha Ingram who part of the approval chain. As a System Administrator, you can impersonate other users in the system. Click your username in the banner area select **Impersonate User**

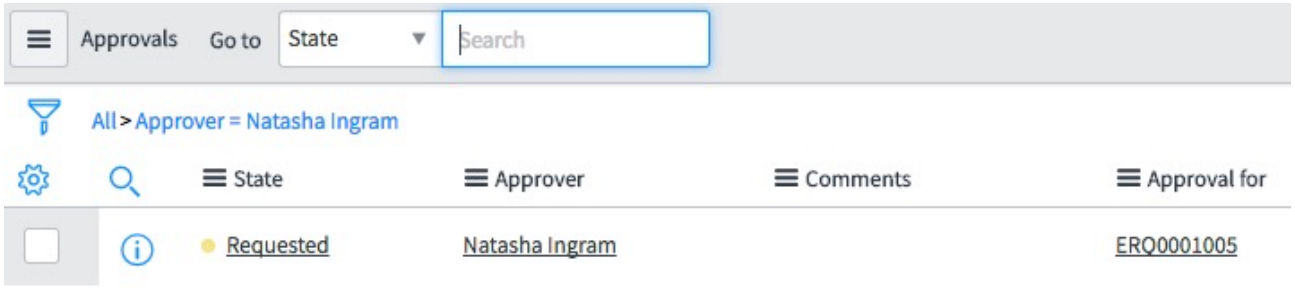


6. In the dialog that appears search for **Natasha Ingram**.



You should now be impersonating Natasha Ingram.

6. Navigate to **Self-Service > My Approvals**. A list view of all the approval requests appears.

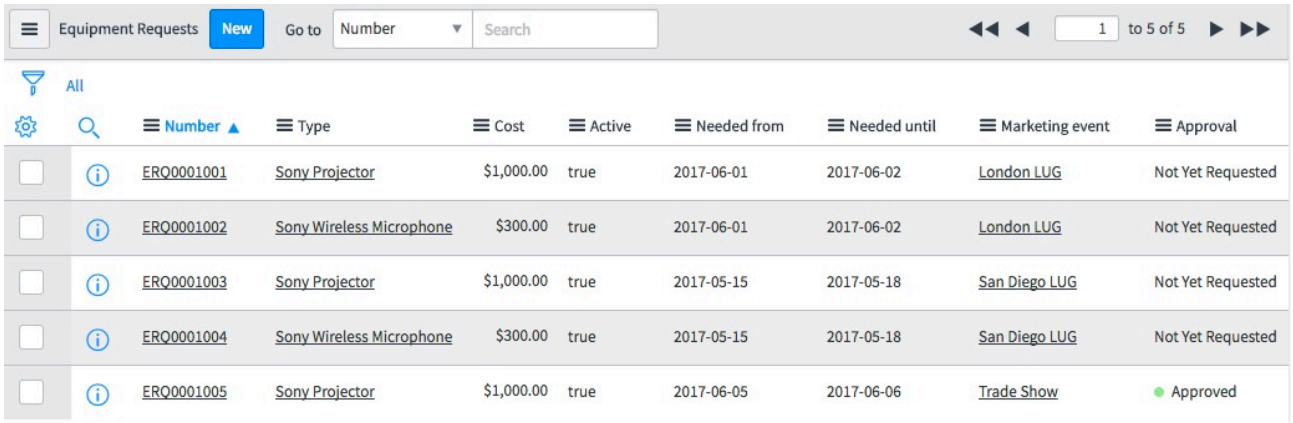


Click the **Requested** link for the recently submitted request.

7. Click **Approve**.

8. Now go back to the role of System Administrator. Click **Natasha Ingram in the banner** again, **Impersonate User** and click **System Administrator**.

9. Navigate to **Marketing Events > Equipment Requests**. The recently approved item should appear with the status **Approved**.



Number	Type	Cost	Active	Needed from	Needed until	Marketing event	Approval
ERQ0001001	Sony Projector	$1,000.00	true	2017-06-01	2017-06-02	London LUG	Not Yet Requested
ERQ0001002	Sony Wireless Microphone	$300.00	true	2017-06-01	2017-06-02	London LUG	Not Yet Requested
ERQ0001003	Sony Projector	$1,000.00	true	2017-05-15	2017-05-18	San Diego LUG	Not Yet Requested
ERQ0001004	Sony Wireless Microphone	$300.00	true	2017-05-15	2017-05-18	San Diego LUG	Not Yet Requested
ERQ0001005	Sony Projector	$1,000.00	true	2017-06-05	2017-06-06	Trade Show	Approved

Notice that in this example you can change the approval field status directly. Though this may be fine in this example, in a real world scenario you may want to make this field read only for the user who submits the request.

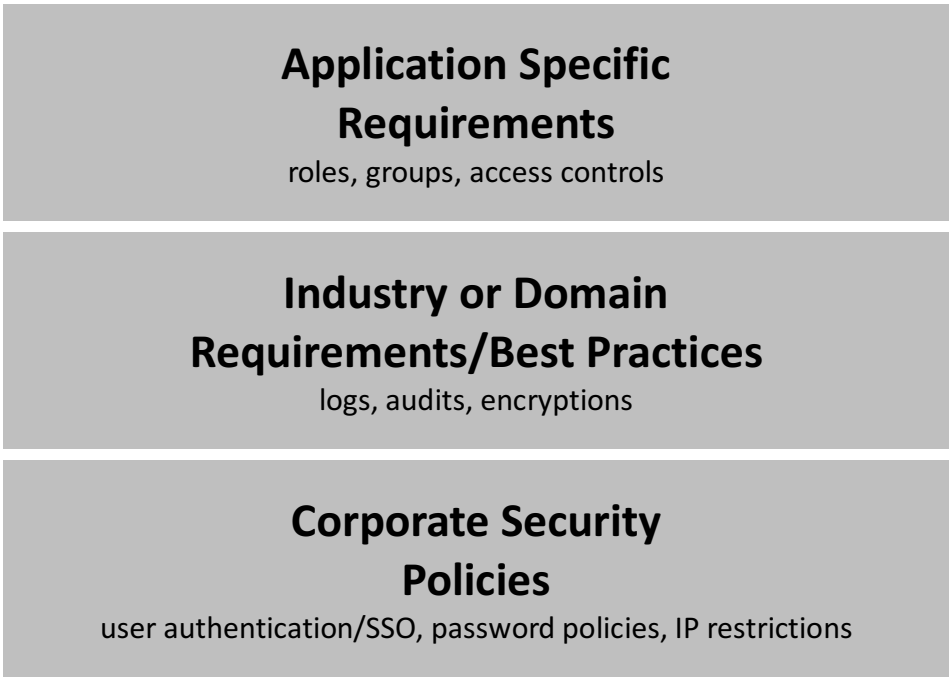
Workflows and approvals are extremely powerful features. With the ServiceNow Service Automation Platform this functionality is easily accessible by your applications.

Secure the Application

Lab 8
Security

Security is an important and multifaceted topic. The ServiceNow Service Automation Platform provides a variety of administrative controls and a fine-grained application security model to secure your applications.

The diagram below depicts one way to think about the security of the applications and some of the platform capabilities provided to address security requirements. At the bottom are the controls that are mandated by your company's security policies. On top of that may be industry-specific requirements and best practices. These two layers may be common across multiple applications. Finally, there is application-specific security that controls access to an individual application.



The platform security capabilities should not be looked at in isolation but should be used together to secure the application in an optimal fashion. In this tutorial, we defer security to a later lab. As a best practice, you should understand the security requirements for your applications and incorporate them into earlier stages of the implementation cycle.

In this lab you look at two of the capabilities provided by the ServiceNow platform to secure your applications— roles and access control lists.

A role is a category that can be assigned to a user and can be granted access to particular parts of the system—in this case the marketing events application. When you created the application (in lab 1), it conveniently created a new role (**x_<YOUR_SCOPE_HERE>.user**) that you can assign to users that need access to this application.

Test Default Application Role

Review the impact of having and not having the **x_<YOUR_SCOPE_HERE>.user** role.

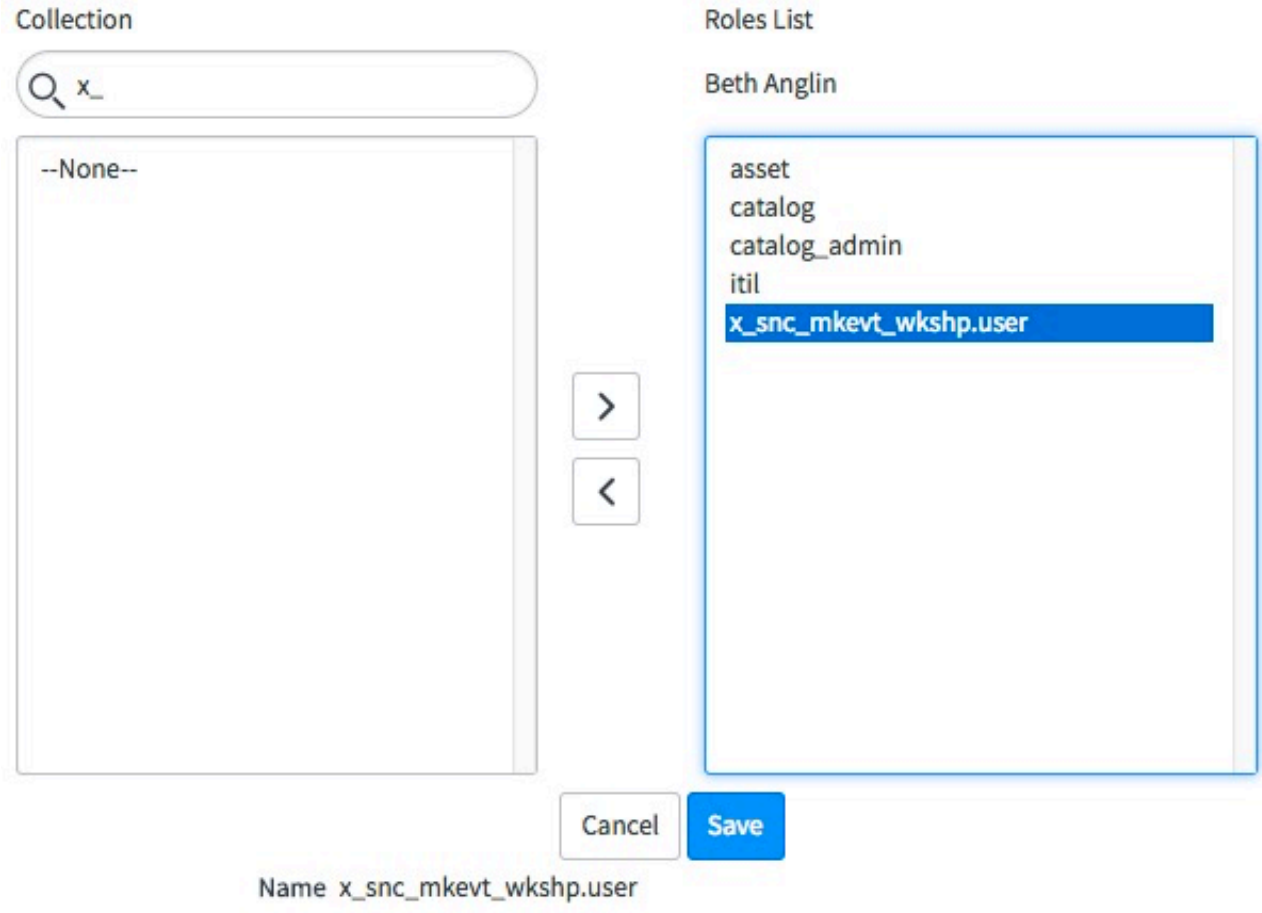
1. To test the current application security, **impersonate** Beth Anglin who does not have this role. If you trying to navigate to the **Marketing Events** application, it should not be visible.

2. Go to the **System Administrator** role to give Beth Anglin the role **x_<YOUR_SCOPE_HERE>.user**.

3. In the main window, navigate **System Security > Users and Groups > Users** and open the user record for **Beth Anglin**.

4. In the **Roles** section, click **Edit**.

5. In the form, search for **x_<YOUR_SCOPE_HERE>.user** and add it to the **Roles List**. Click **Save**.



6. **Impersonate Beth** again and navigate to **Marketing Events**. The application should appear.

7. After you confirm this, return to the **System Administrator** role to continue the lab.

Understand Access Control Lists (ACLs)

Defining a role gives visibility to the application only to users with that role. However, you may want to further restrict what a particular role or user is allowed to do. Access Control Lists is one of the mechanisms provided to restrict access at a fine level of granularity. Though we do not cover ACLs in this lab, it is an important concept to be aware of to secure your applications.

Enforce Business Rules

Lab 9 Business Rules

You have built a fairly sophisticated system so far without having to write a single line of code. This is by design. The ServiceNow Platform is designed to help you build your application declaratively, with mouse clicks. However, there may be instances in which the declarative features are not sufficient. In such cases, you can choose to write code. Almost all ServiceNow platform capabilities (including workflow, security, UI, and database) can be customized with JavaScript code.

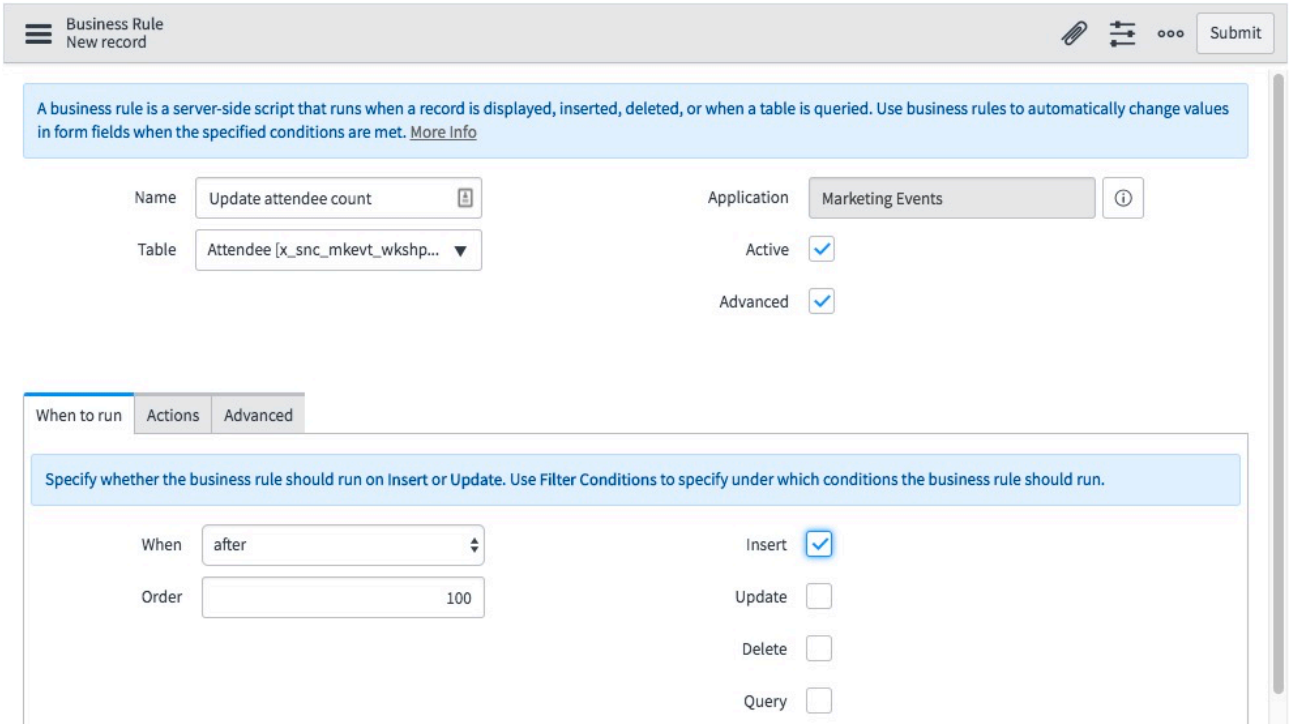
This lab explores Business Rules, one of the more common scenarios that can benefit from custom code. Business Rules can be made to execute before or after a database operation like insert, update or delete. If you are familiar with relational database triggers, you will notice the similarity - Business Rules are indeed similar to triggers in relational databases.

You will write a Business Rule to calculate the number of attendees and update the registered attendee field on the marketing event table. This is a simple rule for illustrative purposes. Code written for the real world can be as complex as you wish.

Create a Business Rule

1. In **Studio**, **Create New Application File** and select **Server Development > Business Rule**
2. Fill out as shown below.

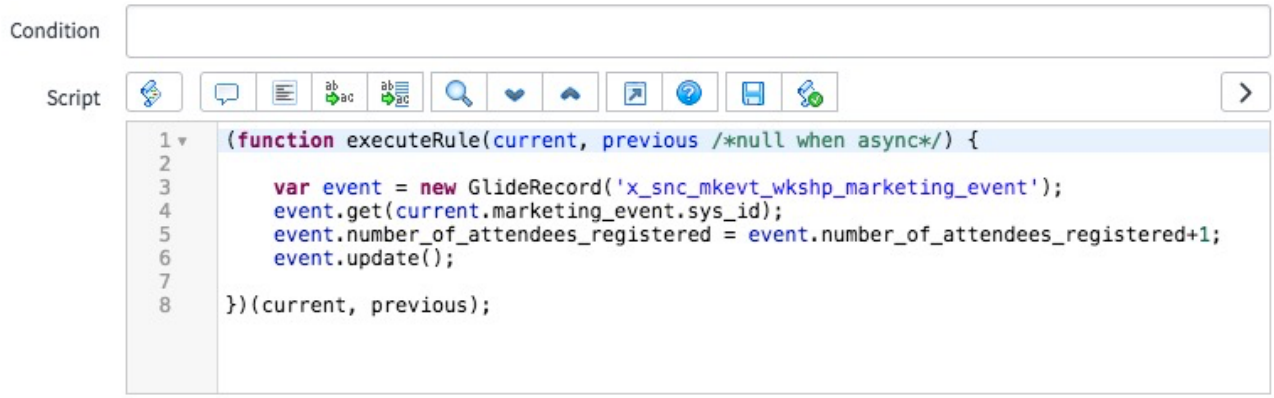
In this field…	Enter..
Name	Update attendee count
Table	Attendee [x_**<YOUR_SCOPE_HERE>** _attendee]
Advanced	Checked
When	After
Insert	Checked



3. In the **Script** field, type the following code (also found in **Lab 9/Step3_BusinessRule.js.txt**) where the comment "// Add your code here" appears. And remember to match the table name in the first line to your application

   Note: `current` refers to the Attendee record in the current insert transaction. `marketing_event` is a field on the current record. `number_of_attendees_registered` refers to the field **Number of attendees registered** on the **Marketing Event** table.

```
  var event = new GlideRecord('x_<YOUR_SCOPE_HERE>_event');
  event.get(current.marketing_event.sys_id);
  event.number_of_attendees_registered =
event.number_of_attendees_registered+1;
  event.update();
```



4. Click **Submit**.

5. Just as we did for **Equipment Requests** earlier, add a **Related List** to the **Marketing Event** table, this time adding **Attendee**.

6. Test by adding a few attendees for any of the events. The current registration count in the **Number of attendees registered** should reflect the inserts.

The ServiceNow Platform enables you to build most of the functionality that you need with clicks. However as an application creator you never have to worry about hitting a wall or limitation of the platform— you can always customize with code when needed. In this lab, you looked at Business Rules that execute when a database operation occurs. Code-based customization is also an available option for hundreds of other platform capabilities such as UI, security and workflows.

Enhance the User Experience

Lab 10
Service Portal Basics

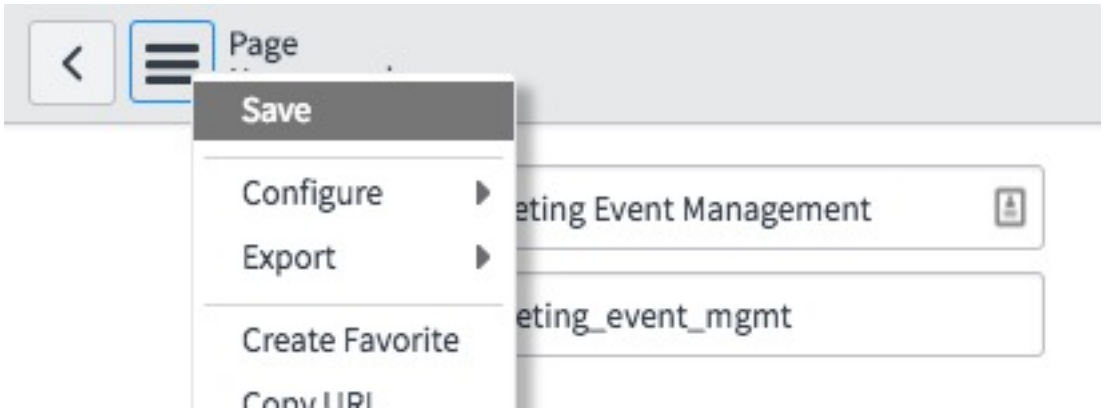
We've already seen how defining a table also creates lists and forms for the user to interact with data in the table. Using Service Portal, we can quickly assemble those components into a responsive and mobile-friendly user experience.

Create a Page

1. In the main application window, navigate to **Service Portal** > **Pages** and click **New**.
2. Fill out the form as shown below.

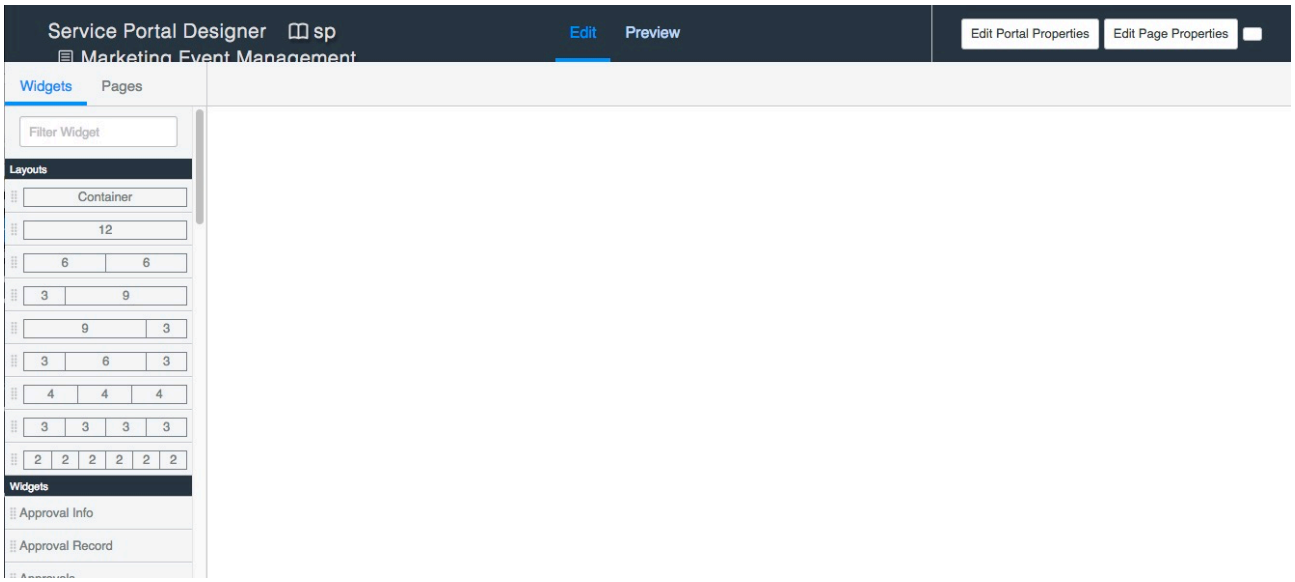
In this field…	Enter..
Title	Marketing Event Management
ID	marketing_event_mgmt

3. Click on the page menu and select **Save**.

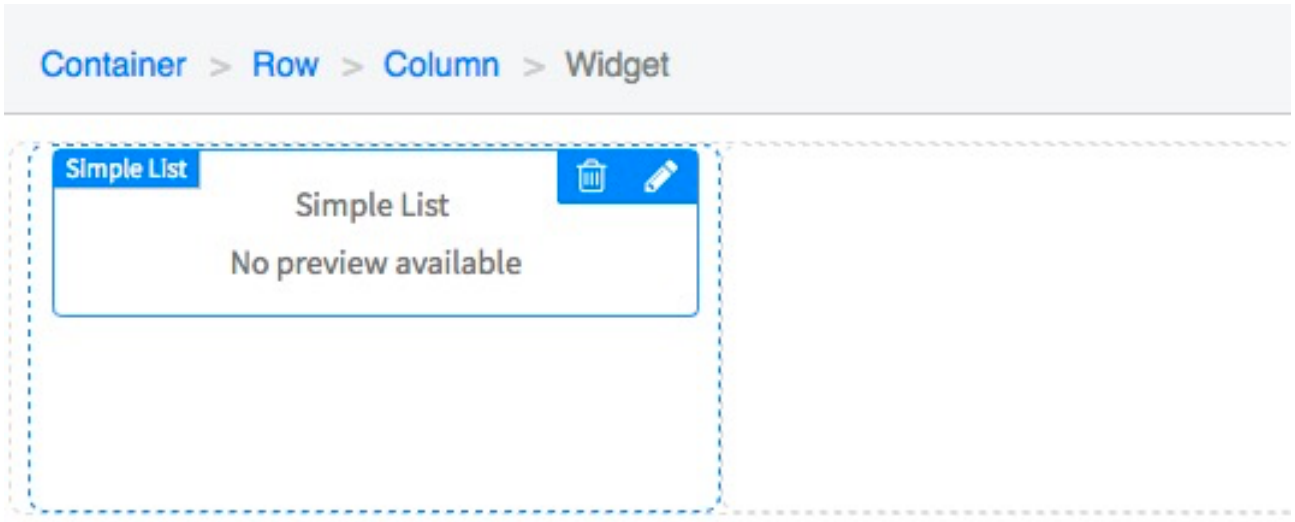


4. Under **Related Links**, select **Open in Designer**.

This opens the Service Portal Designer in a new window



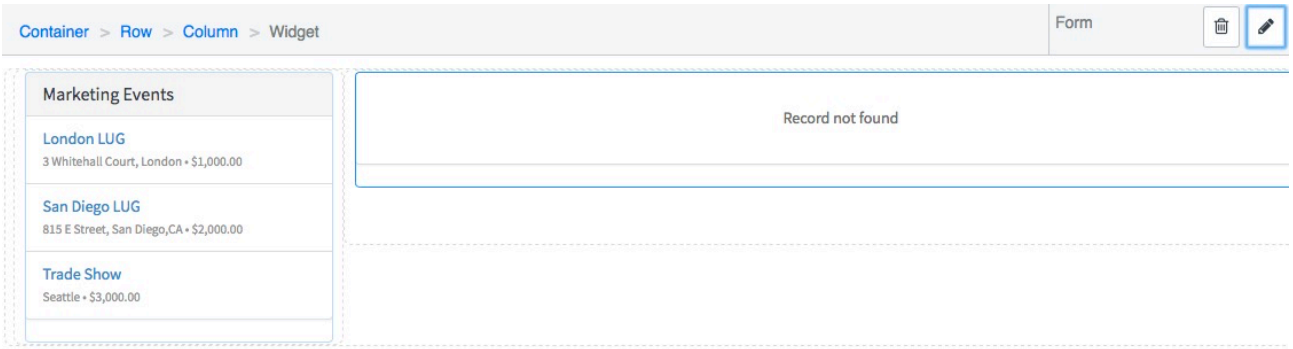
5. From the **Layouts** section, drag a **Container** to the page canvas.
6. Drag a **3/9** layout into the container.
7. From the **Widgets** section, drag a **Simple List** into the left content block
8. In the **Simple List** widget, click the  to edit the widget's properties



9. Fill out the property values as shown below and click **Save**.

In this field…	Enter..
Table	Marketing Event
Display field	Name
Secondary fields	Location, Budget

10. Drag **Form** from the widget list into the right content block. Your page should look similar to this:



11. Congratulations! You have built your first custom Service Portal Page

Testing the page

12. At the top of the Service Portal Designer, click **Preview**.

13. Select a marketing event from the list on the left, note how the form automatically updates.

14. Click on the device icons in the upper left to quickly see how the page will render on phones and tablets

Build Custom Controls

Lab 11
Service Portal Widgets

Using the out of the box widgets lets you build responsive UIs quickly, but Service Portal also let you easily build custom widgets unique to your application. In this module we'll build an Attendee registration page with a custom widget.

Create a Custom Widget

1. Back in the main application window, navigate to **Service Portal > Service Portal Configuration**

2. The **Service Portal Configuration** tool pops up in a new window. Select the **Widget Editor**.

3. Select **Create a new widget**.

4. In the **Widget Name** field, type **Attendee Registration** and hit **Submit**

Service Portal widgets are miniature AngularJS apps with all of the plumbing in place to call server-side code. If you are already familiar with Angular, then you're ready to start building custom widgets for Service Portal. If not, the code needed for the widget is below.

5. Copy the block of HTML below (also found in **Lab 11/Step5_HTMLTemplate.html.txt**) and paste it into the **HTML Template** field, replacing all of the existing text.

```
<div>
  <div class="container">
    <form class="form-horizontal">
      <!-- Event picker -->
      <div class="form-group">
        <label for="mktg_event">Event</label>
        <sn-record-picker field="evt"
                          table="'x_<YOUR_SCOPE_HERE>_marketing_event'"
                          display-field="'name'"
                          value-field="'sys_id'"
```

```
                          display-fields="'short_description'"
                          search-fields="'name'"
                          page-size="20"
                          id="mktg_event">
        </sn-record-picker>
      </div>

      <!-- First name -->
      <div class="form-group">
        <label for="exampleInputName2">First name</label>
        <input type="text" class="form-control" id="attendee_first"
placeholder="Joe" ng-model="c.data.attendeeFirstName">
      </div>

      <!-- Last name -->
      <div class="form-group">
        <label for="exampleInputName2">Last name</label>
        <input type="text" class="form-control" id="attendee_last"
placeholder="Employee" ng-model="c.data.attendeeLastName">
      </div>

      <!-- Email -->
      <div class="form-group">
        <label for="exampleInputEmail2">Email</label>
        <input type="email" class="form-control" id="attendee_email"
placeholder="joe.employee@example.com" ng-model="c.data.attendeeEmail">
      </div>

      <button type="submit" class="btn btn-default" ng-click="c.doIt()"
>Add</button>
    </form>

  </div>
</div>
```

6. Copy the script block below (also found in **Lab 11/Step6-ClientController.js.txt**) into the **Client Script** field. This is your Angular controller.

```
function($scope, spUtil) {
  /* widget controller */
  var c = this;

  /***
  Set up the Reference Field to Events.
  Bind field to c.data variables.
  ***/
  $scope.evt = {
    displayValue: c.data.eventName,
    value: c.data.eventId,
    name: 'evt'
  };

  /***
  Add selected Attendee to the chosen Event.
```

```
  ***/
  c.doIt = function() {
    /***
    Basic field validation. Make sure entire form is filled in.
    ***/
    if (!$scope.evt.value || !c.data.attendeeFirstName ||
!c.data.attendeeLastName || !c.data.attendeeEmail)
      spUtil.addErrorMessage('Please fill out all fields on the form above.');
    else {
      /***
      Package up the form values to pass to the Server Script function.
      ***/
      var input = {
        'event' : $scope.evt.value,
        'first_name' : c.data.attendeeFirstName,
        'last_name' : c.data.attendeeLastName,
        'email' : c.data.attendeeEmail
      };

      /***
      Call Server script passing input to the function
      ***/
      c.server.get(input).then(function(r){
        if (r.data.success) {
          spUtil.addTrivialMessage(c.data.attendeeFirstName + ' ' +
c.data.attendeeLastName + ' was successfully added to ' +
$scope.evt.displayValue + ".");
          c.data.attendeeFirstName = c.data.attendeeLastName =
c.data.attendeeEmail = '';
        } else {
          spUtil.addErrorMessage('There was a problem adding the attendee.
Please contact your administrator.');
                }
      })
    }
  }
}
```

7. Finally copy the code below (also found in **Lab 11/Step7-ServerScript.js.txt**) into the **Server Script field.**

```
(function() {
  /* populate the 'data' object */
  /* e.g., data.table = $sp.getValue('table'); */

  /***
  If a server call is performed, run this function.
  ***/
  if (input){
    var attendee = new GlideRecord('x_<YOUR_SCOPE_HERE>_attendee');
    attendee.initialize();
    attendee.marketing_event = input.event;
    attendee.first_name = input.first_name;
    attendee.last_name = input.last_name;
```

```
    attendee.email = input.email;
    data.success = attendee.insert();
    return;
  }
})();
```

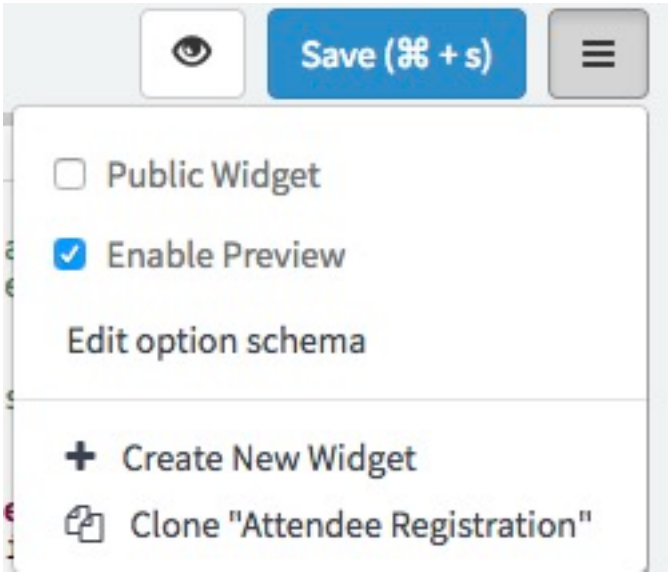
8. Hit **Save**.

What we have built is a simple Angular form with fields for first & last name, email and a reference field to the Marketing Event table. The client script performs some basic data validation (ensuring all fields are filled in) and then packages the data up to send to the server. The server code simply takes the values passed in by the client and uses GlideRecord to insert them into the table.

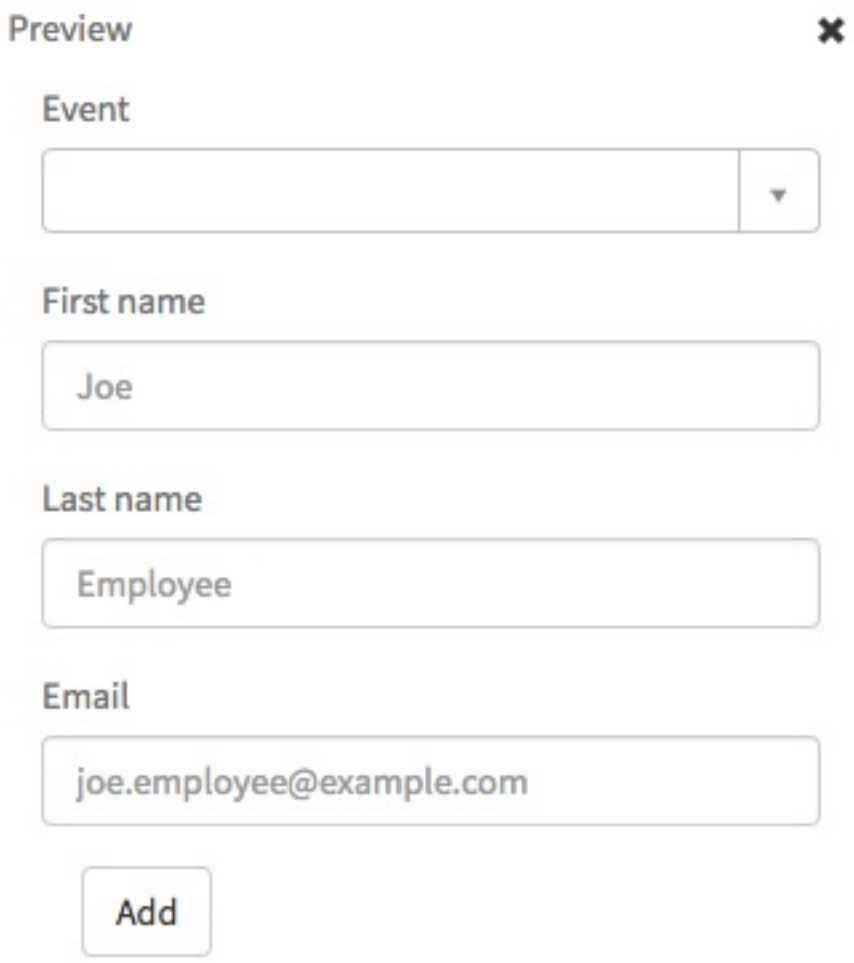
Testing the Widget

Widgets can be previewed from within the Widget Editor. This makes it very quick and easy to see how your widget will look as you're developing it.

1. Click the  icon in the upper right corner of the window and ensure **Enable Preview** is selected



2. Click the  icon to turn on the widget preview.

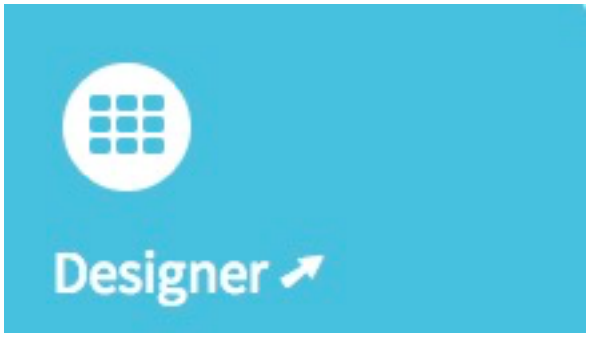


3. Make some changes to the HTML Template and hit **Save**. The preview will update to reflect your changes.

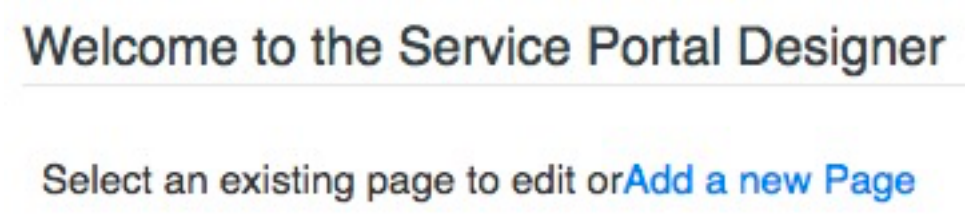
Create a new Attendee Page

1. In the upper left of the Widget Editor, click on the **ServiceNow** logo to go back to the Service Portal Configuration home.

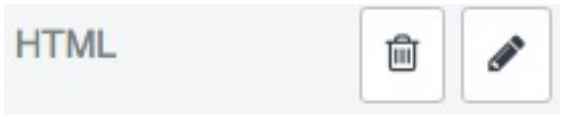
2. Click on **Designer** to open the **Service Portal Designer** window.



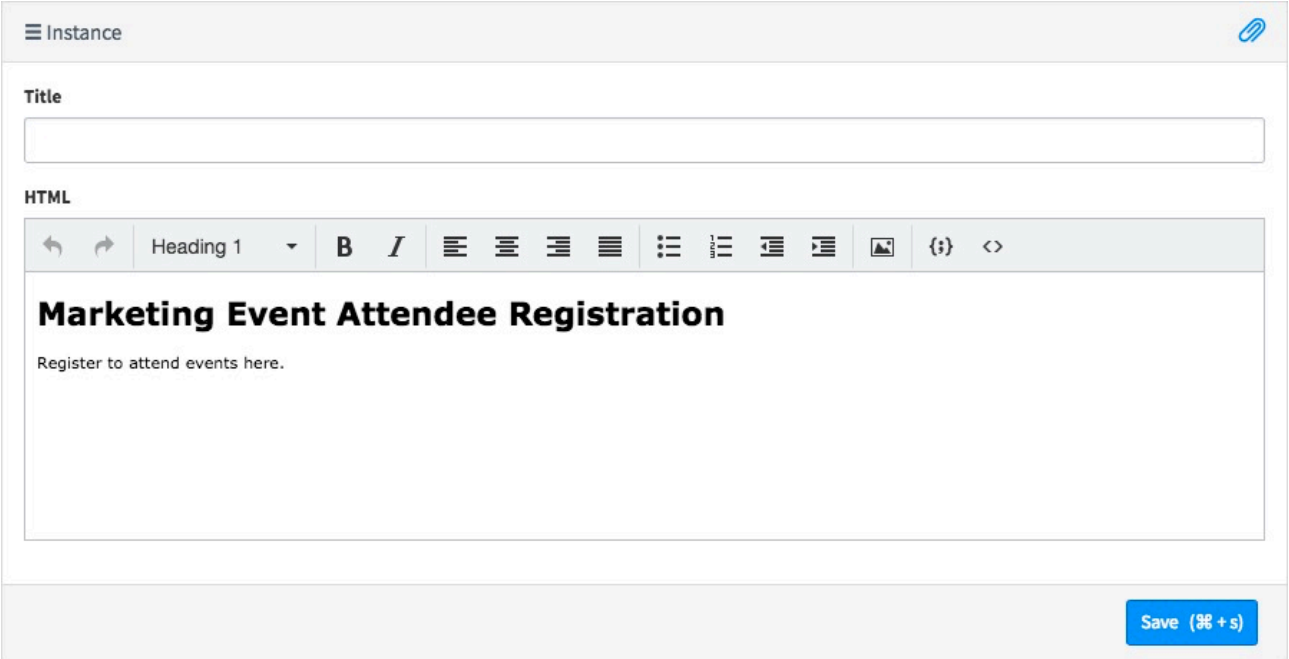
3. Click **Add a new Page**.



4. In the **Page Title** field, type **Attendee Registration** and click **Submit**.
5. A **Container** has already been created on the canvas.
6. Drag a **12** layout to the **Container**.
7. Now drag a **HTML** widget into the layout.
8. Click the  to edit the contents of the HTML widget.



9. Create a welcome message for attendees.



10. Click **Save**.

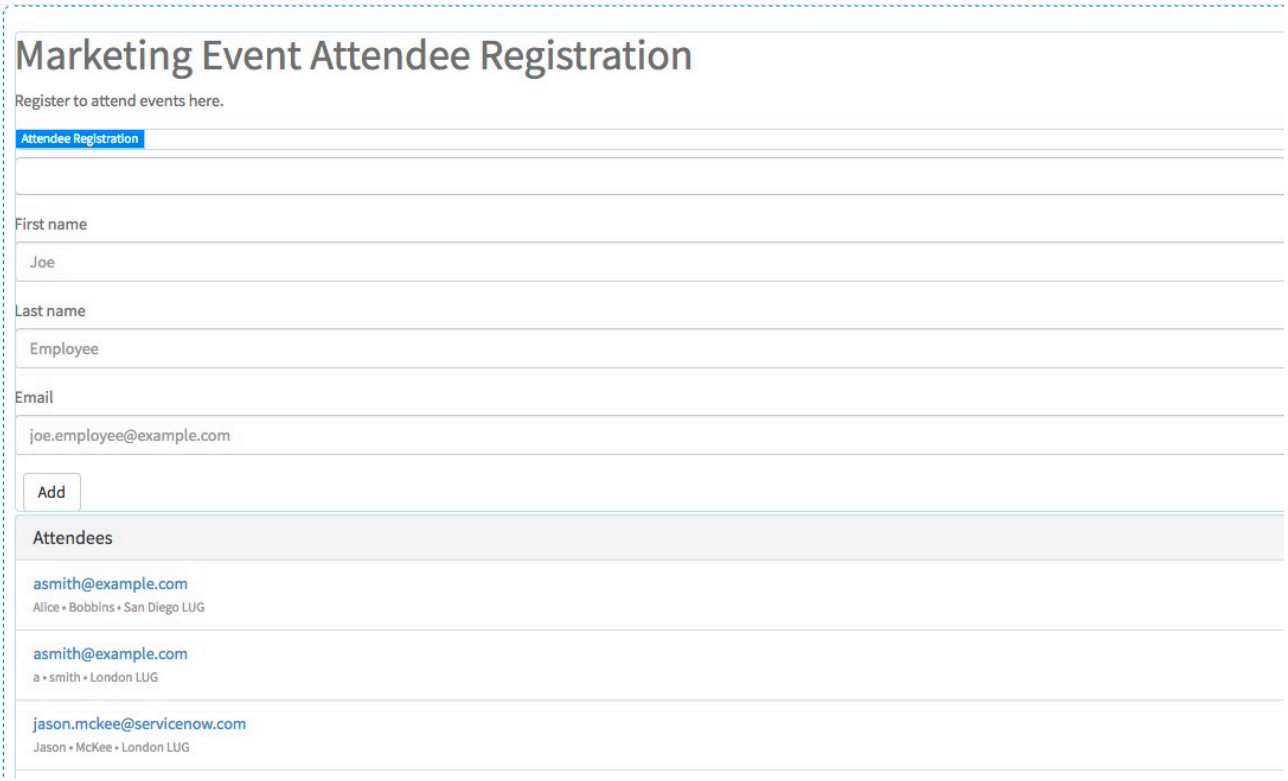
11. From the **Widgets** section, drag **Attendee Registration** to just below the **HTML** widget.

12. Similarly, drag a **Simple List** widget below that.

13. Click the  to edit the widget's properties and set the following values.

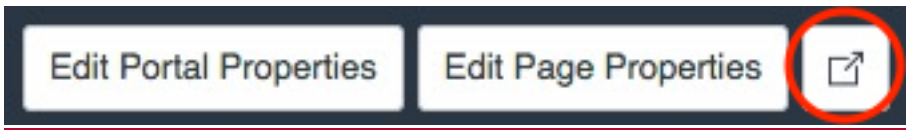
In this field…	Enter..
Table	Attendee
Display field	Email
Secondary fields	First name, Last name, Marketing event

14. Click **Save.** Your canvas should now look similar to this

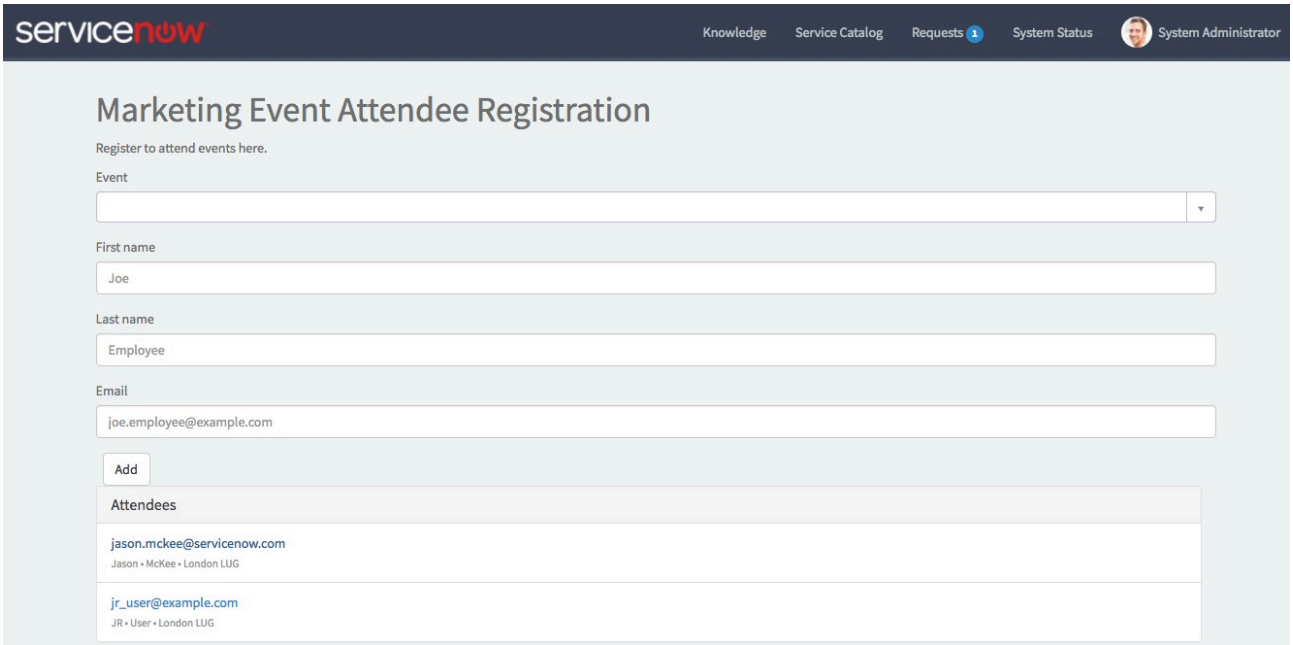


Test the Page

1. In the upper right corner of the **Page Designer**, click the small button next to **Edit Page Properties**.



2. The page will open up in a new widow



3. Try leaving a field blank and adding the registration, note that you'll get the error message defined in the widget's **Client Script**.

4. Now complete the form and click **Add** again. Note that the attendee is inserted into the table and the Simple List widget below updates automatically.

There's much more you can do with Service Portal such as adding the page you just built into a Portal or adjusting the theming and branding to match your organization, but hopefully this module has demonstrated how easy it is to build custom user interfaces for your applications.

Integrate with Other Systems

Lab 12 Integration

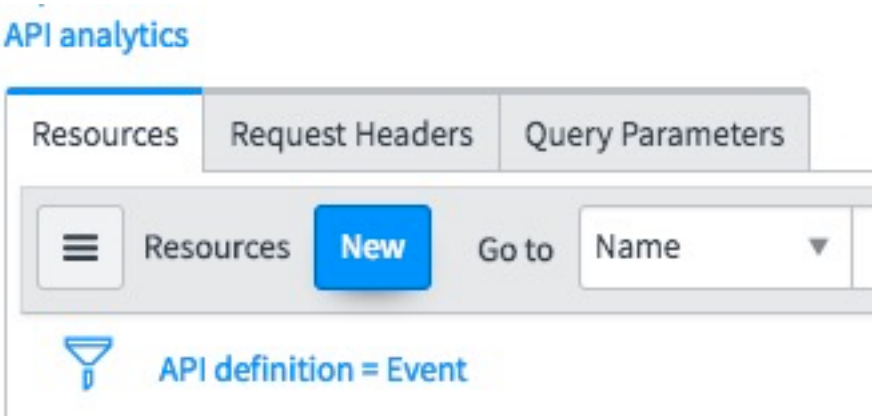
No application is an island. Most applications need to interact with other applications to either exchange data or participate in a business process. The ServiceNow Platform offers a rich set of integration features to solve common integration challenges—data import and export, real-time integration using REST and SOAP Web Services and prebuilt integrations to other systems.

Web Services provide standards based mechanism to connect systems together. The platform provides support for both SOAP and REST services. Both REST and SOAP based web services are automatically generated for all the tables. Sometimes however, a custom service is required and the platform supports that too. In this module, we will create a simple Scripted REST API to return Marketing Events and their equipment in a single "document" and then use the built-in REST API Explorer to test it out.

Create a custom REST service

1. Switch back to the **Studio** window
2. **Create New Application File** and select **Integrations > Scripted REST API**
3. For **Name** type **Events**, the **API ID** will fill in automatically

4. Hit **Submit**
5. Scroll to the Resources section at bottom of the **Scripted REST API** definition



6. Click **New** to create a resource

7. Set the following values.

In this field…	Enter..
Name	Get Event
Relative path	/{id}

8. Copy/Paste the following code (also found in **Lab 12/Step8-RESTScript.js.txt**) into the **Script** field

```
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {

  // query for a marketing event whose sys_id matches the id in the URI path
    var event = new GlideRecord('x_<YOUR_SCOPE_HERE>_marketing_event');
  if(event.get('sys_id',request.pathParams.id)) {
    // build a result object to return
    // be sure to convert GlideElements to Strings
    // using getValue()/getDisplayValue()
    var result = {
      name: event.getDisplayValue('name'),
      location: {
        name: event.getDisplayValue('location'),
        value: event.getValue('location')
      },
      startDate: event.getValue('start_date'),
      endDate: event.getValue('end_date'),
      equipment: []
    };

    // now add any equipment related to the event to the result object
    var equipment = new GlideRecord('x_<YOUR_SCOPE_HERE>_equipment_request');
    equipment.addQuery('marketing_event',event.getUniqueValue());
```

```
    equipment.query();
    while(equipment.next()) {
      var item = {
        type: equipment.getDisplayValue('type'),
        cost: equipment.getDisplayValue('type.cost')
      };
      result.equipment.push(item);
    }

    // return the result as an object, the platform will
    // convert the object to JSON or XML depending on
    // the headers in the request
    return result;
  }

  // if no matching id, return 404
  return new sn_ws_err.NotFoundError('No record matching id ' +
request.pathParams.id + ' found');

})(request, response);
```

9. Hit **Submit**

Explore the REST API

1. Switch back to the main application window and navigate to **System Web Services > REST API Explorer**

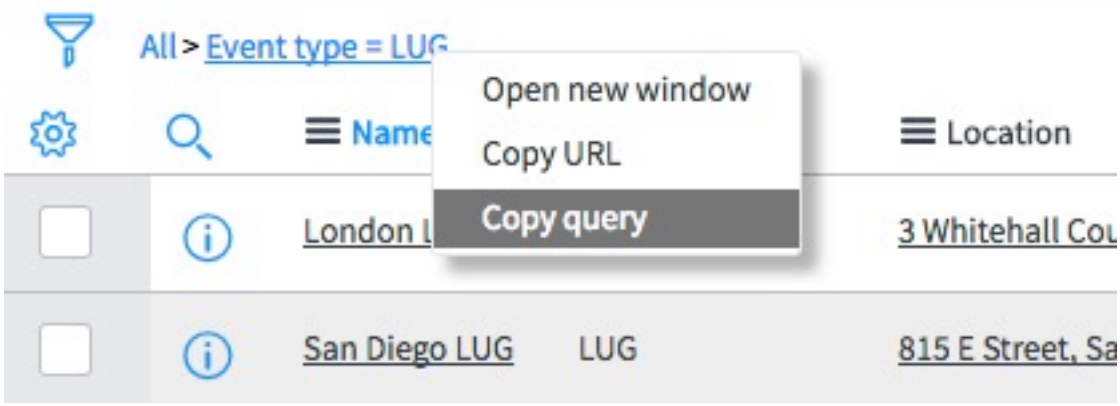
First we'll explore the Table API which is created automatically when you define your table. By default, the **REST API Explorer** opens to the Table API

2. For **tableName** select **Marketing Event**.

3. Scroll down and click 

.

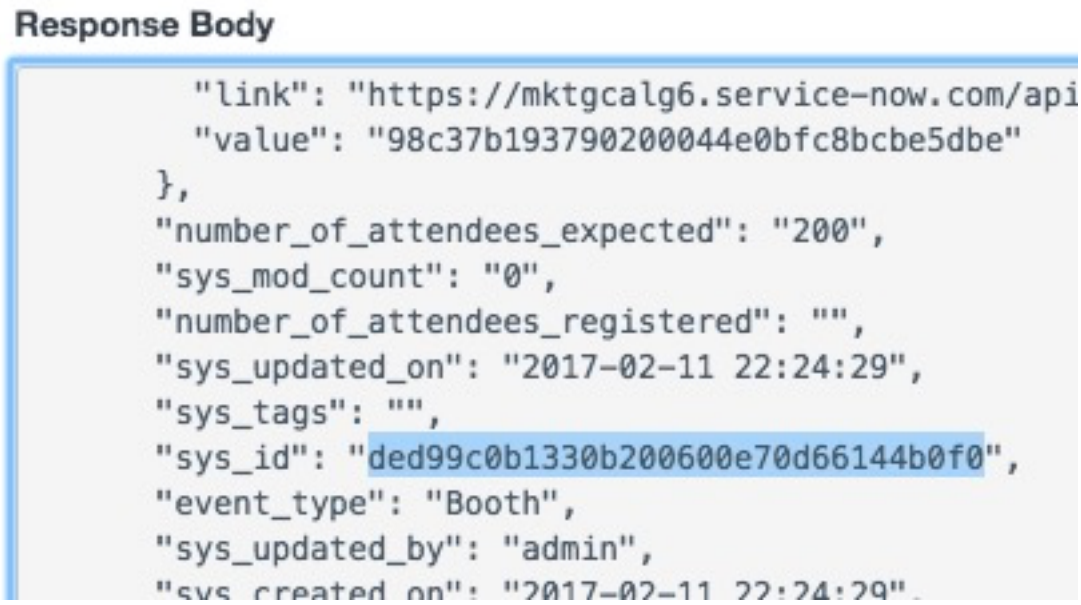
By default, you'll get a single record back, you can change this with the **sysparm_limit** query parameter. Similarly, you can limit which fields come back via **sysparm_fields**. Or which records are retrieved by pasting in an encoded query string into **sysparm_query**. Hint: you can generate an encoded query string by building a filter on a list view, right-clicking on the breadcrumb trail and selecting **Copy query**.



Test & Explore our Scripted REST API

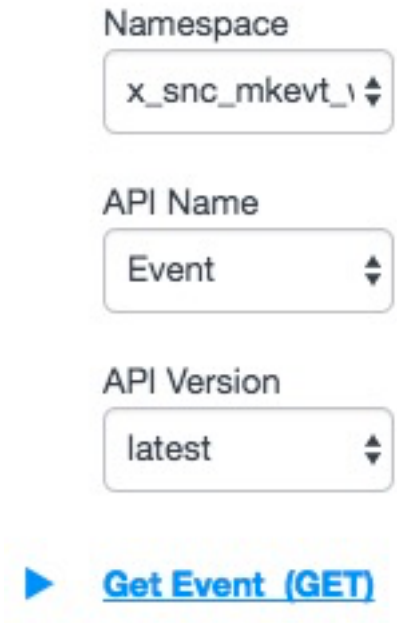
Now that we've played with the generated APIs, let's test the custom one we built earlier.

1. In the **Response Body**, find the **sys_id** for a Marketing Event record and copy it.



2. In the **Namespace** dropdown, select the one that matches your Marketing events app

3. We have only created one API and one resource so far, so **API Name** should already be set to **Event** and **Get Event (GET)** is the only option.



4. Paste the sys_id value you copied in the previous step into the id parameter field.

5. Scroll down and click **Send**. The **Response Body** should have our JSON object in it

Response Body


```
{
  "result": {
    "name": "London LUG",
    "location": {
      "name": "3 Whitehall Court, London",
      "value": "8225b668ac1d55eb679878e192fca315"
    },
    "startDate": "2017-06-01",
    "endDate": "2017-06-02",
    "equipment": [
      {
        "type": "Sony Projector",
        "cost": "$1,000.00"
      },
      {
        "type": "Sony Wireless Microphone",
        "cost": "$300.00"
```


Access Other API Definitions

To access the SOAP web service definition of a type, type the following URL into your browser (replace **<instance_name>** with the name of your instance and replace **<table_name>** with the name of the table you want to access:

> **https://<instance_name>.service-now.com/<table_name>.do?WSDL**
> For example, https://<instance_name>.service-now.com/incident.do?WSDL.

Similar to the **Scripted REST API**, there is also a **Scripted Web Service** capability that lets you build custom SOAP APIs.