

Lab Guide

CCW4229: Building Your First ServiceNow Application

Dave Slusher

Lab instance: <https://clabs.link/ccw4229>

Default Login / Password:

admin / Knowledge17

itil / Knowledge17

employee / Knowledge17

This
Page
Intentionally
Left
Blank

Lab Goal

The goal of this lab is to make sure you are set up properly to complete the subsequent labs.

Be sure that you:

- Have proper access to your assigned instance
- Can find and open Studio
- Create an empty application from Studio

Log in to Your Provided Instance

1. Navigate to the unique instance URL provided to you.
2. Log on with provided credentials.

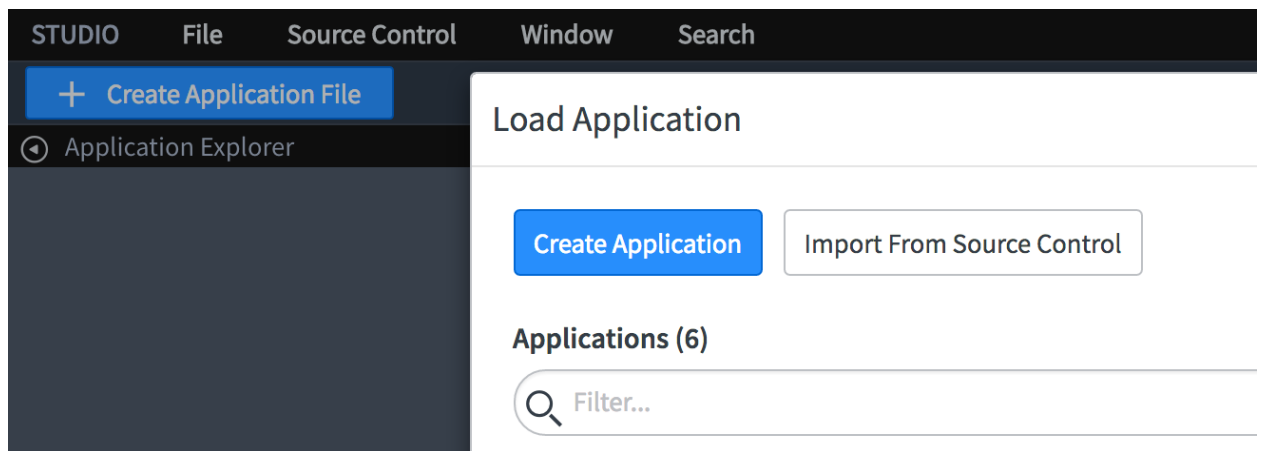
Lab 1 Log on to Instance

Lab Goal

Open Studio and Create an Application

1. On your instance, locate and find the Studio module and open it. Click **Create Application**.

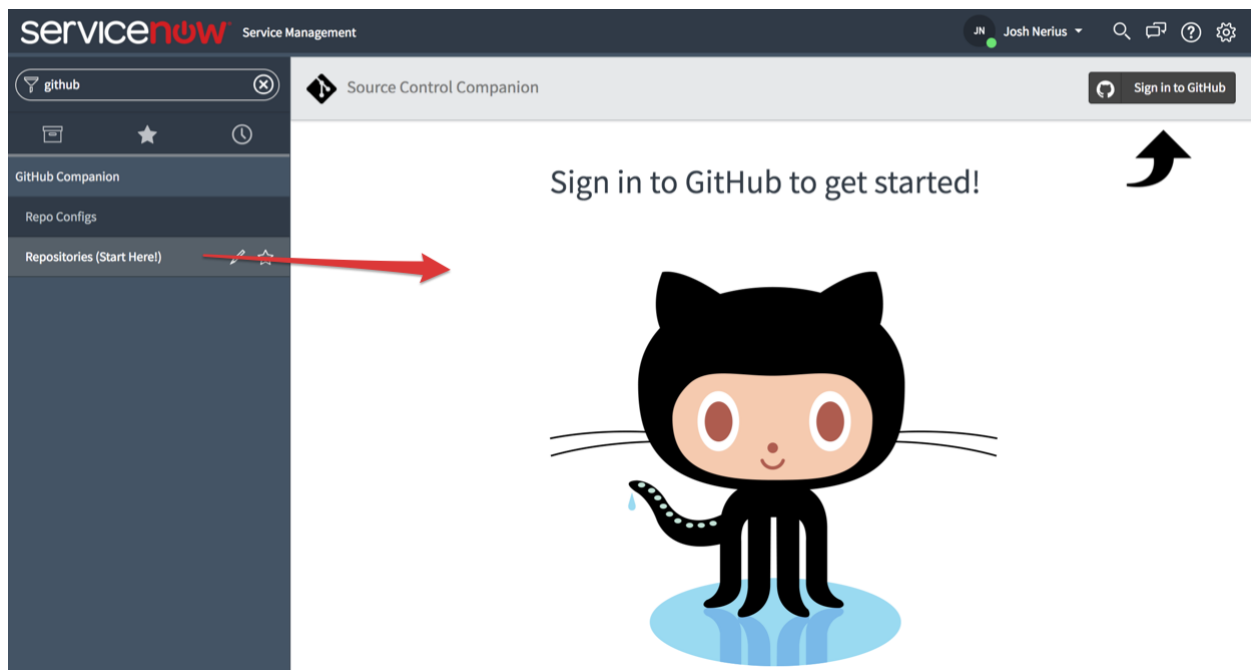
Lab 2 Create an Application



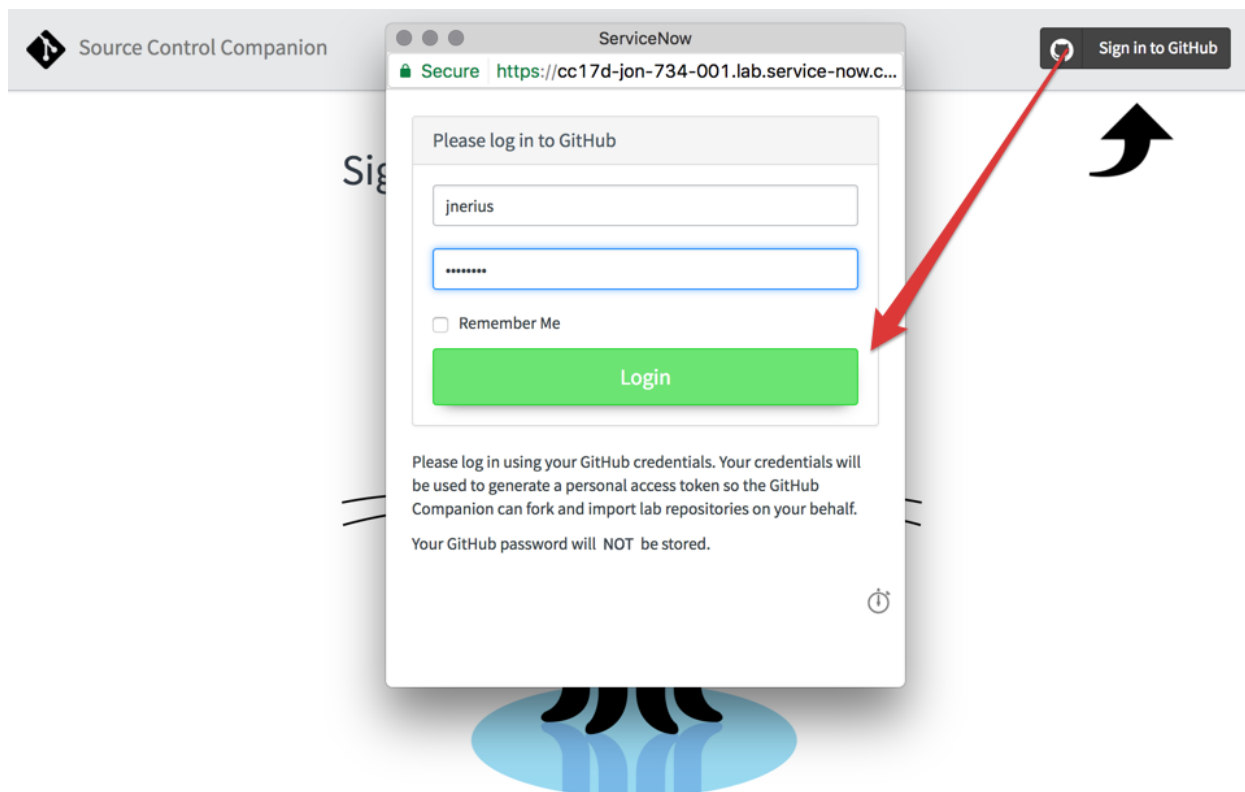
2. Click the "Start from scratch" button. This is to show you the steps, but we will not be keeping the one you create To emphasize this, name the application **Disposable app** and click the **Create** button.
3. These are the steps you follow to create an application. However, for the purpose of this lab we are going to connect to a GitHub repository to allow you to fast-forward to set points if you need to. Starting in the next lab we will connect to the Github repository.

Connect to Github Repository

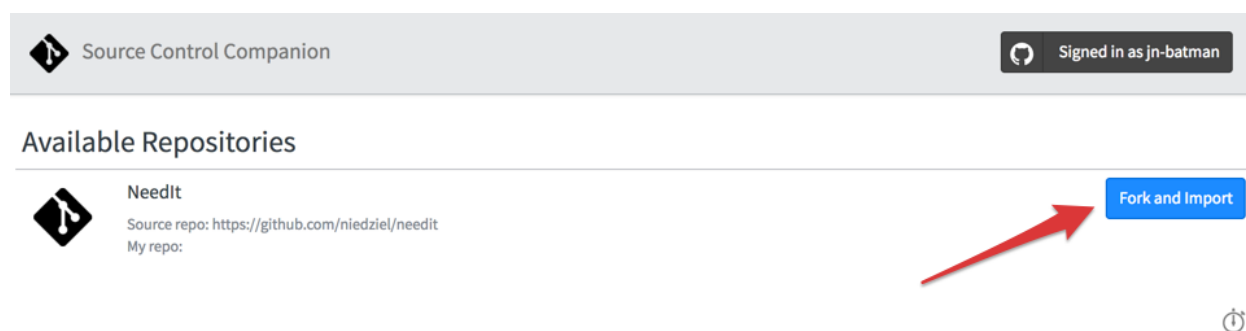
1. Click "Repositories (Start Here!)"



1. Click "Sign in to GitHub" and log in using GitHub credentials.



On successful login, you will be given a list of available repositories which in this case will only be the repo for this lab. Click “Fork and Import” next to the repository you wish to install.

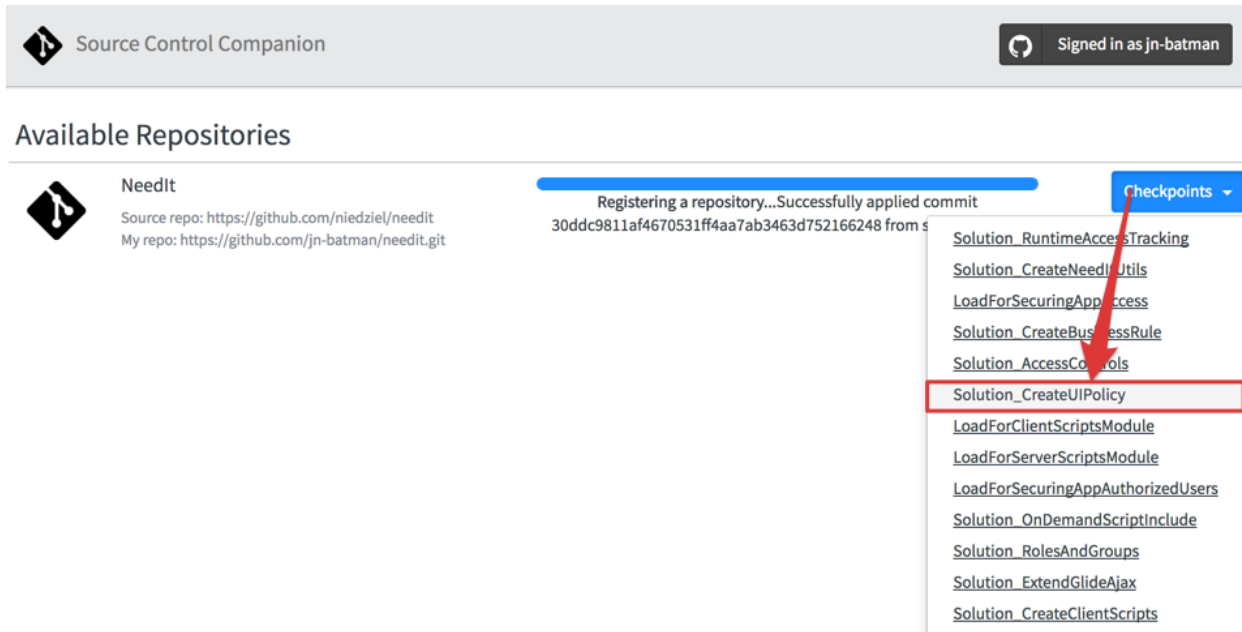


This will do two things:

1. Fork the repo in GitHub on the user’s behalf. From this point forward, all interactions will be with the user’s copy of the repo.

2. Import the repo into ServiceNow / Source Control from the user's fork. After importing, the app will be available in Studio.

After an application has been forked/imported, a "Checkpoints" button will be available. Clicking this button will list all Tags in the repository. The tags are named for the **beginning** of the lab. Thus if you are having trouble with **Lab 5** and want to proceed past it, you would choose the tag **Lab 6** to switch to.



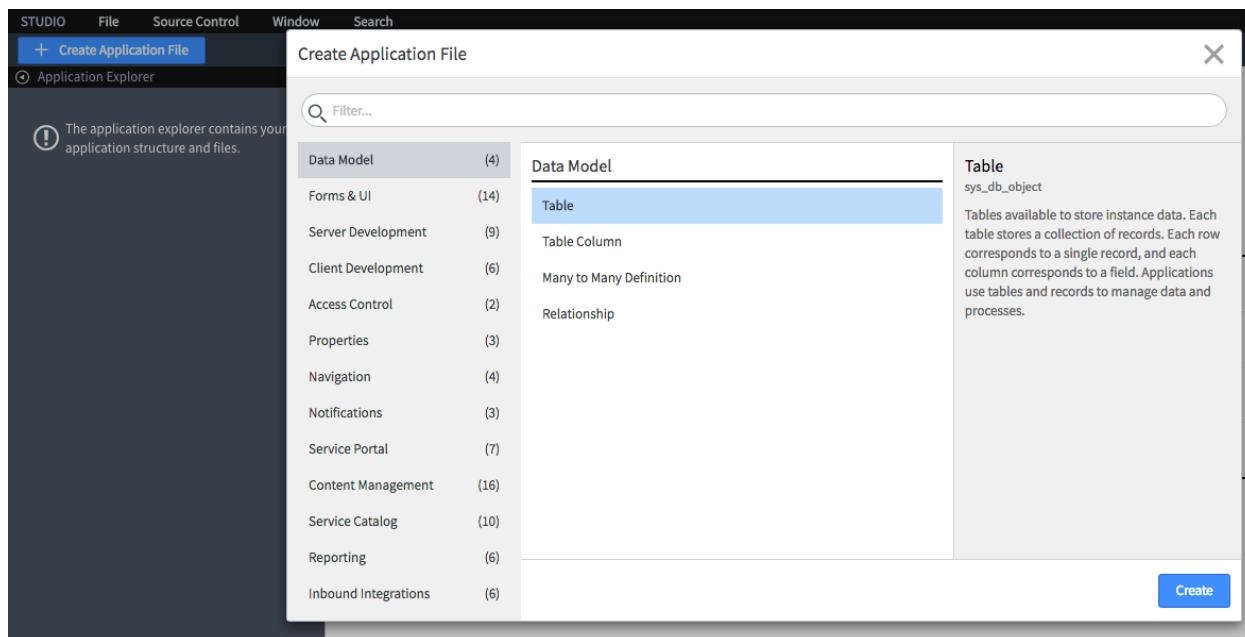
Lab Goal

Creating the Data Model

Now we will begin creating the data model. This is the core of beginning a new ServiceNow application project.

1. Open **Studio** and select the "Media Library" application.
2. Create the **Media** table. Click ***Create Application File**, then select **Table**. Click **Create**.

Lab 3
Create the
Data
Model



- Label: **Media**
 - Create module: **selected**
 - Create mobile module: **selected**
 - Add module to menu: **Create New**
 - New menu name: **Media Library**
3. You create three columns and a new Menu with this step. You can create all the columns at the same time and submit all at once. Add new columns by clicking "Insert a new row".
- Column label: **Title**
 - Type: **String**
 - Display: **True**
 - Column label: **Type**
 - Type: **Choice**
 - Column label: **Creator**
 - Type: **String**
4. Click **Submit**. Note that Access Controls are created by default.

Media
Table

Table
New record

A table is a collection of records in the database. Each record corresponds to a row in a table, and each field on a record corresponds to a column on that table. Applications use tables and records to manage data and processes. [More Info](#)

* Label: Application: ⓘ

* Name:

Extends table: 🔍

Create module: ☒

Create mobile module: ☒

Add module to menu: ▼

New menu name:

Columns Controls Application Access

Table Columns Search for text 🔍 Search

Dictionary Entries

	Column label	Type	Reference	Max length	Default value	Display
✕ ✎	Title	String				true
✕ ✎	Type	Choice				false
✕ ✎	Creator	String				false
+	Insert a new row...					

Submit Cancel

- Click the **Type** column you just created to set up some choices. Scroll down to see the ***Choice List Specification** and the **Choices** tab.

Choice: **Dropdown with – None --**

- Add two new rows under the **Choices** tab:

- Column Label and Value: **Book**
- Column Label and Value: **Movie**

8. Click **Submit**.
9. Click the **Format** column you just created to set up some choices. Scroll down to see the ***Choice List Specification** and the **Choices** tab.

Choice: **Dropdown with – None --**

10. Add four new rows under the **Choices** tab:

- Column Label and Value: **Hardcover**
- Column Label and Value: **Paperback**
- Column Label and Value: **Blu-Ray**
- Column Label and Value: **VHS**

11. Click **Update**.

12. Open the **Status** column from the **Copy Table** tab to set up some choices.

13. Create two new choices:

- Choice: **Dropdown without – None --**
- Column label and Value: **Available**
- Column label and Value: **Loaned**

14. Under the **Default Value** tab, add **Available** as the **Default Value**. Click **Update**.

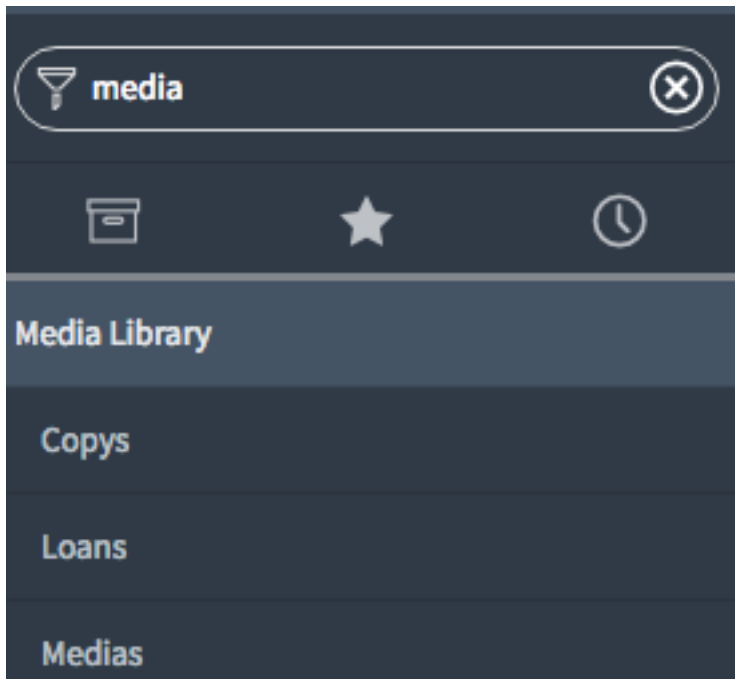
15. Create the **Loan** table and add the table as a module to the **Media Library** application. For **Extends table** select **Task**. Add the below columns:

- Column label: **Loaned item**
 - Type: **Reference**
 - Reference: **Copy**
 - Display: **True**
- Column label: **Loaned to**
 - Type: **Reference**
 - Reference: **Users (sys_user)**

16. Note that this table has many columns you did not create. These are from the **Task** base table. Click the **Opened by** column in the newly created table. It will give you a warning about being in the Global Application but this is expected. Scroll down and choose the **Labels** tab, then create a new entry. In it, add a **Label** value of "Loan time" then **Submit**.

Lab Success Verification

1. You should have three tables created in the Application Explorer, a "Media Library" Application in the Navigator and three Modules beneath it. Switch from the **Studio** tab to the main site tab and do a full page refresh. Type the word "Media" in the filter box at the upper left. You'll note the pluralization of the modules is weird. That will be fixed in a future step.



Lab Goal

The tables created when setting up the data model dynamically generate a form based on the columns set for it. These forms give users, who have access, a simple UI when interacting with the data model.

This lab explains how to get familiar with the Form Designer and create a default view for the tables just created.

Create a Default View for the Records

Each of the tables you created has related links that help design the default forms and list. We prefer not to use these related links as they steal focus away from the rest of Studio.

1. In **Studio** click the Create New Application File button and search for Form.

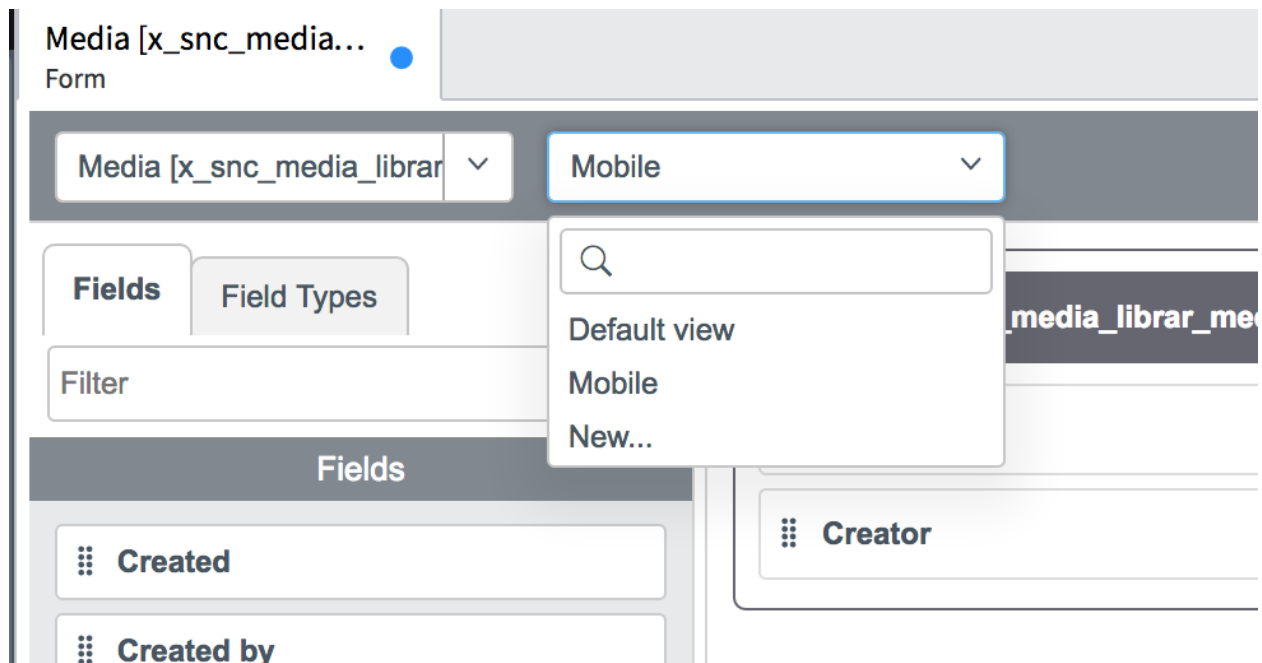
Lab 4 Create the Forms

2. Click Next and choose the table you wish to use the Form Designer on. Select "Media" the first time
3. Notice the many things you can do in the Form Designer. You can change the data model by adding more fields (columns) to the table. You can update and change some basic properties of the fields you have defined (like making a field mandatory). You can create different views, and change the layout of the form by adding new sections and columns to the form. Because **Studio** has a tabbed interface you can have the **Form Design** editor open for multiple Forms simultaneously.

4. Modify the "Media" tables you created in this fashion to have the layout you want. Drag and drop the fields until they have the order you prefer. Click **Save** when you are finished. (The blue dot in the tab indicates unsaved changes.)
5. Repeat the process for the "Copy" table.
6. Next modify the "Loan" table. You will note that because this table is extended from Task you will have a pre-populated form with a number of fields and sections. You will begin by removing those and selecting only the fields you want on that Form. For the "Loan" you will want the fields **Loaned to**, **Loan time**, **Loaned item**, **Due date** and **Active**. Make **Active** read-only by clicking the gear icon on that field in the **Designer**.
7. Remember that you need to click **Save** on each **Form Design** tab when you are completed.

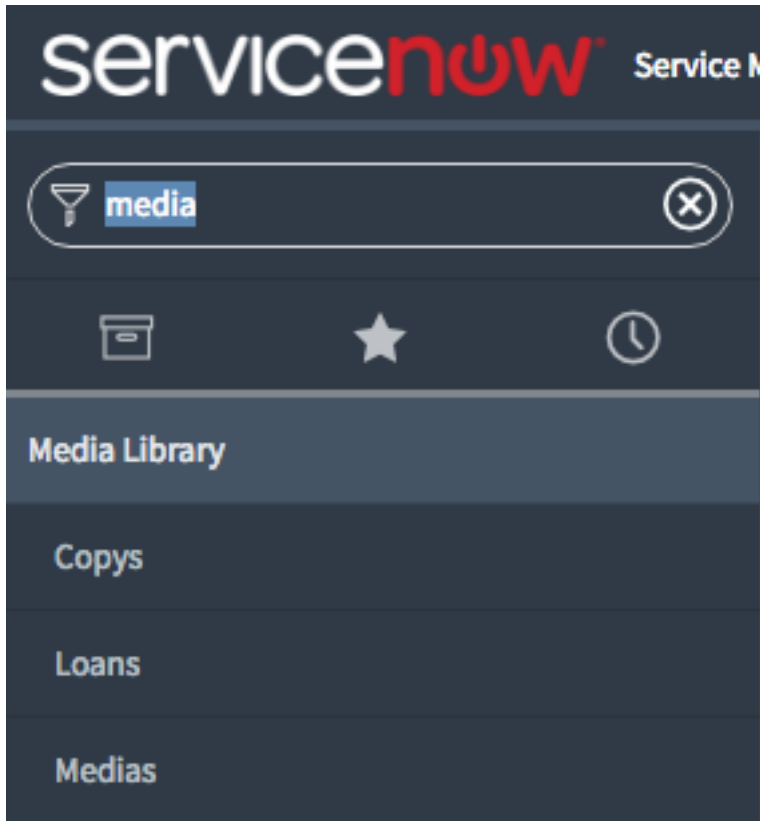
Creating the Mobile View

1. Open the form designer again for our Media table. Do you remember how to do that? Create a new Mobile view from Form Designer. Click the dropdown that says "Default view", and select "New". Name the new view "Mobile".
2. Change our main section to have 1 column to render instead of 2.
3. Click **Save**.



Lab Success Verification

You should be able to open the Form for each of the three Tables you have created and see the layout as you designed it. Do that by clicking the Modules for the Table in the Application Navigator. Create at least two records for each table. Begin with Media, then Copy, then Loan. You will need records created in that order for you to be able to assign references.



Lab Goal

Remember that a form represents a record in ServiceNow, but when you query for a list of records on a particular table you get a spreadsheet-looking grid view. This is called a list view. Often, especially with tables that have a lot of columns, you do not want every column on this view, just the most important fields from left to right.

This lab explains how to create a default list layout for your data model.

Formatting Instructions

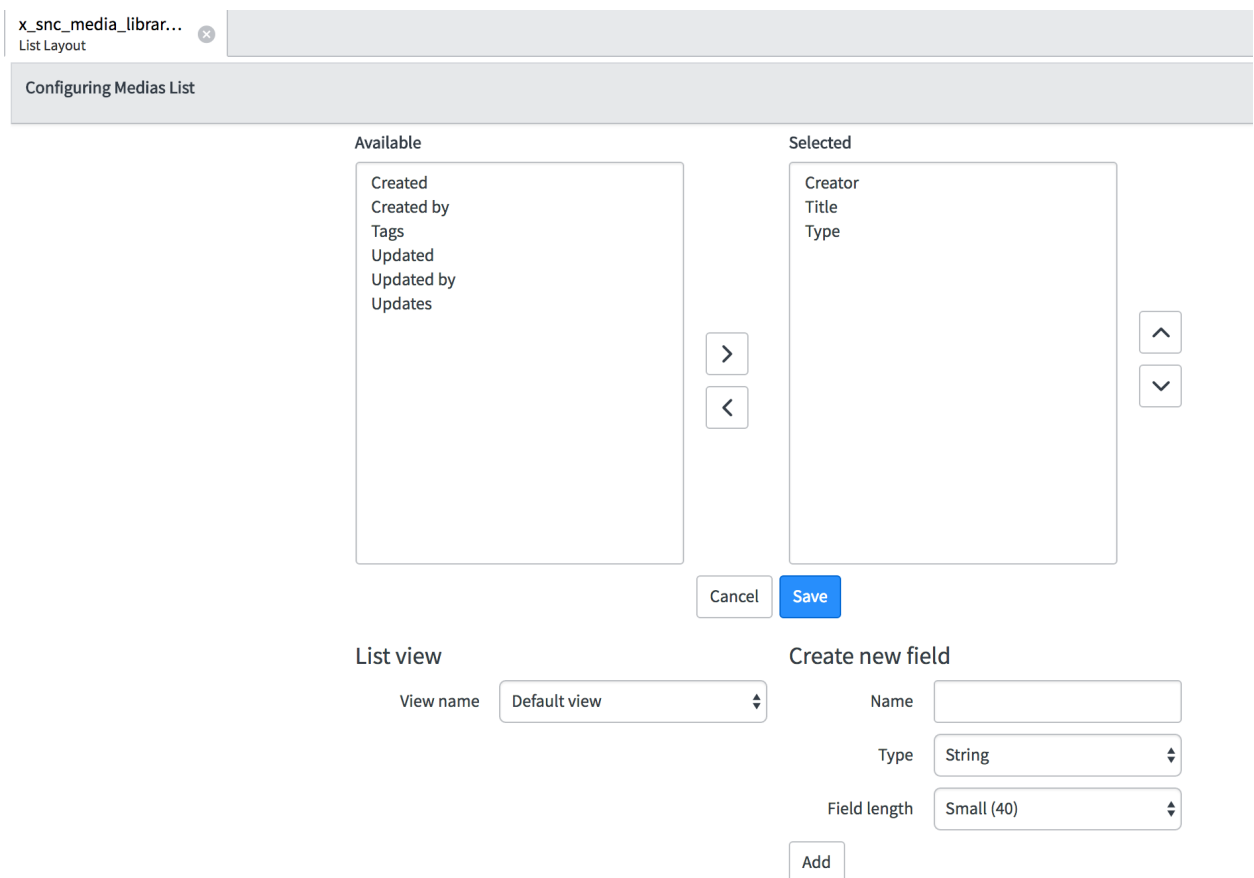
1. Click the **Create New Application File** button and search for List Layout. Click **Next**.
2. Choose "Media" and click Next.
3. There are many sections to this designer. You can select fields that your data model already contains. The left and right arrows will move fields between **Available** and

Lab 5 Create the Lists

Selected. The up and down arrows will change the order of the selected fields. The fields selected show up from left to right on the list based on the order you placed them in the selected slush bucket.

Note: You can create new data fields or a different view from this screen as well. We will not be doing that in this lab – just know you can.

4. Arrange the fields how you like and click **Save**.



x_snc_media_librar...
List Layout

Configuring Medias List

Available

- Created
- Created by
- Tags
- Updated
- Updated by
- Updates

Selected

- Creator
- Title
- Type

Cancel Save

List view

View name Default view

Create new field

Name

Type String

Field length Small (40)

Add

5. Repeat this exercise for the Copy table. Consider adding "Created" and "Updated" to the lists for each item. 1. For the **Copy** table, click in the "Media" field in the left-hand side of the slushbucket. Click the icon to expand the fields of the referred table. This allows you to choose Media fields to add to your Copy list.



6. Choose "Media.Title", double-click it or click the right arrow icon to add it to Selected. Move it to the top of the list. Remove the original "Media" item by double-clicking it or clicking the left arrow icon. Arrange the fields in an order you desire then click **Save**.
7. Repeat these steps for the **Loan** table. Note that as with the Form, the **Loan** list will have a number of columns from the **Task** table. Remove the unwanted ones, add in the fields you used on the Form (Loan time, Loaned To, Active, Due Date) as well as "Created" and "Updated". Use the above steps to add the fields from the "Loaned Item" (which is a reference to a Copy) to the "Media" table. Add "Title" to the list. Add the field "Format" from "Loaned Item" as well.

The screenshot shows a 'Configuring Loans List' window. At the top, there are three tabs labeled 'x_snc_media_librar...' and 'List Layout'. The main area is divided into two columns: 'Available' and 'Selected'. The 'Available' column contains a list of fields: 'Loan fields', '.Loaned item-->Copy fields', '..Media-->Media fields', 'Created', 'Created by', 'Creator', 'Tags', 'Title' (which is highlighted with a blue background), 'Type', 'Updated', 'Updated by', and 'Updates'. The 'Selected' column is currently empty, showing '--None--'. Between the two columns are two arrow buttons: a right-pointing arrow (>) and a left-pointing arrow (<). At the bottom right of the window are 'Cancel' and 'Save' buttons.

Lab Success Verification

1. You should be able to browse to the newly created menu for **Media Library** and open each module. The lists should open with the fields configured as you set them up. If you did not already populate some demo data, add some to see this working correctly.

Lab Goal

In this lab, we will adjust the Modules that were created along with the Tables above. This is to give the application a better look and feel.

1. In **Studio**, open the **Navigation -> Modules** record for "Medias". Update the name to "Media" and the **Order** to 100. Click **Update**.
2. Open the **Module** record for "Copys". Update the name to "Copies" and the **Order** to 200. Click **Update**.
3. Open the **Module** record for "Loans". Update the name to "Active Loans" and the **Order** to 300. Click the **Link Type** tab. Under **Filter** select "Active", "is", and "true". Update the record.
4. Repeat the above steps for the mobile Modules.

Lab 6 Adjust the Modules

Lab Success Verification

1. Switch to the main page UI. Reload the whole browser page and type "Media" into the filter box. Verify that you see the three titles you have updated. Click the "Active Loans" module and verify that it only shows active records.

Lab Goal

This lab explains how to use a UI policy to enforce the mandatory fields necessary for the tables. You could also accomplish this in the Form Designer, but in our efforts to give you more knowledge of the platform we do this a different way.

UI policies are useful for controlling the behavior of a form based on a specific condition. Another example – you may have a field you want visible only if it pertains to the state of another field. This keeps the form complexity low and prevents the form showing more than is necessary.

Create Mandatory Field on the Media and Copy Tables

1. Create a new **UI Policy** application file. This UI Policy enforces the mandatory fields you need on the Media table.
2. Select the **Media** table and give it a meaningful short description such as "Make Title Mandatory". Click **Submit**.

Lab 7 Create UI Policies

3. Notice your form changed after the save. Now you have the option to create a few policy rules. Create the **UI Policy Action** by clicking **New** in the lower section of the form.
4. Select the field for **Title** and for **Mandatory** select "True".

5. Click Submit.

Now you will create a second UI Policy to show how they can respond to changes in the Form.

6. Create a second **UI Policy** for the **Copy** table, with a Short description like "Make Format Read-only". Under **Conditions** choose "Status", "is" and "Loaned". Add a **UI Policy Action** to make the "Format" field read-only.

Lab Success Verification

1. Switch to the main page UI. Open the Media and Copy forms by either opening existing records or creating new records. Verify that the fields are correctly mandatory or read-only. Change the value of the Status box for a copy. Does the Format field change state?

Lab 8 Create Business Rules

Lab Goal

Add **Business Rules** to control integrity of your data.

1. In **Studio** click **Create Application File**. Select **Business Rule**.

Name: **Update Copy Status on Loan**

Table: **Loan**

Active: **checked**

Advanced: **checked** (This will give you access to do scripting.)

2. Under the **When to run** tab:
When: **after**
3. Under the **Advanced** tab, paste the below code in the **Script** box. Delete all the code that is there by default, replace entirely with this snippet. Note all the snippets are also available to paste from <http://bit.ly/CC17-FirstApp-Snippets>

```
(function executeRule(current, previous /*null when async*/) {  
  if (current.active.changes()) {  
    var copyID = current.getValue('loaned_item');  
    var copy = new GlideRecord('x_snc_media_librar_copy');  
    copy.get(copyID);  
    if (current.active) {  
      copy.setValue('status', 'Loaned');  
    } else {  
      copy.setValue('status', 'Available');  
    }  
    copy.update();  
  }  
}
```

```
})(current, previous);
```

4. Examine the code and walk through the logic. What do you think is the intention? Given this, under the **When to Run** tab, which choices of **Insert**, **Update**, **Delete** and **Query** do you think are appropriate to have checked? Check those, then **Submit** the record. **NOTE** if you have nothing checked, the **Business Rule** will never run.

Lab Success Verification

1. Test the business rule by creating Loan records that reference a Copy. After adding the Loan did the referenced Copy record change status?

Create an Abort Rule

1. Create a second **Business Rule** which will abort the creation of a "Loan" record if there is an existing active "Loan" on the same "Copy". Make sure that the Copy's State is "Available" first.
2. In **Studio** click **Create Application File**. Select **Business Rule**.

- Name: **Abort Duplicate Loan**
- Table: **Loan**
- Active: **checked**
- Advanced: **unchecked**

Under the **Actions** tab:

- Add message: **checked**
 - Message: **This item is already loaned**
 - Abort action: **checked**
3. You are going to use a dot-walked condition. Although this Business Rule is running against the Loan table, we want the condition to depend on a field of the referenced Copy item. The interface for doing this can be tricky so we will walk through in detail.
 4. Click the condition drop-down and scroll to the very bottom. Click "Show Related Fields."

When to run

Actions

Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which conditions the business rule should run.

Insert ☐

Update ☐

Filter Conditions

Add Filter Condition

Add "OR" Clause

-- choose field --

-- oper --

-- value --

Role conditions

Upon reject

Urgency

User input

Watch list

Work notes

Work notes list

loan_date

Show Related Fields

Submit

- At this point, all reference fields are now annotated with the Table they refer to. Select "Loaned Item ==> Copy fields".

When to run **Actions**

Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which conditions the business rule should run.

Insert ☐

Update ☐

Filter Conditions Add Filter Condition Add "OR" Clause

Show Related Fields ▼ -- oper -- -- value -- AND OR X

Role conditions

Submit

Loan time

Loaned item

Loaned item ⇒ Copy fields

Loaned to

Loaned to ⇒ User fields

Location

Location ⇒ Location fields

Made SLA

6. Now the drop-down is populated with the fields available on the Copy record. Select "Status" and set it to "is" and "Loaned."

When to run **Actions**

Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which conditions the business rule should run.

Insert ☐

Update ☐

Filter Conditions Add Filter Condition Add "OR" Clause

Loaned item ⇒ Copy fields ▼ -- oper -- -- value -- AND OR X

Role conditions

Submit

Media

Media ⇒ Media fields

Owned by

Owned by ⇒ User fields

Status

Sys ID

Updated

Updated by

7. Look at this setup. What do you think is the intention? Given this, under the **When to Run** tab, which choices of **Insert** and **Update** do you think are appropriate to have checked? Check those then **Submit** the record.

Lab Success Verification

1. Test the business rule by creating a Loan records against a Copy record that already has an active Loan. Was the record inserted?

Lab Goal

UI Actions are a way to add server side scripting to the user interface of a ServiceNow Instance. In this lab we will examine some of the functionality available via this scripting.

1. Create a new **UI Action** Application File.
 - Name: **Loan Copy**
 - Table: **Copy**
 - Action name: **Loan Copy**
 - Form button: **checked**
2. In the **Script** field use the following code, which will create a new loan record for the current copy. Remember that the **Business Rule** we created earlier will prevent you from creating a **Loan** record for a copy that is already loaned.

```
(function loanCopy(current) {  
  var copyID = current.sys_id;  
  var gr = new GlideRecord("x_snc_media_librar_loan");  
  gr.setValue('loaned_item', copyID);  
  gr.insert();  
  action.setRedirectURL(gr);  
})(current);
```

1. Create a second **UI Action**.
 - Name: **Return Item**
 - Table: **Loan**
 - Action name: **Return Item**
 - Form button: **checked**

Lab 9 Create UI Actions

- Show insert: **unchecked**
- Show update: **checked**

2. In the **Script** field paste the below code.

```
(function returnItem(current) {
  current.active = false;
  current.update();
})(current);
```

Lab Success Verification

1. Open the record for a Copy. Click the "Loan Copy" **UI Action**. Did it open a Loan form with the "Loaned item" field populated with your Copy record? If not, correct the **UI Action** until it does.
2. Open the record for an active Loan. Click the "Return Item" **UI Action**. Did it correctly mark the record as inactive? If not, correct the **UI Action** until it does.

Lab Goal

For this lab we will create three interconnected objects: A **Notification** to send an email, an **Event** to trigger the sending, and a **Scheduled Script Execution** to fire the event. We will create them in the order that we need them to exist.

Create the Event Registration

1. Create a new **Application** file for an **Event Registration**.
 - Suffix: **loan_due**
 - Table: **Loan**
 - Description: **Event to trigger Loan Due emails**

Create the Notification

1. **Important** - Click the related link for **Advanced View** at the bottom of the form.
 - Name: **Loan Due Back**
 - Table: **Loan**

Lab 10 Create Notifications

Under **When to send** tab

- Send when: **Event is fired**
- Event name: **x_snc_media_librar.loan_due**

Under **Who will receive** tab

- Users/Group in fields: **Loaned to**

Under **What it will contain** tab

- Subject: **\${loaned_item.media.title} is due soon**
- Message HTML:

```
<div>Your loaned item ${loaned_item.media.title} is due back on ${due_date}. In order to mark this  
item returned, please visit here: ${URI_REF}.  
</div>
```

2. Update

Create the Scheduled Script Execution to Generate the Event

1. Create a new **Application** file for a **Scheduled Script Execution**

- Name: **Check for Loans that are due within a day**
- Time: **1 00 00**
- Script: below code

```
(function checkForDueLoans() {  
  var gr = new GlideRecord('x_snc_media_librar_loan');  
  gr.addActiveQuery();  
  gr.addEncodedQuery('due_date<=javascript:gs.daysAgoEnd(-1)');  
  gr.query();  
  
  while (gr.next()) {  
    gs.eventQueue('x_snc_media_librar.loan_due', gr);  
  }  
})();
```

- ## 2. Examine this code. Do you understand the logic? Which records will trigger events? How many?

Lab Success Verification

1. Create a Loan record with a Due date of tomorrow. Make sure the **Loaned to** field is populated.
2. Open the "Check for Loans that are due within a day" **Scheduled Script Execution** tab in Studio. Click the **Execute Now** button.
3. Note: there will be a lag between clicking the button and seeing the **Notification** created. Open the **Outbox** module in the Application Navigator of the main UI window. You should see the Notification record for all Loan records with a due date of tomorrow or earlier. If not, look at the System Logs -> All to see if you see any errors. Your lab gurus can help you troubleshoot if necessary.
4. Bonus question: What happens if the due date is in the past? What can you do to improve this system to handle overdue loans?

Lab Goal

This lab explains how to create a related list to show up when the form is rendered. Related lists are tables of data that are related in some way (either by foreign key in the database or an explicit relationship you create) to the data / form you are rendering. They can be created automatically with default queries but these steps will show how to use a custom relationship.

1. Create a new **Application** file for **Relationship**.
 - Name: **Active loans**
 - Applies to table: **Copy**
 - Queries from table: **Loan**
 - Query with: below code

```
(function refineQuery(current, parent) {  
  var media = [];  
  var gp = new GlideRecord("x_snc_media_librar_copy");  
  gp.addQuery("media", parent.getValue("sys_id"));  
  
  gp.query();
```

Challenge Lab Custom Related Lists

```
while(gp.next())
    media.push(gp.getValue("media"));

current.addQuery("sys_id", "IN", media.join());
})(current, parent);
```

Create a new related list application file for the Copy table and move "Active Loans" to the **Selected** box.

2. Click Save.

Lab Success Verification

1. For a given Copy record, create and return multiple Loan records so that you have at least one inactive Loan and one active Loan. Open the Form for that Copy record and look at the Related List that displays. Does it contain only the active Loan record? Why or why not?