Utility Bar API Implementation Guide

Salesforce, Spring '24

@salesforcedocs

Last updated: November 9, 2023

CONTENTS

UTILITY BAR API IMPLEMENTATION GUIDE

You can use the Tooling API and Metadata API to create a utility bar and add it to a Lightning app.

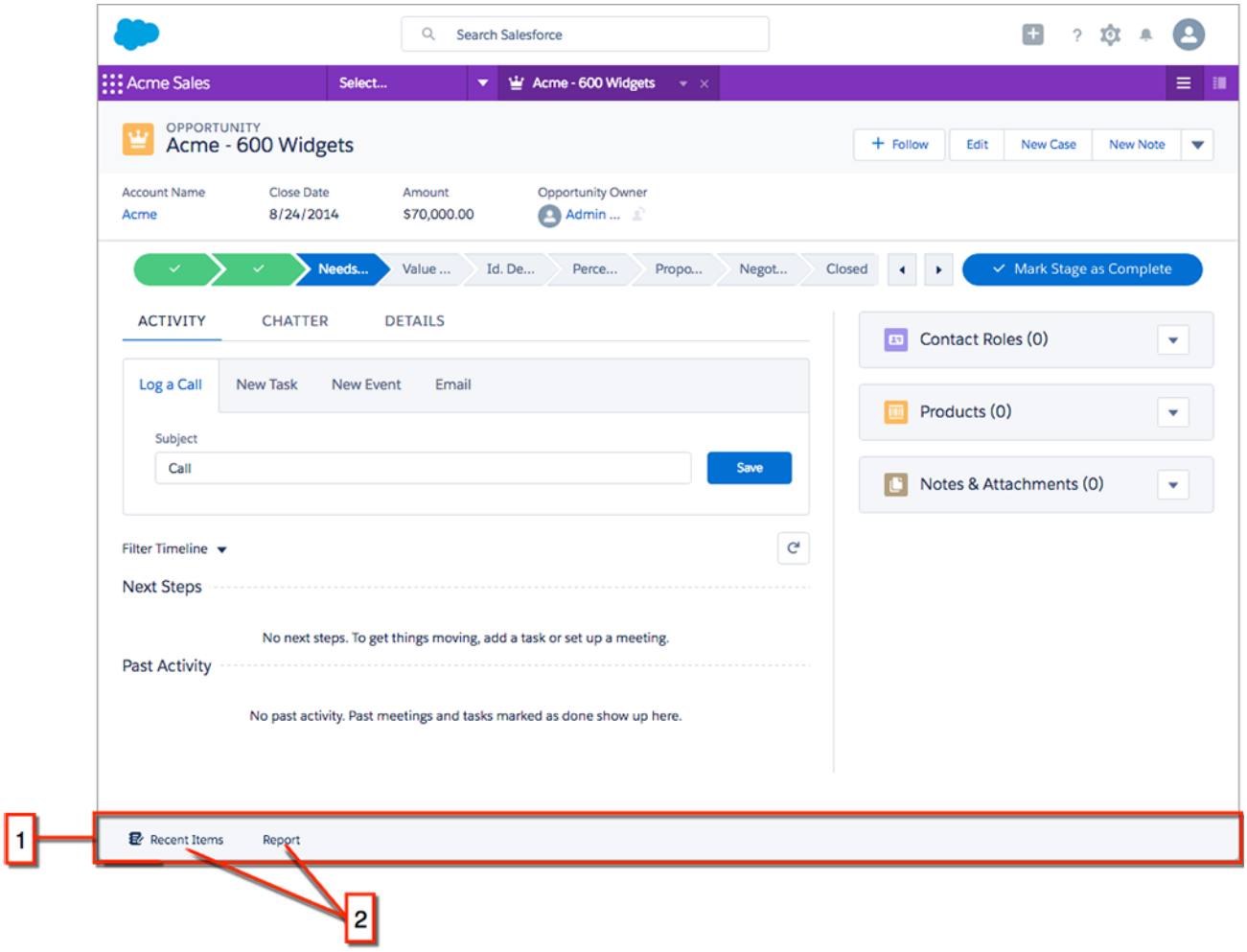
Tip: You can also set up and configure a utility bar using the Lightning App Wizard in the UI. For detailed information, see Add a Utility Bar to Lightning Apps. in the Salesforce help.

IN THIS SECTION:

What Is the Utility Bar?

Lightning Experience allows you to add your components to prime real estate in any Lightning app so that your users have one-click access to powerful productivity tools. Now, you can access those same productivity tools in a horizontal footer, called the utility bar.

The utility bar in Lightning Experience combines the best features from home page components in Salesforce Classic and the footer in Salesforce Classic console apps. The utility bar shows components in Lightning Experience so your users can easily access tools like Lightning Voice and Notes.



The utility bar is implemented as a Lightning page with `"type": "UtilityBar"` and `"pageTemplate": "one:utilityBarTemplateDesktop"`.

The utility bar Lightning page includes a single region (1), and, like any other Lightning page, contains Lightning components (2). In this case, these Lightning components are utilities. If this seems confusing right now, don't worry. We'll break it down as we go.

Note: A Lightning page region can contain up to 100 components.

What Is a Utility?

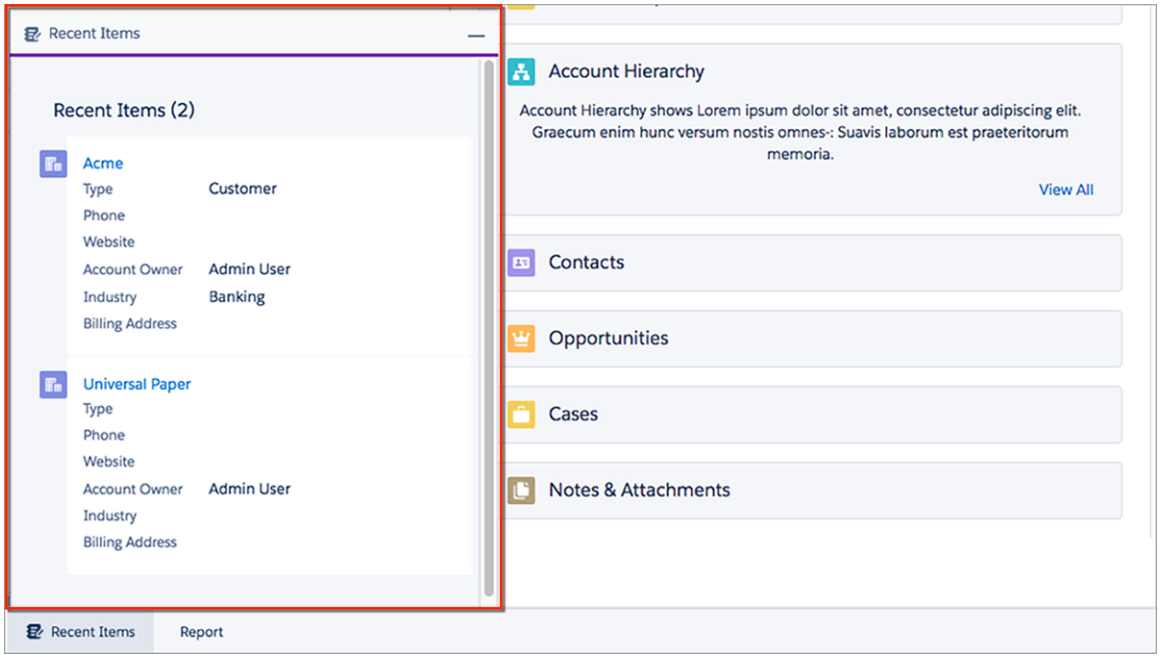
Utilities are Lightning components defined by a few characteristics. To be a utility, a Lightning component must implement `flexipage:availableForAllPageTypes`, be added to a utility bar FlexiPage, and have utility-specific attributes defined.

When you add a component to the utility bar, you can define attributes that determine things like the utility's height and width.

Other Lightning components that implement `flexipage:availableForAllPageTypes` have other attributes you must define. For example, when you create a report chart component you have to provide a `reportId`, and decide whether to show the refresh icon.

You can create a utility bar using custom Lightning components or out-of-the-box components. You can also easily adapt existing custom Lightning components as utilities by updating them to implement `flexipage:availableForAllPageTypes`.

This utility bar, built with out-of-the-box Lightning components, displays a report chart using `flexipage:reportChart` and a list of recently viewed items using `flexipage:recentItems`.



Defining a Utility Bar with JSON

Creating a utility bar is similar to creating any other Lightning page. Unlike other Lightning pages, however, you cannot create or edit utility bars in the Lightning App Builder.

A utility bar is a Lightning page of a specific `type` and `pageTemplate`.

```
"type": "UtilityBar"
"pageTemplate": "one:utilityBarTemplateDesktop"
```

In Lightning Experience, Lightning pages—also known as FlexiPages—are custom layouts that let you design pages. Typically, you create and edit Lightning pages in the Lightning App Builder.

Utility bar Lightning pages, however, aren't available in the Lightning App Builder. To create a utility bar, use the Tooling API to POST a JSON definition.

The REST endpoint for FlexiPage is `/services/data/v38.0/tooling/sobjects/FlexiPage/`. You can create a utility bar and define its components at the same time. For this example, we'll add a single Recent Items component, `flexipage:recentItems`, to show how the process works. This `recentItems` component implements `flexipage:availableForAllPageTypes`, which is a required interface for Lightning components added to a utility bar.

Utility Bar Syntax

The JSON definition of a utility bar is similar to that of any other Lightning page.

[other]: Where possible, we changed noninclusive terms to align with our company value of Equality. We maintained certain terms to avoid any effect on customer implementations.

Once you understand the basic structure of a JSON utility bar definition, it's easy to customize your own by adding out-of-the-box or custom Lightning components.

Each utility in any utility bar is defined by five attributes.

- **Height:** Optional. The component's height.
- **Width:** Optional. The component's width
- **Eager:** Optional. Whether the component loads when the app loads. If `false`, the component loads when a user first opens it.
- **Label:** Required. The component's label.
- **Icon:** Optional. The component's icon. Supported values for this attribute are listed on the Salesforce Lightning Design System website. Custom icons are not supported at this time.

This utility bar has one utility, `recentItems`, which displays a list of recently accessed records. You also have to define attributes specific to the `recentItems` FlexiPage:

- **ListItems:** Required. An array of sObject types representing which types of record appear in the list of recent items.
- **MaxRecords:** Optional. The maximum number of records displayed.

```
{
  "FullName": "recentItemsUtility",
  "Metadata": {
    "masterLabel": "Recent Items Utility Bar",
    "pageTemplate": "one:utilityBarTemplateDesktop",
    "type": "UtilityBar",
    "description": "This utility bar displays a list of recently accessed records.",
    "flexiPageRegions": [
      {
        "appendable": null,
        "name": "utilityItems",
        "type": "Region",
        "componentInstances": [
          {
```

```
            "componentName": "flexipage:recentItems",
            "componentInstanceProperties": [
              {
                "name": "entityNames",
                "value": "['Account']"
              },
              {
                "name": "maxRecords",
                "value": "5"
              },
              {
                "name": "icon",
                "type": "decorator",
                "value": "note"
              },
              {
                "name": "label",
                "type": "decorator",
                "value": "Recent Items Utility"
              },
              {
                "name": "width",
                "type": "decorator",
                "value": "425"
              },
              {
                "name": "height",
                "type": "decorator",
                "value": "320"
              }
            ]
          }
        ]
      }
    ]
  }
}
```

Let's look at this code line by line.

The first thing we define is a `FullName`, `recentItemsUtility`, which must be unique within your org. Next we define a `masterLabel` and `Description`, which are the user-facing label and description, respectively, for this utility bar.

To specify that this FlexiPage is utility bar, we set `"type": "UtilityBar"` and `"pageTemplate": "one:utilityBarTemplateDesktop"`.

The next chunk of code, starting with `"flexipageregions"`, defines a region within our FlexiPage. Utility bar FlexiPages only have one region, which can contain multiple components. In this case we're going to add one Lightning component, a `flexipage:recentItems` component, to this region. In general, you can add as many Lightning components as you want to a utility bar.

The last section of code, beginning with `"componentInstances"`, defines the attributes specific to the utility bar, as well as attributes specific to the `recentItems` Lightning component. Notice that we defined `"entityNames"` as an array that includes only the Account object and `"maxRecords"` as `5`, meaning that the utility will show a maximum of five recently viewed accounts.

Log In to Workbench

Use Workbench to make calls to the Tooling API and Metadata API.

You can't create utility bar Lightning pages in the Lightning App Builder. Instead, use a JSON POST request to the Tooling API. Workbench allows you to make API calls from a web browser with a simple and intuitive user interface.

1. Go to https://workbench.developerforce.com/login.php.
2. Choose your environment and API Version.
3. Click **Login with Salesforce** and enter your credentials.

Create a Utility Bar

Use the Tooling API in Workbench to create a utility bar FlexiPage.

[other]: Where possible, we changed noninclusive terms to align with our company value of Equality. We maintained certain terms to avoid any effect on customer implementations.

1. In Workbench, click **Utilities** > **REST Explorer**.
2. Enter `/services/data/v38.0/tooling/sobjects/FlexiPage/`.
3. Copy the JSON definition of the Recent Items Utility into the **Request Body**:

```
{
  "FullName": "recentItemsUtility",
  "Metadata": {
    "masterLabel": "Recent Items Utility Bar",
    "pageTemplate": "one:utilityBarTemplateDesktop",
    "type": "UtilityBar",
    "description": "This utility bar displays a list of recently accessed records.",
    "flexiPageRegions": [
      {
        "appendable": null,
        "name": "utilityItems",
        "type": "Region",
        "componentInstances": [
          {
            "componentName": "flexipage:recentItems",
            "componentInstanceProperties": [
              {
                "name": "entityNames",
                "value": "['Account']"
              },
              {
                "name": "maxRecords",
                "value": "5"
              },
              {
                "name": "icon",
                "type": "decorator",
                "value": "note"
              },
              {
```

```
                "name": "label",
                "type": "decorator",
                "value": "Recent Items Utility"
              },
              {
                "name": "width",
                "type": "decorator",
                "value": "425"
              },
              {
                "name": "height",
                "type": "decorator",
                "value": "320"
              }
            ]
          }
        ]
      }
    ]
  }
}
```

4. Click **Execute**.

If you see a **success: true** message, you've successfully added the utility bar to your org. Next, add it to a Lightning app.

Create a Lightning App

Create a Lightning app to add the utility bar to.

When you create a Lightning app, it's represented by a `customApplication` sObject in the Metadata API.

1. From the Home tab in Setup, enter *App* in the Quick Find box, then select **App Manager**.
2. Click **New Lightning App**, and walk through the New Lightning App wizard.

   You can give your app a name, set its primary color, give it a logo, customize which items appear in the app's navigation bar, and assign the app to user profiles. Name the app *My Lightning App*, with the developer name `My_Lightning_App`.

   Tip: The app description displays alongside the icon in the App Launcher. Make the description meaningful to your users.

Learn more about Lightning apps on Trailhead.

Now that you've created a Lightning app, use the Metadata API to add the `recentItemsUtility` to it.

Retrieve your Custom Applications Using the Metadata API

Use the Metadata API to retrieve information about your org's Lightning apps.

The Metadata API has two types of calls, deploy and retrieve. A retrieve call returns data from your org, while a deploy call updates or inserts data in your org.

Use a Metadata API retrieve call to get a list of custom applications in your org, and then add the Recent Items Utility Bar to a Lightning app.

1. In a text editor, copy and paste the following:

```
<?xml version='1.0' encoding='UTF-8'?>
<package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>CustomApplication</name>
    </types>
    <version>38.0</version>
</package>
```

2. Save the file as `package.xml`.

   The package file tells the Metadata API which objects you want to retrieve. This `package.xml` returns the XML definition of all the Classic apps and Lightning apps in your org.

3. In Workbench, click **Migration** > **Retrieve**.
4. Click **Choose File** and select `package.xml`.
5. Click **Next**, then click **Retrieve**.
6. Wait for the page to refresh and look for `success: true` in the output. If the call succeeded, click **Download ZIP File** to save the results.

Add Your Utility Bar to a Lightning App

Add your utility bar to a Lightning app by including the utility bar's `FullName` in the XML definition of the app.

1. Unzip the .zip file you downloaded. It should have a name similar to `retrieve_09SR00000000QjpMAE.zip`.
2. Navigate to the `applications` folder, and open `My_Lightning_App.app` in a text editor.
3. Copy and paste `<utilityBar>recentItemsUtility</utilityBar>` into the second-to-last line of code, right above `</CustomApplication>`. It should look something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomApplication xmlns="http://soap.sforce.com/2006/04/metadata">
    <brand>
        <headerColor>#FD30D8</headerColor>
    </brand>
    <description>My Lightning App</description>
    <formFactors>Large</formFactors>
    <label>A Lightning App with a utility bar.</label>
    <navType>Standard</navType>
    <tab>standard-Account</tab>
    <tab>standard-Case</tab>
    <tab>standard-Product2</tab>
    <tab>standard-Order</tab>
    <tab>standard-Opportunity</tab>
    <tab>standard-ContentNote</tab>
    <tab>standard-OtherUserProfile</tab>
    <tab>standard-Lead</tab>
    <tab>standard-AppLauncher</tab>
    <uiType>Lightning</uiType>
    <utilityBar>recentItemsUtility</utilityBar>
</CustomApplication>
```

This line of XML ties the utility bar you created earlier, recentItemsUtility, to the customApplication sObject you chose. This sObject represents a Lightning app in your org.

4. Save `My_Lightning_App.app`, and zip the `unpackaged` directory. The name of the file doesn't matter, so leave it as `unpackaged.zip`.

   Important: If you are on macOS, create the .zip file from Terminal. In Terminal, navigate to the directory of the folder you'd like to compress, and enter `zip -r unpackaged.zip nameOfFolder`, where `nameOfFolder` is the folder you're compressing.

   If you create a .zip file from Finder, macOS changes the file structure and your compressed package will be invalid.

Although you can add the same utility bar to multiple Lightning apps, we recommend assigning a utility bar to only one app. A utility bar assigned to multiple Lightning apps is shared. If you update a utility bar assigned to multiple apps, it changes across those apps.

Deploy Changes to Your Org

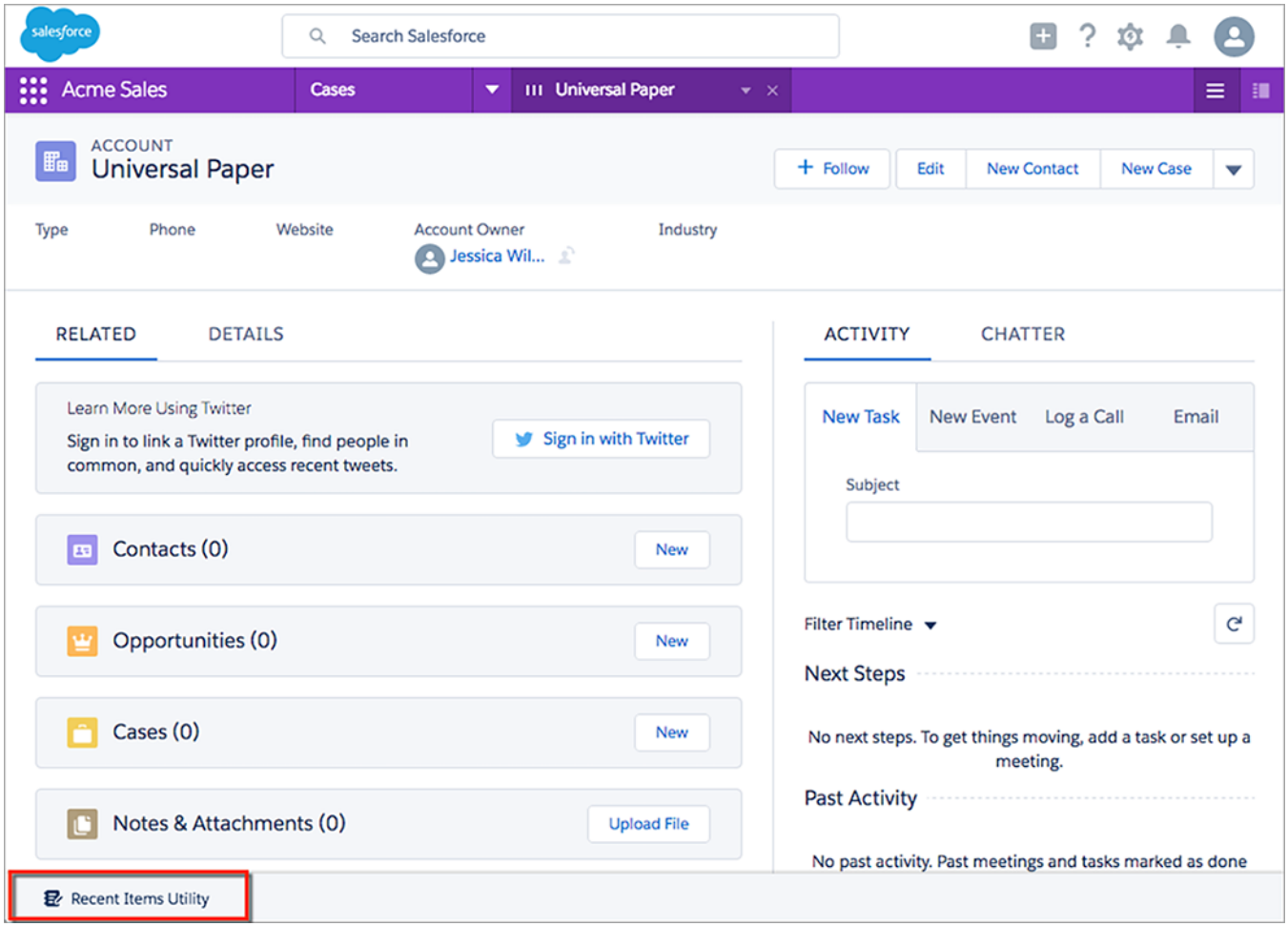
Use the Metadata API to deploy your updated package to your org.

1. In Workbench, click **Migration** > **Deploy**.
2. Click **Choose File** and choose `unpackaged.zip`, the .zip file you created.
3. Select **Rollback on Error** and choose an option from the **Test Level** dropdown.
4. Click **Next**, then click **Deploy**.
5. Wait for the page to refresh—this might take a few seconds—and look for **success: true**.

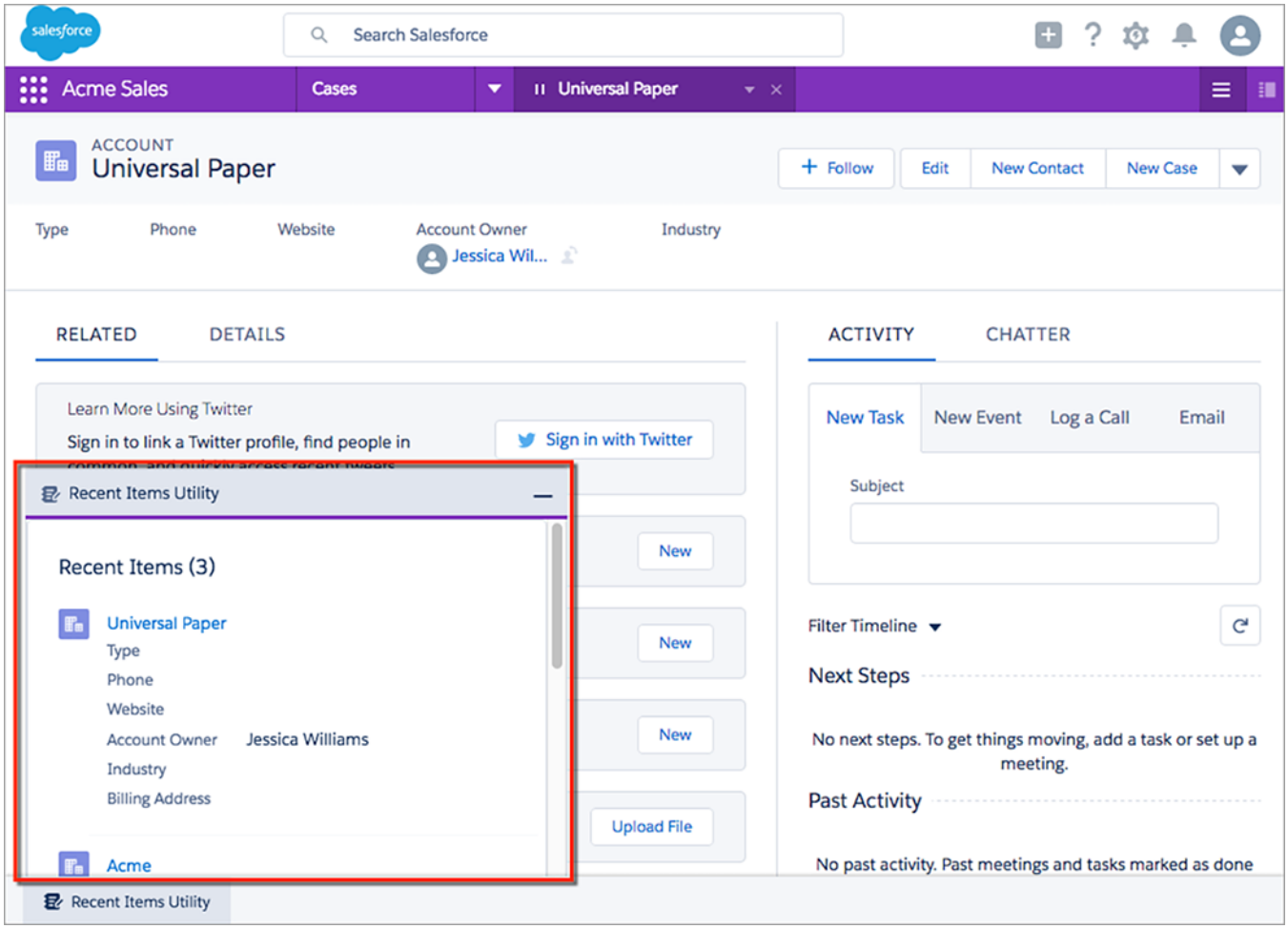
And that's it! Now you can log in to your org to check it out.

View the Recent Items Utility Bar in Lightning Experience

In Lightning Experience, click the App Launcher and select the Lightning app you created earlier. Check that the utility bar appears at the bottom of the app, and open the Recent Items Utility.



To ensure that the list of recently accessed items is populated, open a few Account record detail pages. When you open the utility, you see a list of recently viewed Accounts.



To see more utilities in action, install the Discount Calculator package. To install the package, log in to Salesforce, then open a separate tab or window and navigate to https://login.salesforce.com/packaging/installPackage.apexp?p0=04tB0000000HDKU.

This package includes a custom utility bar with several utilities built from out-of-the-box Lightning components, and one, Discount Calculator, built with a custom Lightning component.