| Click-Through Pages | Topics | Dialog | Notes |
| --- | --- | --- | --- |
| index.html | Overview | Don't know how to code? Don't want to code? No worries! In just a few minutes, you can build a mobile app that runs on either Android or iOS devices. No coding required!  So let's get started. Say you work for a company called Fix-it-Fast, and you need to create a mobile app for appliance repair technicians who work in the field. You want a dashboard that shows a technician the number of open incident reports, by priority, along with a graph that measures the technician’s productivity.  You also want the app to support the technician's workflow: for this, it needs a screen that lists all of the open incident reports. By clicking an item in this list, the technician can drill down to a detail screen to see some specifics about an incident or customer contact information. She can then update an incident’s details, or create a new incident report.  Creating all of these screens might sound like a daunting task, something that might take you hours or even days. But, as you’ll see, you can get a mobile app up and running in minutes using MAX. |  |
| 01.html-03.html | Picking the style for the first screen. | Now that we've logged in, we can start off by giving our app a name.  With that out of the way, we can jump into designing the app by choosing the format for the first screen that displays when a technician logs in. MAX gives you a set of basic styles that are typically used in mobile apps. For our app, we're picking the **Screen with Top Tabs**. We’ll use one tab for the dashboard and another for the list of incident reports. |  |
| 04.html | First page template | Next we’ll set the basic layout for our first page, the dashboard. We could pick **Custom**, if we wanted to create our own layout, but since we're making the first page a dashboard, we'll pick that template instead. |  |
| 05.html |  | And with that, we've laid the foundation of our app! Now we just need to add some data and finish up our page using the Designer. |  |
| 06.html-07.html | Intro to Designer/Adding tiles and Metric components | The Designer's toolbar on the far left lets you configure and create screens, browse the services that provide data for your app, and also dress up your screens with components for displaying data and text. And that's where we'll begin.  To help you get started with the dashboard, MAX displays the dashboard page in the preview.  The Component Palette on the left shows the user interface widgets that you can drag into the preview.  A dashboard is laid out using tiles. MAX already has one ready for you to configure in the preview. | There’s a scroll in the Component Palette in 07, which hides the Metric component that you want to use. |
| 08.html |  | Click the tile in the preview to take a look at the layout options in the Property Inspector on the right. |  |
| 09.html | PI for Tile | We want to create counters for high-, medium-, and low-priority incidents, so we’ll use the Property Inspector to set both the tile's dimension and adjust the layout to allow these counters to display properly. We're also going to add a title ("Metrics") and center it over the section. |  |
| 09.html-13.html | PI for the Metric component | We have our tiles configured, so let's pick a component that best suits numeric displays. For this, we'll drag **Metric** from the Component Palette into each of the tiles. Did you notice that the Property Inspector reflects the active component? Let’s use the Property Inspector now to format each Metric component by adding labels and colors. |  |
| 14.html | PI for line graph tile | Let’s add a new tile to the dashboard, so we can drag over a line graph that shows how the mobile repair technician’s performance stacks up against her peers.  Click **New Tile** in the Property Inspector to add a new tile. Like before, we’ll configure the tile's dimension and position within the dashboard. |  |
| 15.html | PI for Line Chart component | From the Component Palette, drag the **Line Chart** component into the preview.  Taking a look at the Property Inspector, you can see that MAX gives you different display options for the graph. If you want to show data as patterns that occur in cycles, for instance, you'd choose the radar (or web) graph display. But to display our data (which will get to in minute), we’ll stick with the standard X-Y line graph. |  |
| 16.html |  | Notice that MAX also gives you options for orienting the graph’s legend. | Are we going show legends and titles? |
|  | Adding List component/formatting the List tab | We've roughed in our dashboard, so now let's create the list screen for the incident reports assigned to the repair technician. Click the **List** tab in the preview. Next, drag in the **List** component from the Component Palette.  Your dashboard and list tabs are lookin' sweet! But if you ran this app on your phone right now, you wouldn’t be able to do much with it (other than admire it, that is). To make your app come alive (and useful), you have to wire up the user interface that you've just created to some data.  Ideally, you've worked with developers to describe the data needed by your app and how your app will interact with that data. To support your app, developers create a set of business objects. These are models of data which are portrayed as recognizable objects. For our app, we have business objects like Incident, Technician, and Customer. |  |
| 18.html | Data binding the Meter components | Let's start by adding the high, medium, and low incident counts to the dashboard. With the high-priority counter in the leftmost tile selected, we’ll open the Data tab in the Property Inspector. From here, we'll open the Service Catalog and then pick a service that contains the business objects that give us the data we need. |  |
| 19.html |  | From the Service Catalog, we pick the service created for our app, FixItFast. | Is it worth mentioning that in some cases (like the one here), you might use a service created expressly for your app, but in others, you might use one intended for more general use? |
| 20.html-21.html |  | To show a tally of the incidents by severity, our development team created a business object called **Incident Statistics.** |  |
| 22.html |  | To display the number of high-priority incidents in the dashboard counter, we drag the property called **high** into the **MetricValue** field. |  |
| 23.html-24.html |  | To return the number of incidents associated with a particular technician, we add the technician username as a parameter. A parameter acts as a filter on the data that's returned to the app, so only the data you care about is shown. |  |
| 24.html-25.html | Data binding the Line Chart component | After we repeat these steps to display data for the medium- and low-priority counters, we're ready to back the line chart with data by picking the business object called **Technician Performance**. |  |
| 26.html-27.html |  | The Technician Performance business object has properties that you use to map data to the Y axis (**IncidentCount**), the X axis (**month**), and identifying colors (**technician**) for the data series.  You can see how your mapping is going by keeping an eye on the Live Data Preview. |  |
| 28.html | Data binding the List component/Add Data wizard | Now that we've wired our visual components to data on the Dashboard tab, let’s move on to the Incidents tab and hook up the fields there to the data. To help us, we'll use the Quickstarts. The MAX Quickstarts not only prompt you along, but also anticipate your next step: they’re listed in roughly the order you’ll need them. |  |
| 29.html |  | Clicking **Add Data** opens the first step of the wizard. With the Fix-it-Fast service selected, select the business object that has the right fields to display incident-related data. Can you guess which one? That’s right -- it's **Incident**! | Is FIF already selected? |
| 30.html |  | Pick the type of data that you want the fields and other components in the List tab to display. Again, you can see if you're dragging and dropping the right properties by checking out the Live Data Preview. | I don’t know if this makes any difference, but we’ve also used the priority field for the icon. |
| 31.html |  | Because the incident assignments are specific to a field technician, you need to filter the data by the technician’s username. |  |
| 32.html-33.html | Add Detail wizard | With the overview done, MAX suggests that you create a detail screen so that your mobile technician can drill down from the List screen. Sounds like a good idea! |  |
| 34.html |  | Your first step is to pick the basic screen type for the Details screen. Once again, we pick the **Screen with Top** **Tabs**. We'll name the tabs **Summary** and **Customer**. |  |
| 35.html-36.html |  | We'll pick the **Form** template to display read/write information. | Show clicking **Go to Detail Screen**? |
| 37.html-38.html |  | Now we need to add some Incident data to the form. Let’s use the Add Data Quickstart again to help us. |  |
| 39.html |  | First, select the **Incident** business object from the FixitFast Service. | Not Incident Activities? |
| 40.html |  | One of the fields that we want in the Details pages is a long description of an incident. We’ll use the description property to create this field and its label.  Then add the fields that provide more information about an item in the List page. | Could you select Incident Activities and the select Add all fields? |
| 41.html |  | Like before, I can pass parameters from the client app but in this case I don’t need to do anything and the parameter that ties the screens together is already pre-filled for me |  |
| 42.html |  | Notice that the Details page lets users navigate back to the List screen. |  |
| 42.html-43.html | Add Edit Mode wizard | We need our field repair technicians to be able to update an incident report, so let’s follow MAX's next QuickStart suggestion: **Add Edit Mode**. |  |
| 43.html |  | With just one click we’ve created a typical editing screen, complete with Cancel and Save options, so the technicians can update the fields on the Detail screen. Nice! | Show clicking **Go to Edit Screen**? |
| 44.html |  | Now let’s use the Properties Inspector to format the fields of the Edit screen. |  |
| 45.html |  | Want to see the app’s flow so far? Click **Screen Flow** in the toolbar. |  |
| 46.html | Customer details/Card component | One last thing we need to do: create a customer detail screen that displays customer contact information. We can create the layout for this by dragging the **Card** component into the preview. |  |
| 47.html-49.html |  | The **Customer** business object provides the data that we need. Again, the live data preview comes in handy. |  |
| 50.html-51.html | Add Create Screen wizard | To let technicians update an incident report, we’ll use the last Quickstart wizard: **Add a Create Screen**. |  |
| 52.html-53.html |  | You can format this screen, but since it's for inputting data, there’s no need to wire it up to any data. | Show clicking **Go to Create Screen**? |
| 54.html | Building app | We're done designing our mobile app, but before we run it on a device, let’s open Screen Flow once more to check for errors or bugs. |  |
| 55.html |  | Nope, no bugs! Time to build our app. |  |
| 56.html-57.html | Running the mobile app/MAX App | ...And it's finished! You're ready to run the app on a device to see how it handles real data. To get this app on your device, you don't need to deal with Google Play or the App Store. All you need is this QR Code and the MAX App. |  |
| 57.html-58.html |  | On your phone, open the MAX App. Click the plus sign to use the built-in scanner on the QR code. |  |
| 59.html |  | Sign in. |  |
| 60.html |  | And here's the dashboard! |  |
| 61.html |  | Play around with the app by clicking the List tab, drilling down to the detail screens, and using the edit screen. It all works great! |  |
|  |  | You've just seen how to create a mobile app with no coding, no IDEs, and no emulators. No muss…no fuss! Thanks for watching! |  |