Maqetta means mockup, Part 1: Design an HTML5 mobile UI

Drag-and-drop HTML5 authoring in your browser

Tony Erwin
Software Engineer
IBM

04 January 2013

Need to prototype an HTML5 app? Forget coding. Hand-eye coordination is just about all you need to prototype with Maqetta, a browser-based WYSIWYG tool for desktop and mobile applications. This first article in a three-part series introduces this free, open source project that runs in a browser and lets designers drag and drop a rich set of widgets to build live UI mockups. In Part 1, get to know Maqetta's major functions and features while prototyping a realistic mobile application.

View more content in this series

Introduction

About this series

This series shows you how to use Maqetta to prototype HTML5 user interfaces.

- In this part, learn about Maqetta's major features while creating a prototype for a rich mobile application.
- In Part 2, take your prototype application to the next level by writing custom JavaScript to add interactive functionality.
- In Part 3, use PhoneGap to turn a Maqetta-generated mobile prototype into a native app that is ready to deploy to actual devices.

Learn more about using Maqetta in Tony's blog on developerWorks.

Maqetta is a free, open source visual authoring tool that makes designing rich HTML5 user interfaces simple and fun. A UI designer can use Maqetta, which runs in a browser without plugins or downloads, to build working prototypes for both desktop and mobile applications. Simply drag and drop widgets onto a device canvas to assemble a live UI mockup, and then pass that UI prototype to a developer for coding.

Maqetta and the Dojo Foundation

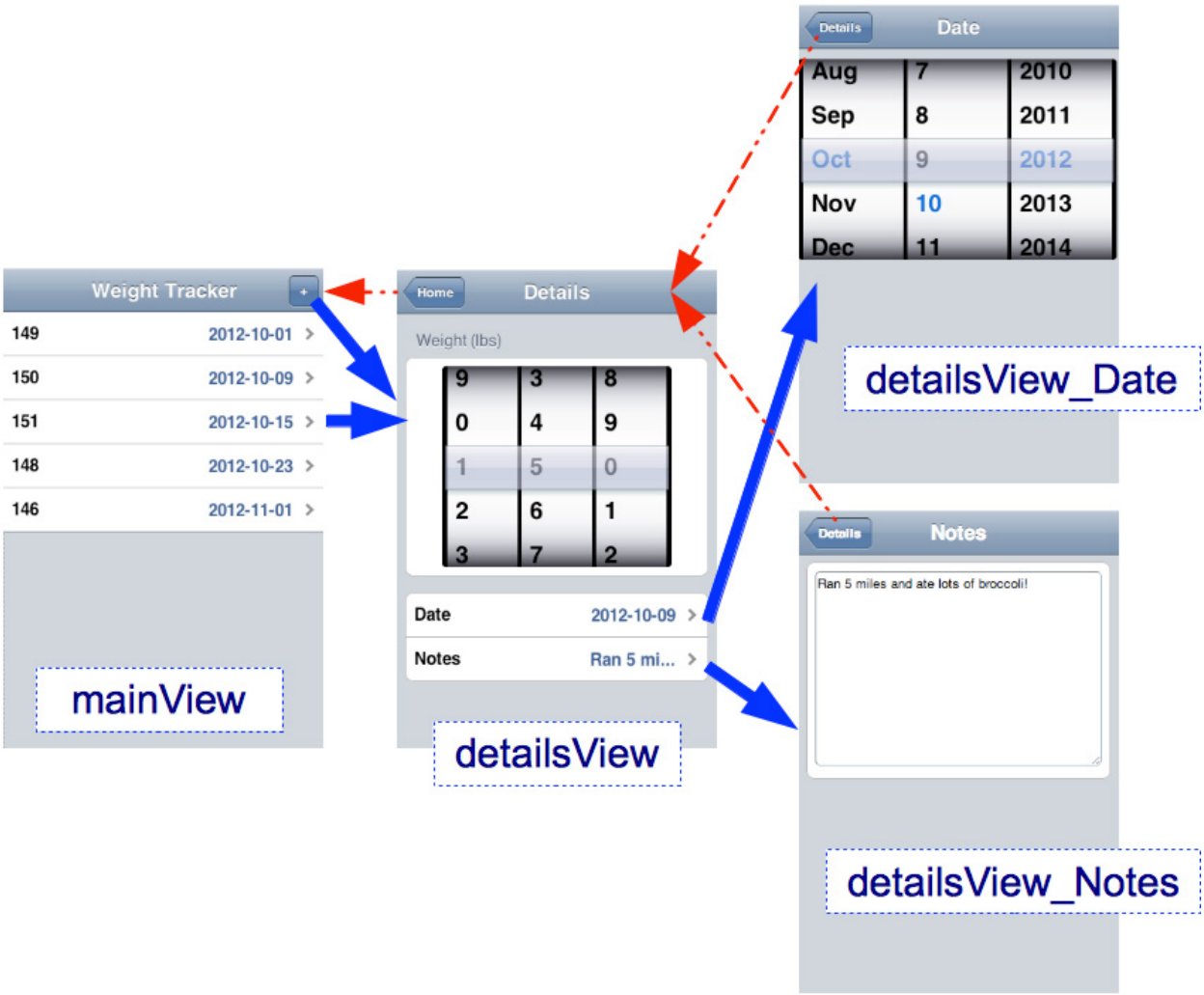
Maqetta was originally created by IBM and donated to the Dojo Foundation as an open source project in April 2011. It is made available under Dojo's liberal and commercial-friendly open source licensing terms. Maqetta is typically updated every few months and has undergone continual improvement since it was made publicly available. Releases in the second half of 2012 allowed for a major UI makeover and significant performance enhancements.

UI design for the weight tracker app

Mobile user interfaces typically consist of a series of screens or panels called *views*, which come and go as the user navigates to various parts of the app. The flowchart in Figure 1 shows the four views that will comprise our app as well as the navigational flow of user interaction.

The solid arrows in the flowchart represent forward navigation, and the dotted arrows represent backward navigation to a previous view. The label on each view represents the `ID` we will assign to that view when we create it in Maqetta.

Figure 1. A flowchart for the weight tracker application



The four application views are as follows:

- **mainView** is the first view shown when a user launches the app. It contains a list of numbers representing weights entered by the user. Clicking any of the weights in the list causes the next view, `detailsView`, to appear. Alternatively, a user could click the plus (+) button in the application heading to add a new weight.
- **detailsView** shows the properties (such as, weight, date entered, and notes) associated with a given weight entry. Clicking the **Date** item causes the `detailsView_Date` view to appear. Clicking the **Notes** item causes the `detailsView_Notes` view to appear. Clicking the **Home** button takes the user back to the `mainView`.
- **detailsView_Date** allows the user to set the date of the weight entry with a spin-wheel. This is an end-point in the navigation, as the only place the user can go from here is back to the `detailsView`. Clicking the **Details** button in the application heading brings the user back to the `detailsView`.
- **detailsView_Notes** allows the user to enter any notes related to the weight entry in a free-form text area. This is also an end point in the navigation, as the only place to go from here is back to the `detailsView`. Clicking the **Details** button in the application heading brings the user back to the `detailsView`.

While the weight tracker's design might seem complex, you can visually author a UI mockup in Maqetta without writing any code. You only need to drag-and-drop some widgets and edit some properties.

Launch Maqetta in the cloud

Maqetta installations

The Maqetta team provides a free, non-commercial, cloud-hosted version of Maqetta at Maqetta.org, which you may use as is, at your own risk.

You can also install Maqetta on your own intranet server after retrieving it from the Maqetta downloads page. You'll need Maqetta 8.0 or higher to successfully follow the examples in this series.

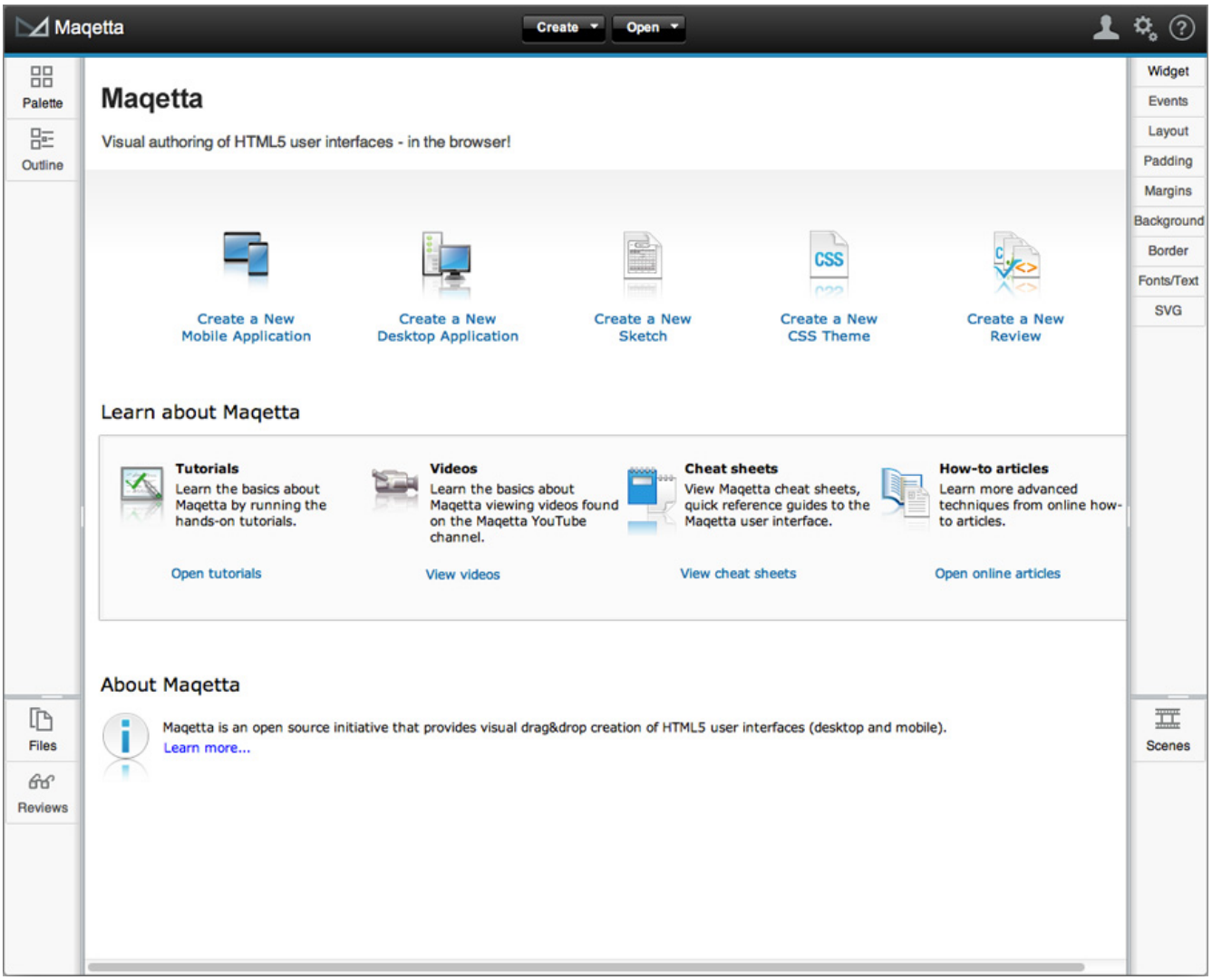
You'll get the most out of this article if you follow along and build the app as you read. It's easy — and free! — to get started:

1. Go to Maqetta.org using the most recent release of either Chrome, Firefox, or Safari.
2. Click the **Launch Maqetta** button.
3. If you already have an account:
    a. Enter your email and password.
    b. Click **Log In**.
4. If you don't have an account:
    a. Click the **Register** button at the bottom of the page.
    b. On the Create an Account page, enter your email address and click **Sign up**.
    c. Watch for an email with a link to confirm your registration and click the link when it arrives.
    d. After your request is confirmed, enter your email address and password and click **Log In**.

Maqetta welcome page

Once you've logged in, you'll see a workbench showing the Maqetta welcome page. The welcome page, shown in Figure 2, provides a convenient launching point for the key functions of Maqetta. It's your starting point for creating a new application, and it also offers links to Maqetta's documentation, tutorials, and cheat sheets (see Resources). You can also access all of the features of the welcome page from the pull-down menus at the top of the page.

Figure 2. The Maqetta welcome page

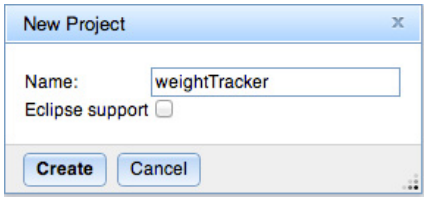


Create a new project

When you first use Maqetta, you will work with files in the default project called `project1`. While not strictly required, creating one project for each application you develop will help you stay organized and make it easier for you to export your work for use in other tools. So, let's start by creating a new project for the weight tracker application:

1. Choose the **Create > Project...** menu option from the toolbar at the top of the Maqetta page.
2. In the New Project dialog, enter "weightTracker" for the project's name, as shown in Figure 3.

Figure 3. New Project dialog



3. Click the **Create** button.

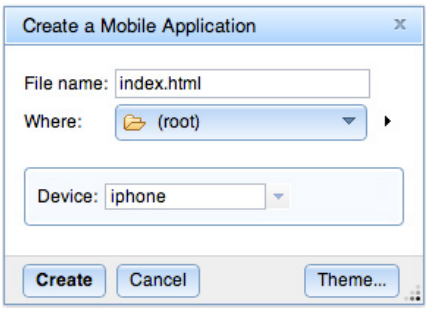
If you want to switch back to `project1` (or any other project), choose from a list of projects in the Files palette.

Create a mobile application

Now let's create the HTML file that will be used by the application. (When you've completed the steps in this article, you can compare your work with the HTML file in the Downloads section below.)

1. Choose the **Create > Mobile Application...** menu option.
2. In the Create a Mobile Application dialog (see Figure 4), enter "index.html" for the file name. (Note that `index.html` will be the required file name should you choose to build the weight tracker application for deployment to mobile devices using PhoneGap. We'll explore that option in Part 3 of this series, so you might as well name the file correctly now.)
3. Leave the device field set to the **iphone** option. You can change the device type later if you wish.

Figure 4. Create a Mobile Application dialog

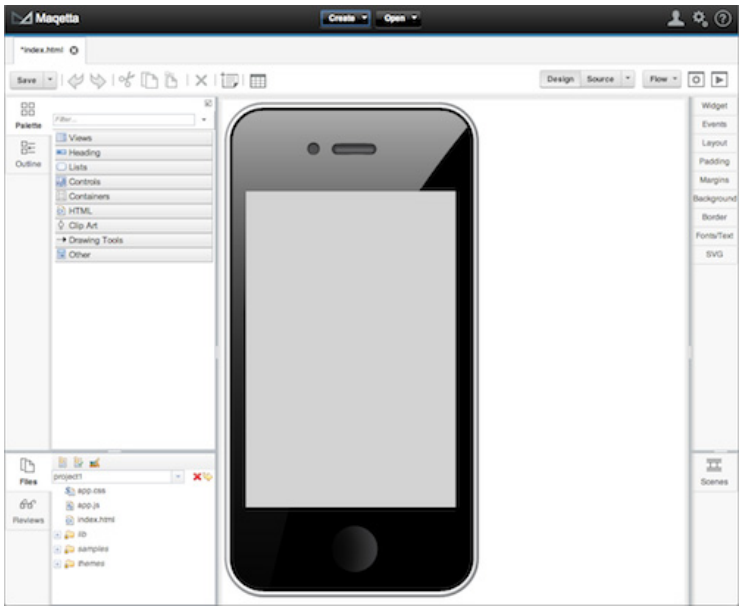


4. Click the **Create** button.

Mobile Page Editor

After you dismiss the Create a Mobile Application dialog, the Maqetta welcome page will be replaced by Maqetta's Mobile Page Editor displaying an empty iPhone silhouette, as shown in Figure 5. Besides iPhone, Maqetta provides authoring support for other popular phones and tablets such as iPad, Android, and Blackberry. Regardless of what device you've selected, the Mobile Page Editor will display a life-sized replica of that device with your application in view. As you drag and drop mobile widgets onto the silhouette, they will be rendered with the look-and-feel of your target platform.

Figure 5. A new mobile application canvas with the iPhone silhouette



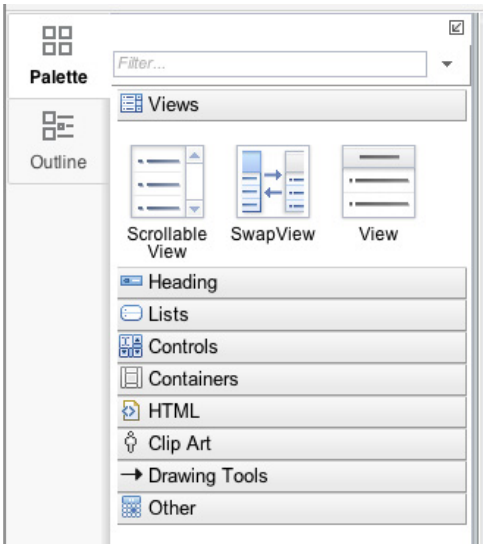
Build the Main view

Once you've created the HTML file, it's time to start building the `mainView` from the flowchart.

The Palette tab on the left side of the workbench contains all the widgets that are applicable to the type of application being designed (a mobile app, in this case). The widgets are organized by section according to their purpose. The widgets in the Views, Heading, Lists, Controls, and Containers folders are from the Dojo Mobile library (see Resources). We'll use only a fraction of the 50 or so widgets in those folders for the weight tracker.

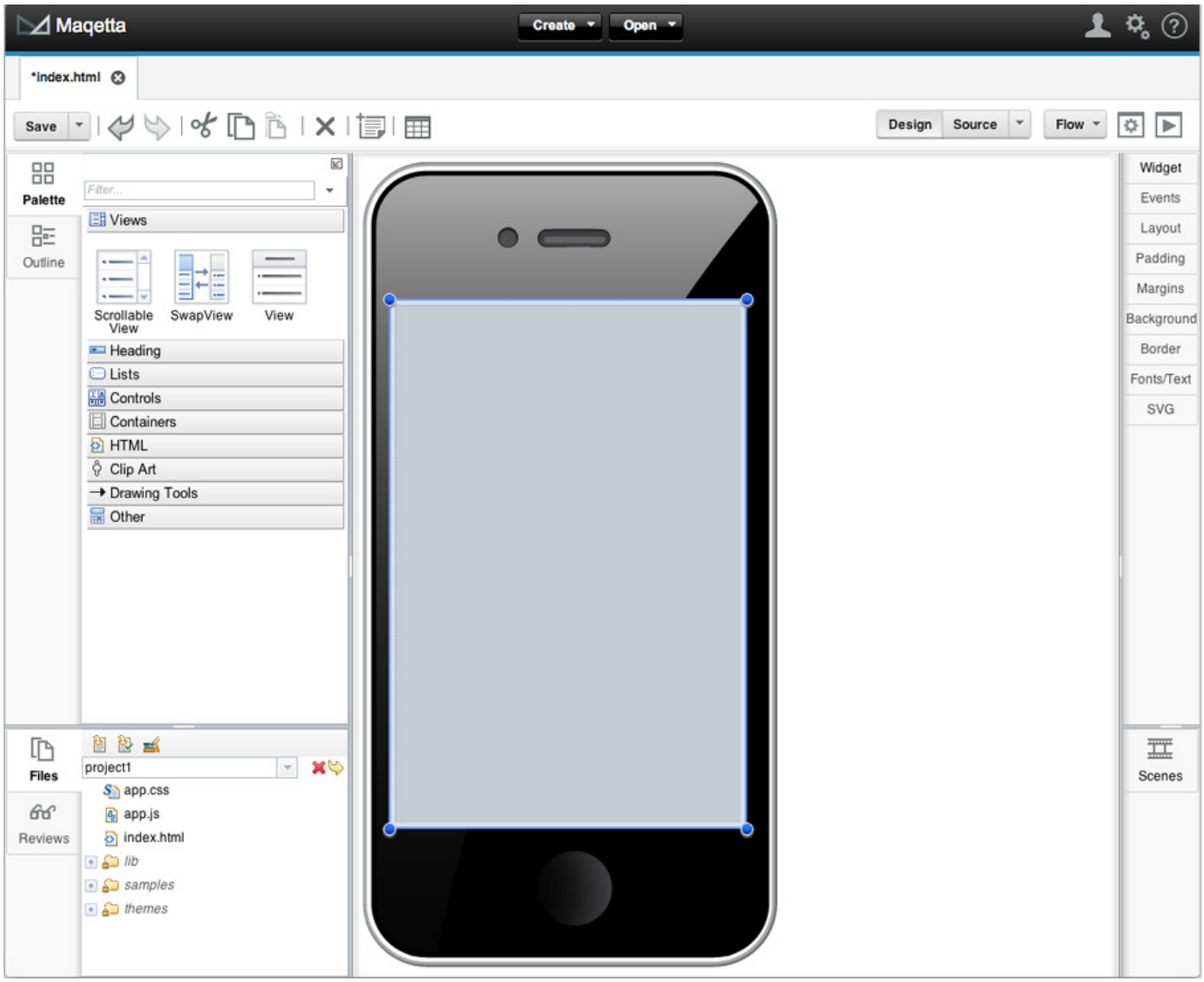
1. Open the Views section to see the available view widgets, as shown in Figure 6:

   ### Figure 6. Maqetta mobile view widgets

2. Select the `ScrollableView` widget, which is the first one in the Views section, and drag and drop it onto the device silhouette. Your display should look something like the screenshot in Figure 7:

Figure 7. Adding ScrollableView to the iPhone silhouette

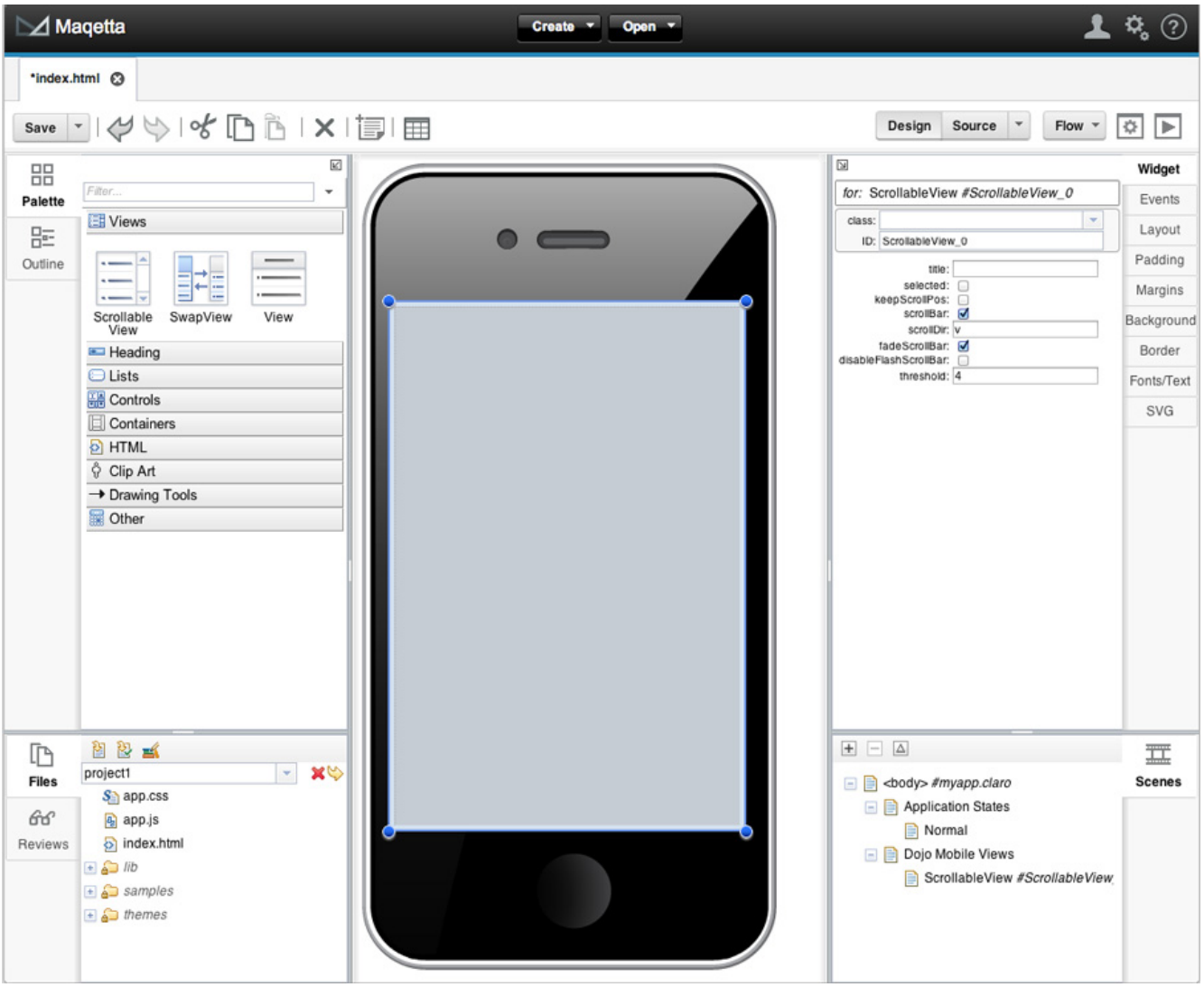


Notice the following:

- The page canvas now has a bluish-gray background. This is because the first Dojo Mobile widget added to the page brings in the CSS stylesheets that provide a native look-and-feel for a particular mobile device.
- There is a selection rectangle around the entire page. This is because the `ScrollableView` is selected, and it has default width and height of 100%.

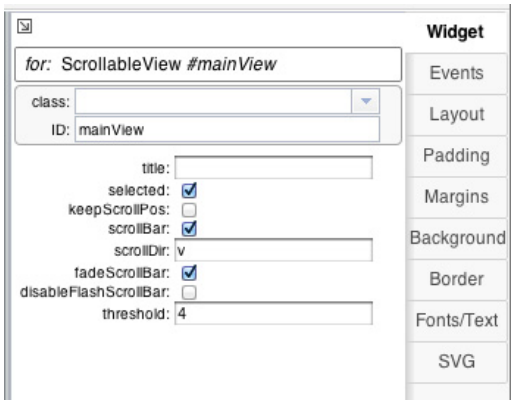
3. Now click the Widget tab on the right side of the Maqetta workbench. This expands the Properties palette with focus on the widget-specific properties section, as shown in Figure 8. Notice that the `ID` defaults to `ScrollableView_0`, and that the top of the Properties palette shows `for: ScrollableView #ScrollableView_0` (indicating the type and `ID` of the widget that the properties are for).

Figure 8. Maqetta workbench showing the Properties tab



4. Type "mainView" into the `ID` field and hit **Return** to commit the change. As you see in Figure 9, the top of the Properties palette now shows `for: ScrollableView #mainView`. This `ID` matches the design shown in the original application flowchart. (See Figure 1.)

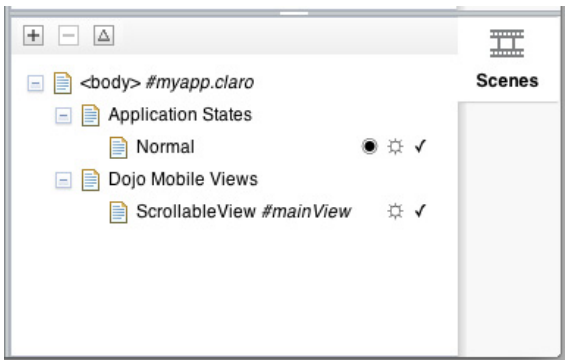
Figure 9. Properties palette for views



Note that the `ID` for a view is important. The Dojo Mobile widget library uses the view `ID` to reference views when hiding and showing them during user navigation. With that in mind, you will reference view `ID`s as you set up the app's navigational flow using Maqetta. And, if you continue on to Part 2 in this series, you will use the `ID`s in your code, so it is important to set them correctly.

5. Notice the Scenes palette in the lower-right corner of the workbench, which is shown in Figure 10 below. All the views that you create appear under the Dojo Mobile Views node in this palette. In this case, the node should now contain `ScrollableView #mainView`.

Figure 10. The Scenes palette

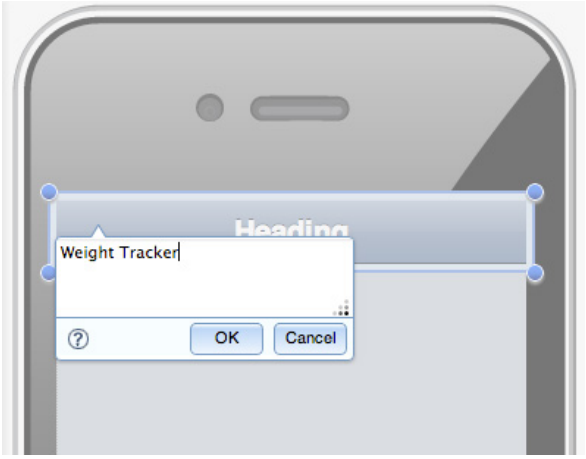


Add a heading

Let's continue constructing the `mainView` by adding a heading:

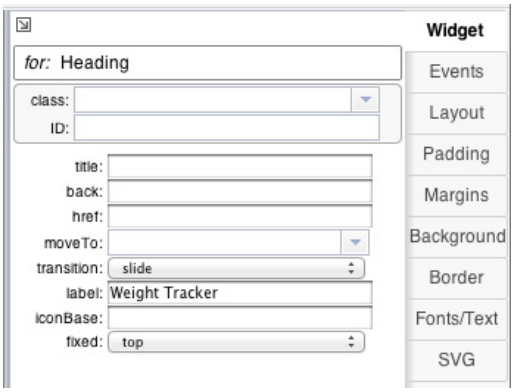
1. Drag and drop a `Heading` widget from the Heading section of the widget palette onto the middle of the device silhouette. (If you have trouble finding a widget, you can type its name into the search box at the top of the widget palette.)
2. Enter “Weight Tracker” in the textbox that appears, as shown in Figure 11, and click **OK** to set the text of the heading. If you wish to change the text later, just double-click the heading to see the textbox again. (Note that this process applies to all widgets that present a data-entry dialog when the widget is first created.)

Figure 11. Add a heading



3. In the widget section of the Properties palette shown in Figure 12, change the `fixed` attribute to `top`. This causex the heading to remain in the same location (at the top of the application display) if vertical scrolling is required elsewhere in the view.

Figure 12. Set the heading properties



Add the weight list

> **Units of measurement**
>
> This prototype uses English units of measurement, also known as the United States Customary System. Prior to deploying, though, we would want to enable the user to choose between English and Metric units of measurement.

Our next step is to add a widget that shows the list of weights entered by the user:

1. Drag and drop an `EdgeToEdgeDataList` from the Lists section of the widget palette, and place it on top of the pane you just added to the iPhone silhouette. The data list will be created, and a dialog to set the list's contents will appear, as shown in Figure 13. In each line, the first value is the label to show for that row, and the second value is the `ID` of the view that the application will navigate to when that row is clicked.

Figure 13. Add EdgeToEdgeDataList

2. Change the contents of the textbox to the set of weights (in pounds) shown in Listing 1 and click **OK**. Notice that `detailsView` is the `ID` for the view that the application will navigate to when a given row or weight value is clicked. We defined this navigation in the original application flowchart (see Figure 1).

Listing 1. Input for the EdgeToEdgeDataList

```
149, detailsView
150, detailsView
151, detailsView
148, detailsView
146, detailsView
```

3. Return to the Properties palette and change the `ID` of the list to `weightList`. At this point, you've mostly completed the `mainView`. You should see something like the screenshot in Figure 14:

Figure 14. Main view in the Mobile Page Editor with iPhone rendering

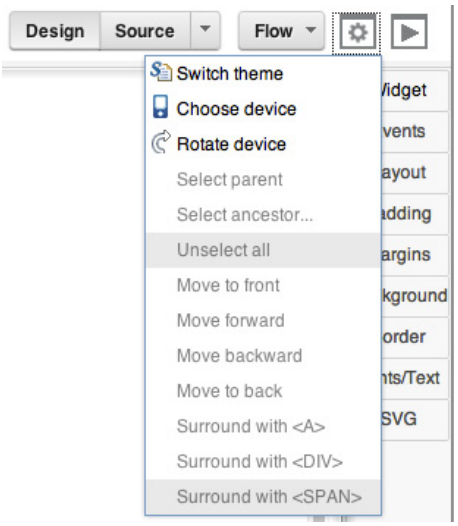
Note that we've set the `ID` here because it will be important in Part 2. A UI designer would not typically set the `ID` for a non-view widget like `weightList`; that would happen during the coding of the application UI.

Change the device type

Designers often want to see a UI design rendered for different environments. Fortunately, it's easy to change the device type and rendering in Maqetta.

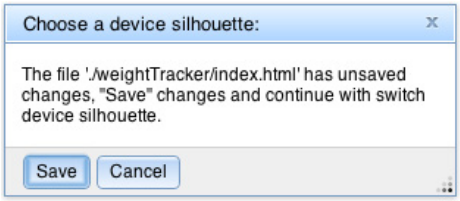
1. Open the Document Settings menu, which is the second icon from the right in the mobile editor's toolbar, and click **Choose device**. (Note that this menu also holds an option to rotate the device.)

Figure 15. Document Settings menu



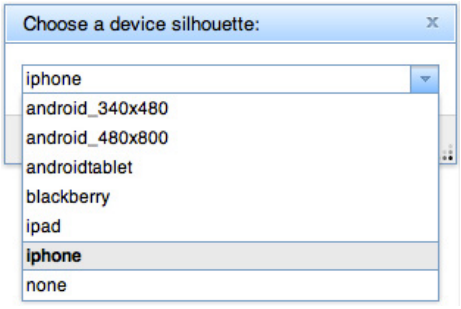
2. If you have unsaved changes in your HTML file, you'll get a warning that you need to save your changes before switching devices, as shown in Figure 16. Click the **Save** button.

Figure 16. Warning!



3. Clicking **Save** brings up the dialog for choosing a device silhouette. The dialog contains a drop-down list with a number of device options, as shown in Figure 17:

Figure 17. Choose a device silhouette



4. Choose the device that you want to see your UI rendered on (for example, **android_340x480**), and click **Select**. Maqetta's device rendering will change to match the selected device, as shown in Figure 18:

Figure 18. The Maqetta page editor renders mainView for Android

You can continue to drag and drop widgets and edit widget properties just as if you had never changed devices, but the look-and-feel will be for the newly selected device.

5. To reset the rendering, select **Choose device** from the Document Settings menu, select **iphone** from the device list, and click **Select**.

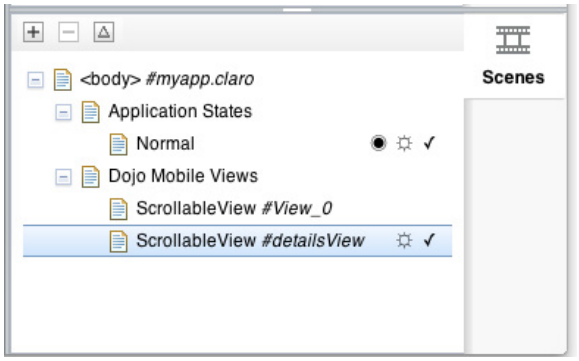
Start the Details view

Next, we'll begin constructing the Details view (`detailsView`) from the application flowchart shown in Figure 1. This view shows the details of any given weight entry.

1. Select a `ScrollableView` in the Views section of the widget palette, and drag and drop it onto the `mainView`. This action adds a new view as a sibling (meaning it's at the same level in the element tree) to the current one (`mainView`, in this case). The newly added view will be empty and selected.

2. In the Properties palette, change the `ID` of the new view to `detailsView`. As mentioned earlier, setting the view `ID` correctly is very important. When we configured the `EdgeToEdgeDataList` in `mainView`, we set it up to navigate to a new view with an `ID` of `detailsView`.
3. Take a look at the Scenes palette shown in Figure 19:

Figure 19. Scenes palette showing the new view



Note the following:

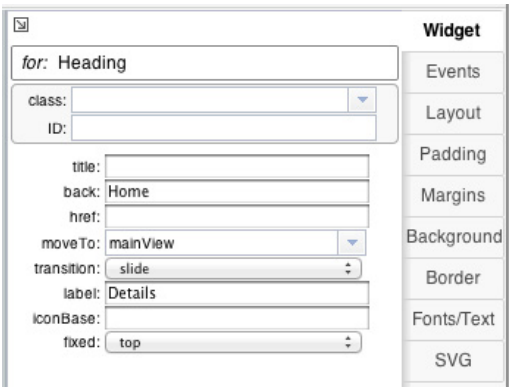
- There are now two views under Dojo Mobile Views: `ScrollableView #mainView` and `ScrollableView #detailsView`. The new view is a sibling of the first one.
- The second view becomes selected and visible (just like what happens whenever a new widget is added), but also notice that the first view widget is now invisible. With Dojo Mobile, only one view widget is visible at any given time. Maqetta mimics this runtime behavior at design time by managing the visibility of view widgets such that whenever one view is shown, all sibling view widgets become hidden.
- You can alternate between view widgets by clicking them under Dojo Mobile Views. The view selected here will be the one made visible in the device silhouette.

Add the Details view heading

Let's continue constructing the `detailsView` by adding a heading:

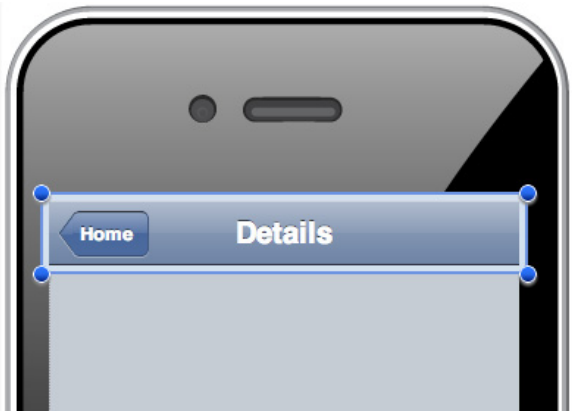
1. Ensure that `detailsView` is selected in the Scenes palette.
2. Select the `Heading` widget in the Heading section of the widget palette, and drag and drop it onto the view.
3. In the dialog that appears, enter "Details" into the textbox, and click **OK** to change the heading text.
4. Make the following changes to the Properties palette, which is shown below in Figure 20:
   - Change the `back` attribute to `Home`. This creates a button with a label of "Home" within the heading, which sends the user back to the previous view.
   - Change the button's `moveTo` attribute to `mainView` by choosing **mainView** from the pull-down menu. This causes the application to navigate to the main view whenever the **Back** button is clicked.
   - Change the button's `fixed` attribute to `top`.

Figure 20. Properties palette for heading



The `detailsView` should now look something like the screenshot in Figure 21.

Figure 21. Details view with a heading



Preview and run the application

You can run a live preview at any time when you're working in Maqetta. Because we've built two views and defined the navigation between them, let's preview a live version of the weight tracker prototype to make sure that everything is set up correctly so far.

1. In the Scenes palette, start by selecting the `mainView`. The view selected when the preview is run is the first one shown.
2. In the toolbar, click the **Preview in Browser** icon (the last entry in the toolbar). This launches a new browser tab that contains the weight tracker app running inside of the iPhone mobile device silhouette, as shown in Figure 22:

Figure 22. Weight tracker preview running in the iPhone silhouette

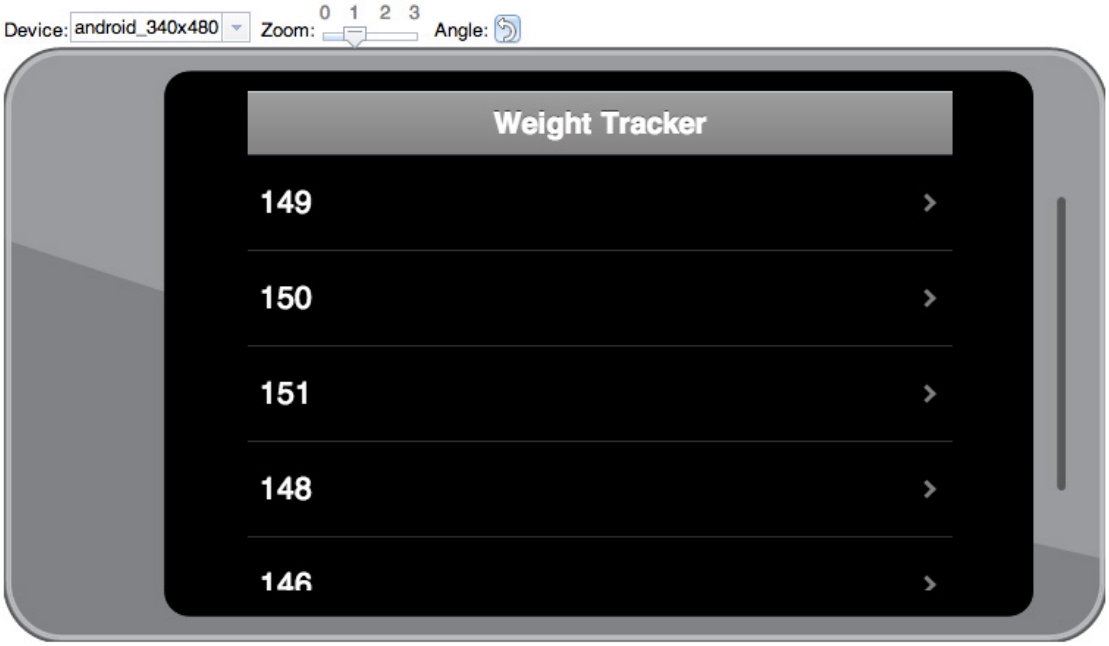
3. Click one of the entries in the drop-down weight list, and notice that the `detailsView` replaces the `mainView`. Application widgets in preview mode respond to user interaction just as though they're running live on the device.
4. Click the **Home** button in the `detailsView` heading, and notice that `mainView` reappears.

And, with that, this prototype has been successfully tested!

Change the device and orientation in Preview

Here are a few more preview tips. First, notice that the live preview has controls for switching the device, controlling the zoom level, and changing the application's view orientation from portrait to landscape. You can experiment with these controls by switching to another device and using the **Angle** button to flip the view on its side. My updated rendering is shown in Figure 23.

Figure 23. Android preview in landscape

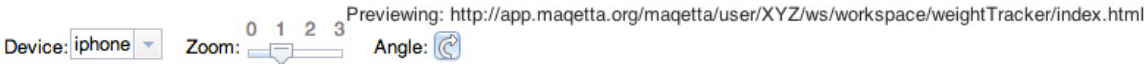


Note that the code remains live so you can still test the navigational flow between `mainView` and `detailsView` with the new configuration.

Run the app in a mobile bowser

Finally, notice the URL next to the **Previewing** label in the Maqetta preview, which is shown in Figure 24.

Figure 24. Preview toolbar with URL



You can use this URL to view a prototype on a mobile device browser, for instance by pasting it into a Safari browser on an iPhone. While previewing the app in a mobile browser is valuable (especially if your final goal is to have a web-hosted app), keep in mind that it's different from installing and running the application as a true native app. We'll see how that works in Part 3 of this series, when we combine Maqetta with PhoneGap.

Create a SpinWheel for the Details view

Having previewed our mobile application's basic features and navigation, let's return to the Details view and construct the widgets that allow users to specify their weight.

We will construct a `RoundRect` (labeled with a `RoundRectCategory`) to act as a container for a `SpinWheel` widget. The `SpinWheel` looks a bit like a weight scale. We'll populate it with numerals representing weights in pounds (recall that we're deliberately using English units of measurement for the application prototype) then adjust its presentation a bit. In the process, we'll get to use some parts of the Properties palette that we have not yet explored — namely, the Layout and Margin tabs.

Construct the RoundRect and SpinWheel

1. Return to the Maqetta page editor and select `detailsView` from the Scenes palette.
2. Select the `RoundRectCategory` widget from the Lists section of the widget palette, and drag and drop it onto the `detailsView` canvas. This will act as the label for our `SpinWheel`. Enter "Weight (lbs)" into the textbox that appears, as shown in Figure 25, and click **OK**.

   Figure 25. Adding RoundRectCategory to the Details view

3. Now drag and drop a `RoundRect` from the **Containers > Dojo** section of widget palette onto the view below the `RoundRectCategory`, as shown in Figure 26.

   Figure 26. Adding a RoundRect to the Details view

4. Drag a `SpinWheel` widget from the **Controls > Pickers** section of the widget palette, and drop it on top of the `RoundRect`, making it a *child* of the rectangle (that is, contained by the rectangle).

The `SpinWheel` shown in Figure 27 consists of two slots containing letters. Eventually, we want to have three slots containing digits, thus simulating a weight scale.

Figure 27. The default SpinWheel in the Details view

Customize the SpinWheel with three slots

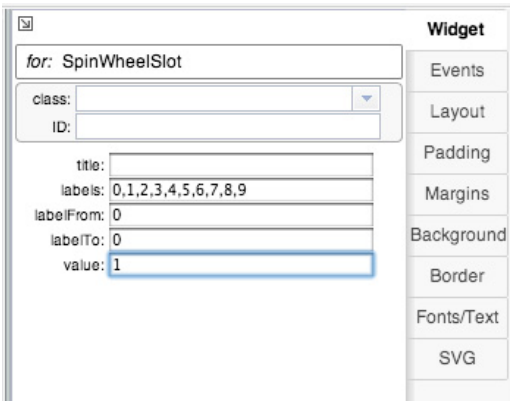
1. Using the Properties palette, change the `ID` of the `SpinWheel` to `weightSpinWheel`. Next, select the first `SpinWheel` slot, double-click it, and enter "0,1,2,3,4,5,6,7,8,9" into the textbox that appears. Click **OK**.

   ### Figure 28. Changing the content of first SpinWheel slot

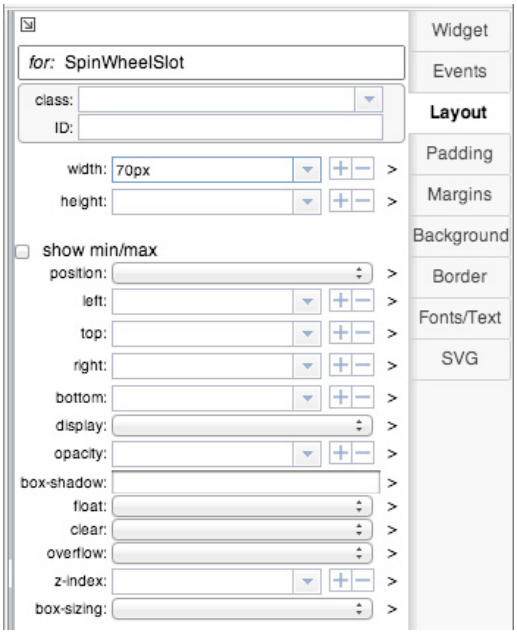
2. With the first slot still selected, change the `value` attribute in the widget section of the Properties palette to `1`. Notice how the value of the `labels` attribute shown in Figure 29 now matches what we've specified:

Figure 29. Set properties for the first SpinWheel slot



3. Click the Layout tab in the Properties palette, and change the first `SpinWheel` slot's `width` attribute to `70px`, as shown in Figure 30. Notice how the slot size changes in the page editor so that there's less space around the digits.

Figure 30. Layout properties for the first SpinWheel slot



4. Select the second `SpinWheel` slot and double-click it. Enter "0,1,2,3,4,5,6,7,8,9" into the textbox that appears, and hit the **Return** key.
5. With the second slot still selected, change the `value` attribute in the widget section of the Properties palette to `5`, and set the `width` attribute in the Layout section to `70px`.
6. Now drag a third `SpinWheelSlot` from the **Controls > Pickers** section of widget palette onto the `SpinWheel`, and drop it to the right of the second slot. This new slot will be a child of the `SpinWheel` and a sibling to the other two slots.

7. Change the text in the textbox that appears to "0,1,2,3,4,5,6,7,8,9" and click **OK**.

Figure 31. Adding the third slot

8. With the third slot still selected, change the `value` attribute in the widget section of the Properties palette to `0`, and change the `width` attribute in the Layout section to `70px`. Your third slot should now look like what you see in Figure 32:

Figure 32. The finished third slot

Tip!

Sometimes it can be tricky to select a parent widget when its child widgets are also selectable. An easy way to select a parent widget is to right-click any selected child widget and choose the **Select parent** option.

Tweak the layout

The `SpinWheel` is really too wide for its three slots, but we can fix this by changing the `SpinWheel`'s `width` attribute:

1. Select the `SpinWheel` by clicking the area next to the last slot.
2. In the Layout section of the Properties tab, change the `width` attribute to `222px`. The `SpinWheel` should look like what you see in Figure 33.

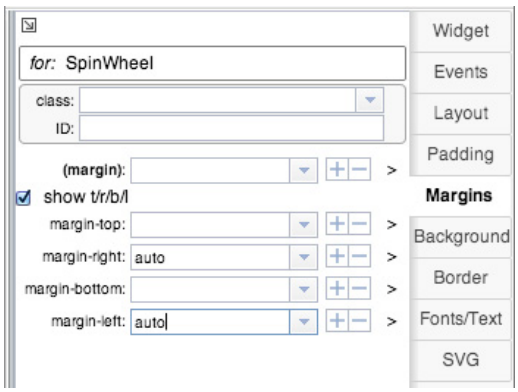
Figure 33. Improved SpinWheel

Next, notice all the whitespace to the right of the `SpinWheel` in the iPhone canvas. Centering the `SpinWheel` will improve the overall UI layout. We can adjust the `SpinWheel`'s margins as follows:

1. Click the Margins section of the Properties palette.
2. Check the box next to **Show t/r/b/l**.
3. Change `margin-right` and `margin-left` to auto.

These adjustments are shown in Figure 34.

Figure 34. Adjusting the SpinWheel's margins



The `SpinWheel` is now finished with a value of 150 and a centered alignment within the round rectangle, as shown in Figure 35.

Figure 35. The completed SpinWheel

Finish the Details view

We'll complete the Details view by adding a `RoundRectList` widget to show the date and user notes associated with a given weight entry. Application users will use the `RoundRectList` to navigate to the Date and Notes views. Before we build the `RoundRectList`, we need to create placeholders for the Date and Notes views, which will make it easier to configure the navigational flows.

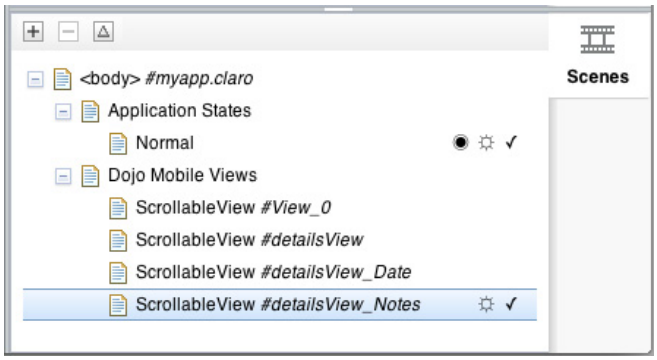
Add placeholder views

Start by creating the two placeholders:

1. Drag a `ScrollableView` widget from the widget palette onto the current view (either onto the heading or onto the gray area). As you've seen before, this will create an empty view that is initially selected.
2. Using the Properties palette, change the `ID` of the new view to `detailsView_Date`.
3. Drag another `ScrollableView` widget from the widget palette onto `detailsView_Date`.
4. Using the Properties palette, change the `ID` of the new view to `detailsView_Notes`.

You should now see all four views from the flowchart in the Scenes palette, as shown in Figure 36.

Figure 36. Scenes palette showing all four views



Create a list for view navigation

Our next step is to add a `RoundRectList` widget, which will enable the user to navigate from the Details view for a given weight to its Date and Notes views.

1. Select `detailsView` from the Scenes palette so that it's the active view.
2. Drag a `RoundRectList` from the widgets palette onto the view, just below the `RoundRect` containing the `SpinWheel`.
3. In the textbox that appears, each line will be translated into the label for a new `ListItem`. Enter "Date" on one line and "Notes" on the second line, as shown in Figure 37:

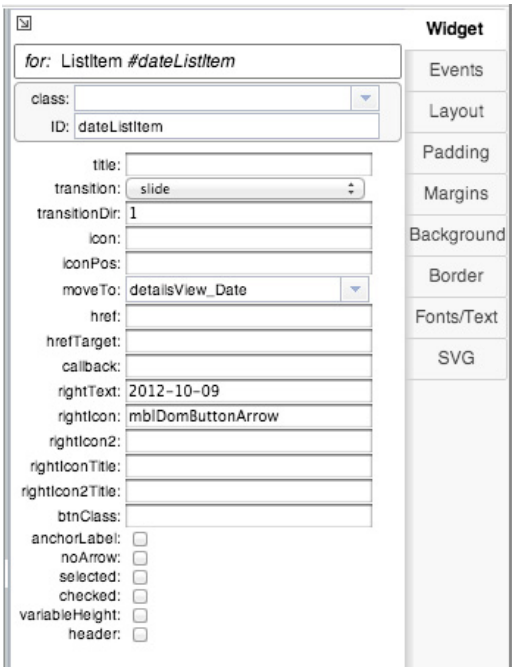
Figure 37. RoundRectList input

4. Click **OK**.

Select the list item labeled "Date" and make the following changes in the widget section of the Properties palette:

1. Set `moveTo` to `detailsView_Date` by selecting it from the pull-down next to the field's label. The weight tracker application will now navigate to the `detailsView_Date` view when the user clicks on this item.
2. Modify `rightText` to "2012-10-09", which will cause that text to appear on the right side of the item. (Note that we're not doing date localization for this prototype; all dates will be formatted according to the generic ISO-8601 standard. Please stick to this format because it will be important for Part 2 of this series.)
3. Change `ID` to `dateListItem`.

Figure 38. Date list item properties



4. Select the second list item (with the label “Notes”) and make the following changes in the Properties palette:
    - Set `moveTo` to `detailsView_Notes` by selecting it from the pull-down next to the field's label. The weight tracker application will navigate to `detailsView_Notes` when the user clicks on this item.
    - Modify `rightText` to "Ran 5 mi..." This gives the user a preview of the full note, which in this case reads: "Ran 5 miles and ate lots of broccoli!"
    - Change the `ID` to `notesListItem`.

Our `detailsView` is complete and should look like the screenshot in Figure 39.

Figure 39. The completed Details view

Build the Date view

Creating our third view, `detailsView_Date`, should be pretty intuitive by now. Follow these steps:

1. Select the `detailsView_Date` view from the Scenes palette.
2. Drag and drop a `Heading` from the widget palette onto the view. Enter "Date" into the textbox that appears, and click **OK**.
3. In the Properties palette, change the `back` attribute to "Details," the `moveTo` attribute to `detailsView` (to match the design in the flowchart), and the `fixed` attribute to `top`.
4. Drag and drop a `SpinWheelDatePicker` (from the **Controls > Pickers** section) below the heading.
5. In the Properties palette, change the `ID` to `dateSpinWheel`.
6. In the Margins section of the Properties palette, click the checkbox next to **show t/r/b/l** and change `margin-top` to `10px` to add a little space between the heading and the spin-wheel.

The completed view should look like the screenshot in Figure 40.

Figure 40. Completed Date view

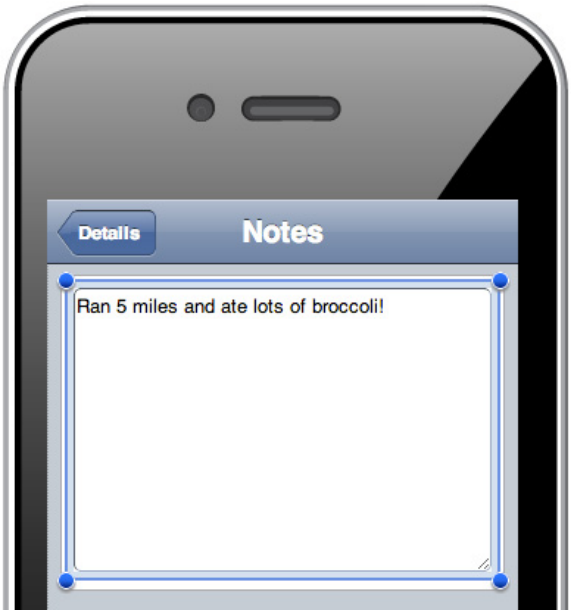
Build the Notes view

Now we'll build our fourth, and final, view:

1. Select `detailsView_Notes` from the Scenes palette.
2. Drag and drop a `Heading` from the widget palette onto the view. Enter "Notes" into the textbox that appears, and click **OK**.
3. In the widget section of the Properties palette, change the `back` attribute to "Details," the `moveTo` attribute to `detailsView` (to match the design in the flowchart), and the `fixed` attribute to `top`.
4. Drag and drop a `RoundRect` (from the **Containers > Dojo** section) below the heading.
5. Drag and drop a `TextArea` (from the **Controls > TextBox** section) into the `RoundRect`.
6. Double-click the new text area, enter "Ran 5 miles and ate lots of broccoli!" in the textbox that appears, and click **OK**.
7. Increase the size of the text area. In the Layout section of the Properties palette, change the `width` attribute to `100%` and the height attribute to `150px`.
8. Change the `ID` to `notesTextArea`.

The Notes view should look like the screenshot in Figure 41.

Figure 41. Notes view



Fine-tune and finalize the app

At this point, the UI prototype is fairly complete, but we still need to provide a mechanism for the user to add new weight entries. We'll add a plus button for this purpose. Then we'll add one last piece of polish to our weight tracker's UI, by enabling users to see a date along with the weight entry in each row of the weight list.

Add a plus button

To add the plus button:

1. Select the `mainView` from the Scenes palette to make it active.
2. Drag and drop a `ToolbarButton` (from the Heading section of the widget palette) onto the "Weight Tracker" heading. Change the value in the text box that appears to "+" and click **OK**.
3. Buttons for adding new entries in mobile apps are often placed on the right side of headings. To do this, change the `float` attribute to `right` in the Layout section of the properties palette.
4. In the Widget section of the Properties palette, change `moveTo` to `detailsView` so that navigation will occur when the user clicks the button.
5. Change the `ID` attribute to `addWeightButton`.

Figure 42 shows how the `mainView` should look after completing these steps.

Figure 42. Plus button to add a weight entry

Add right text to the EdgeToEdgeDataList

Next (and finally) we want to show a date along with the weight in each row of the weight list. For this we'll take advantage of Maqetta's ability to bind an `EdgeToEdgeDataList` to data in a file.

1. Choose **Create > JavaScript File...** from the main Maqetta menu to create the file where we will place our new data.
2. In the dialog that appears, change the file name to `weights.json` and click the **Create** button. This will cause a new editor tab to appear.
3. Replace the contents of the editor with the contents of Listing 2.

Listing 2. Contents for weights.json

```
{
    "items": [
        {label: "149", moveTo: "detailsView", rightText: "2012-10-01"},
        {label: "150", moveTo: "detailsView", rightText: "2012-10-09"},
        {label: "151", moveTo: "detailsView", rightText: "2012-10-15"},
        {label: "148", moveTo: "detailsView", rightText: "2012-10-23"},
        {label: "146", moveTo: "detailsView", rightText: "2012-11-01"}
    ]
}
```

You'll notice that the data is similar to what we entered in the smart input box when we first created the `EdgeToEdgeDataList`. Recall that in the textbox each row had a `label` and a `moveTo` value separated by a comma. In the file, each row has explicit `label`, `moveTo`, and (new) `rightText` properties.

4. Save and close the file.
5. Go back to the page editor and double-click the `EdgeToEdgeDataList` containing the weights. This will bring up the dialog where you first set the list's contents.
6. Change the pull-down from **Comma separated data** to **Data file from workspace**.

7. Enter `weights.json` (the name of your file) into the textbox (as shown in Figure 43) and click **OK**. (Note that you can also navigate to a file in your workspace by clicking the folder icon next to the textbox.)

Figure 43. Specifying a data file from workspace

You should now see a date at the right of each entry in the list, as shown in Figure 44.

Figure 44. Dates added to weight list

Verify the prototype

Our weight tracker application prototype is now complete — and we haven't written a single line of code! If you haven't already, preview the app (use the **Preview in Browser** icon in the toolbar) and verify that you can navigate forward to all of the views and then get back to the first view.

You might also want to exercise some of the mobile widgets, like the spin-wheels for adjusting the application's weight and date variables, and verify that the plus button in the `mainView` navigates to the `detailsView`.

Conclusion

In this article, you've seen how to quickly built a reasonably complete, live prototype for a personal mobile weight tracker application. If this were a production application, you might take any of the following next steps:

- Use Maqetta's live preview feature to:
  - Demo the proof of concept to your boss and possibly convince him or her that your company should get into the mobile weight tracking business.
  - Conduct usability tests with your potential customers before moving on to the development phase.
- Use Maqetta's Review & Commenting feature (see the **Create > Review...** menu option) to get feedback from your team.
- Continue building out the prototype; for example, a production application would eventually need a Settings view for English versus Metric units of measurement. You might also want to add panels for managing different users, setting goals, entering meals, and tracking exercise routines or progress.
- Create a custom theme for your application with Maqetta's Mobile Theme Editor. For instance, you could style the UI with your company's branded color scheme.
- Decide that the design is solid and ready to be passed on to your development team for development into a product. In that case, you could use Maqetta's mechanism to export projects as a zip file, which could then be imported into another development environment such as Eclipse or IBM's Rational Application Developer.

In this article, you've learned how to build a sophisticated and fairly complex prototype for a mobile application UI — without writing any code! In Part 2, we'll add some custom JavaScript for handling user interactions, and in Part 3, we'll use PhoneGap to build another prototype as a native application that can be deployed to actual mobile devices. In the meantime, check out the Resources section to learn more about Maqetta, HTML5, and Dojo Mobile.

Acknowledgments

Special thanks to the Maqetta team (Jon Ferraiolo, Javier Pedemonte, Adam Peller, and Bill Reed) for thoughtfully reviewing and providing constructive feedback on this series of articles.

Downloads

Description	Name	Size
Resulting HTML file	maqetta_part1.zip	3KB

Resources

Learn

- Maqetta documentation: Get tutorials and cheat sheets from the Maqetta development team.
- MaqettaYouTube Channel: Check out online video demonstrations, including an introduction to Maqetta composition types.
- Tony Erwin's Maqetta blog: Learn more from this author, who is part of the Maqetta development team.
- @Maqetta: Follow Maqetta on Twitter.
- Maqetta on developerWorks: More resources for learning how to work with Maqetta.
- HTML5 fundamentals: Follow this developerWorks knowledge path to learn the fundamentals of working with HTML5.
- The W3C HTML5 Wiki: Learn even more about HTML5.
- "Getting Started with dojox/mobile" (Dojo.org): Find out more about Dojo Mobile, a framework for creating cross-device-compatible mobile web applications.
- "What's new in Dojo Mobile 1.8, Part 1: New widgets" (Yoshiroh Kamiyama, developerWorks, November 2012): Discover the new widgets, widget-related utility classes, and modules introduced in Dojo Mobile 1.8.

Get products and technologies

- Maqetta.org: Launch Maqetta in the cloud.
- Download Maqetta: Install Maqetta on your own intranet server after retrieving it from the downloads page.

Discuss

- Join the Maqetta user group: Interact with other designers and developers using Maqetta to create desktop and mobile UIs.
- Get involved in the My developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.

About the author

Tony Erwin

Tony Erwin is a Software Engineer in IBM's Emerging Internet Technologies group and a core member of the Maqetta development team. Tony has been with IBM since 1998 and has extensive UI design and development experience using a wide variety of technologies and toolkits. Before joining IBM, Tony earned an MS in Computer Science from Indiana University and a BS in Computer Science from Rose-Hulman Institute of Technology.