Navigation Strategies in Perspective

In addition to traditional navigation, Perspective offers a variety of new approaches, including many that might look familiar to smartphone users. In designing Perspective, we incorporated a few common trends in UI design, particularly those favored in responsive design. The strategies below should help phone-savvy users navigate easily in a Perspective project.

There are many ways to set up your navigation, and the best looking projects will mix several types together. In this section we'll walk through a few examples:

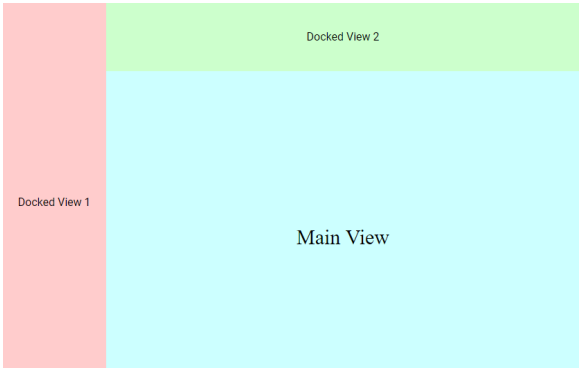
- Simple Buttons
- Drill Down in Views
- Navigation Drawers
- Single-Page Projects
- Back Buttons
- Navigating to External Websites

On this page

...

Configuring a Docked View

In setting up navigation, most projects will call for one or more **docked views**. These are views that sit flush with one of the four edges of the page.

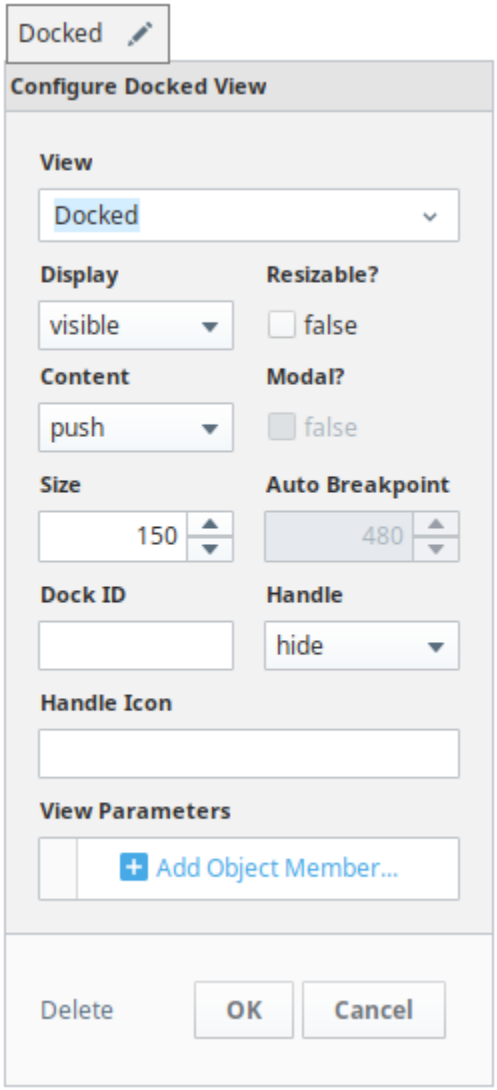


To dock a view you created:

1. Click the Settings icon in the Designer.
2. If you want your docked view to appear on all pages, select the **Shared settings** option in the menu, otherwise select the page you want the view to appear on.
3. Click one of the four **Add** icons, corresponding to where you want your view to be docked.

4. Select the path to your view in the **View** dropdown.
5. You may also want to configure other properties on this dialog. Click on the **Edit** icon next to the docked view name. You have many options, we'll just discuss a few:

a. Pay particular attention to the **Size** property, as this controls how far the view protrudes toward the center of your page.
b. If you want to show or hide the docked view from an event action or script, you'll need to specify a **Dock ID**. This can just be a keyword (like "menu") that you'll provide again later when you want to control this view.
c. **Handle Icon** can be used to provide a small icon for a user to show or hide your docked view at will.



Navigation Actions

Throughout Perspective development, you'll have the opportunity to set **Actions** in response to **Events**. Simply put, an event is something that happens on a component or session, and an action is how we respond to it. Many of the navigation options you'll have in Perspective depend on events and actions. Select the navigation action that works best for your needs:

- **Navigation** is the most self-evident. It allows us to open a new **Page** (either in our current browser tab or a new one), **View** (overriding our current *main* view), or **URL** (again, either in our browser tab or a new one).
- **Popup** allows us to open (or close) a popup view that hovers over our page's existing views. Configurable options include parameters to pass to the view, a title, where to open it and how big to make it, and some behavior options. The behavior options are straightforward apart from two:
    - **Modal** indicates whether the user should be allowed to interact with the page behind the popup. Essentially, should the user be able to move the popup to the side and continue working?
    - **Background dismissible** indicates whether the popup should be automatically closed if you click on the page area behind the popup.

- **Dock** allows you to open or close a docked view. Note that the view needs to be preconfigured as a docked view, complete with a **Dock Id** that the **Dock** action requires.
    - You can configure a **Dock Id** for a view by clicking the **Settings** icon in the Designer, adding the view using one of the four **Add** icons, then selecting the view to configure it.
- **Script** allows you to configure a script, from which you are free to call any of the built-in Perspective navigation functions. See system.perspective in the scripting section of the Appendix for details.

The Component Events and Actions section has more info about configuring events and actions.

Other Resources

The Common Perspective Tasks section provides a couple of examples specific to navigation:

- Creating Popup Views
- Horizontal Menu Component
- Self-Hiding Navigation Drawer

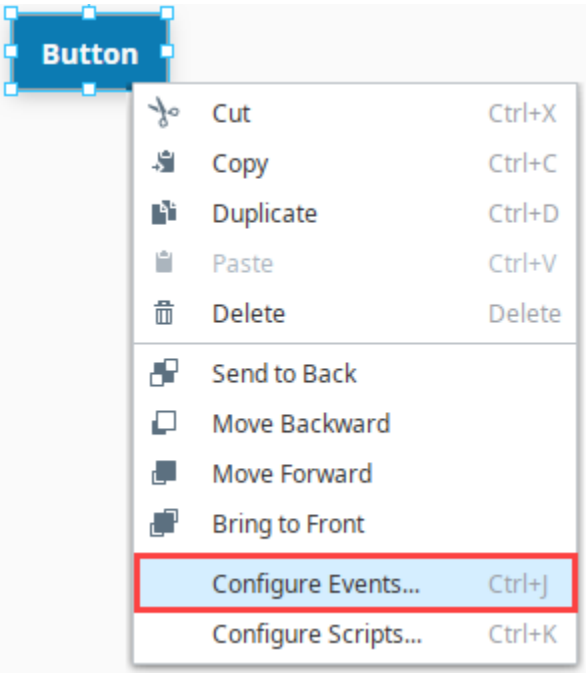
Basic Page Navigation

Buttons and Clickables

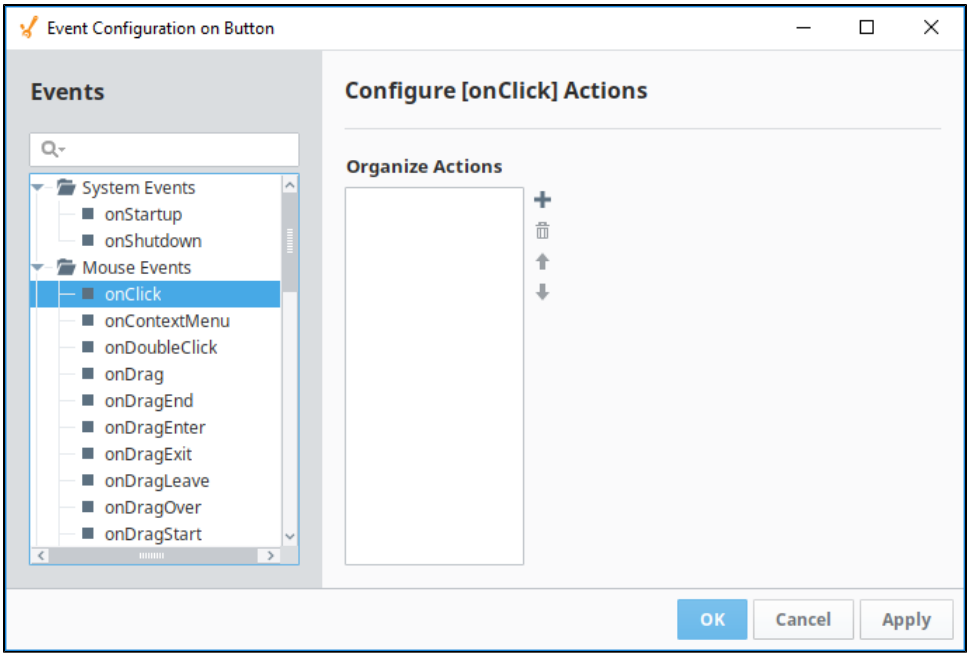
Before we get into fancier strategies, it's worth noting that we can use Button components (or any other components or containers you want users to click) to easily navigate from one page to another.You can use buttons on each main view, or create a docked view with buttons for each your main pages. This is one of the simplest ways to do navigation and will help you get a project started very quickly.

Let's walk through a quick example of how to make a button navigate when clicked:

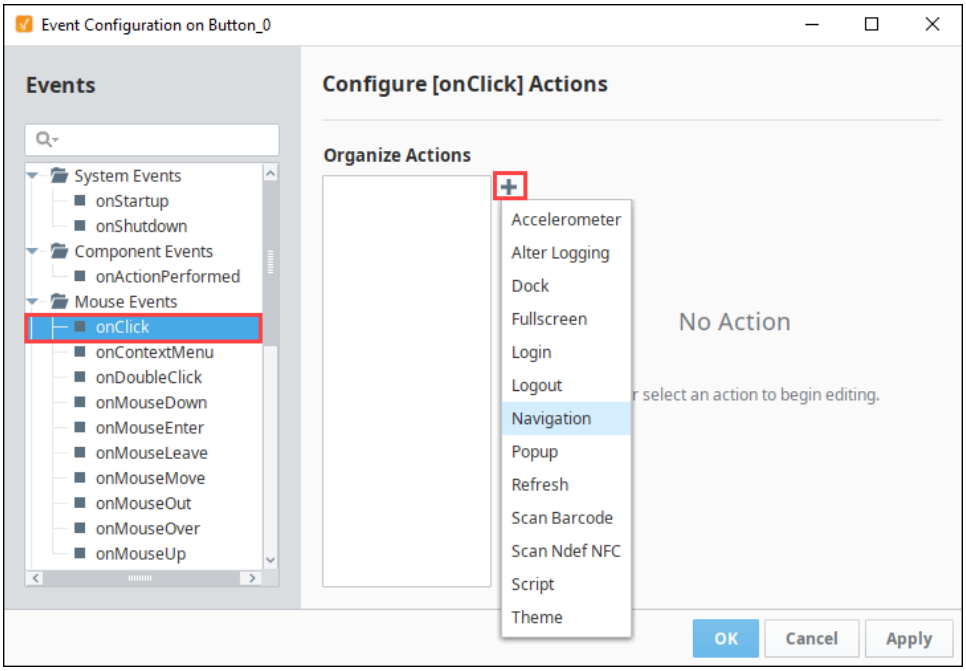
1. For a more consistent and hassle-free project, configure a **docked view** for your buttons. See **Configure a Docked View** above for a walkthrough.
2. Drag a Button component onto the container.
3. Right-click on the component and select **Configure Events...**.



4. Expand **Mouse Events**. As you can see, Perspective provides a multitude of ways to interact with your component. Select **onClick**.



5. Click the **Add** + icon to add a new action to the event. Select an action you'd like to use for the navigation. See the Navigation Actions section above for details about each action.



6. Repeat Steps 2 through 5 for each new Button that you want.

Tab Container and Menu Component

A Tab Strip or Menu is an effective upgrade to the basic button navigation strategy, particularly when you want an indicator that shows which page is selected.

There's a caveat to this approach, however; the tab *container* offered in Perspective cycles between views, not pages. This means that, if you'd like to organize your project into pages (and you probably should), the tab layout doesn't quite do what we need in terms of tabbed top-level navigation. There are of course other options, from using a row of buttons to using a Flex Repeater with a "tab" view.

> This feature is new in Ignition version **8.0.3**
> Click here to check out the other new features

The Horizontal Menu component enables you to build a menu structure by setting up multiple links to different page URLs from the component. An example is provided in Common Tasks in Perspective.

Navigation Tree

If you're looking for a pre-configured navigation option that is sleek and customizeable, the Menu Tree component allows you to create a menu that has expandable sections with buttons that users can click through. The Menu Tree works best inside a docked view (see **Configuring a Docked View** above), or, if you're feeling fancy, within a Navigation Drawer.

Only one property on a menu tree, the **items** property, controls the *structure* of the tree, all others control its look and feel. The **items** property can become fairly complex, but fundamentally each element in **items** has five sub-properties:

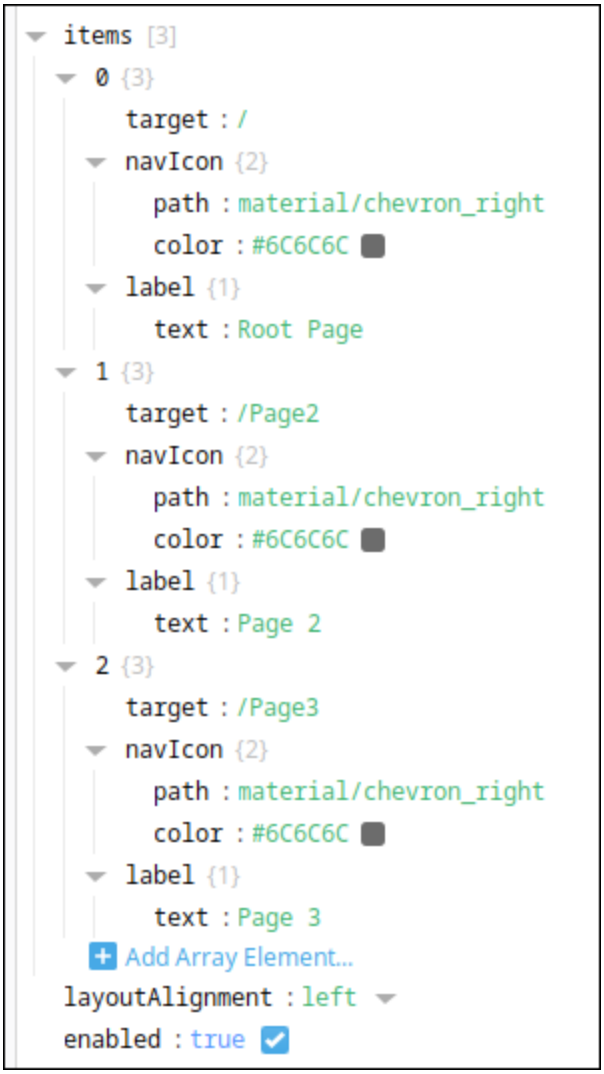
- **target** is where clicking on the item in the menu should take you. It can be a page URL, or an external (non-project) URL.
- **items** provides an opportunity to *show another menu* (instead of navigating) when this option in the menu is clicked. A *one-menu* configuration will ignore this sub-property.
- **navIcon** and **label** control the content of the menu item.
- **showHeader** controls what is at the top of a submenu when it is shown.

Simple Menu

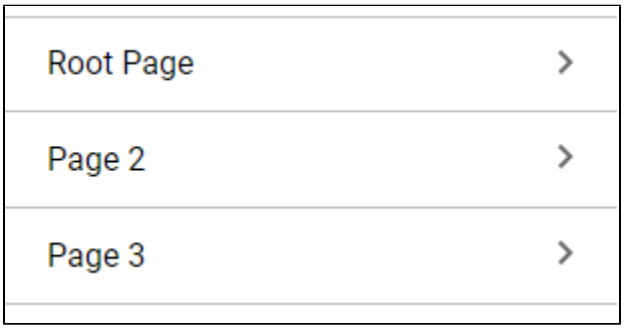
In a simple menu, you have a list of pages you'd like to navigate to, and you want them displayed in a column. The Menu Tree component can accommodate this without much fuss.

1. Drag the Menu Tree component into the view you'd like to use. Again, using a docked view is probably the best approach.
2. Click on the menu tree. We'll be walking through its properties in the Property Editor.
3. The **items** property on our menu tree will contain an entry for each page we'd like to navigate to. We won't walk through configuring those pages; here we'll just use the page URLs **/**, **/Page2**, and **/Page3**.
    a. The default menu tree has two entries, add a third one by right clicking on **0** or **1** (*not* **items**).
    b. Configure the **target** properties on the three entries to point to the appropriate page URLs.
    c. Configure the **label** property on each entry to show desired text.
        - You can customize the icon accompanying the label by changing the **icon** property (not to be confused with **navIcon**).
        - You can also delete the icon property if you'd prefer not to have one.
    d. The **navIcon** property can be customized or removed. It controls the right-most icon in the menu item.

    Here's how the final **items** structure should look. We deleted unnecessary properties for clarity.



Here's our new menu tree:

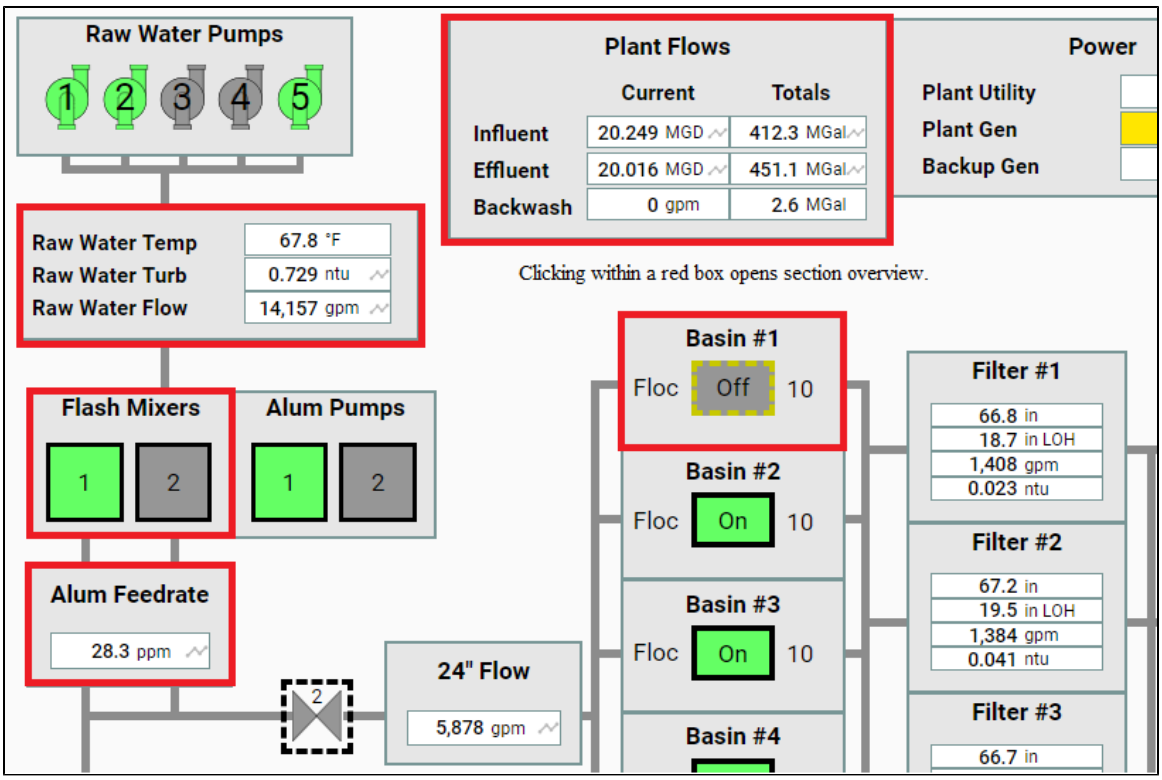


Drill-Down Navigation

"Drill down" (also known as *forward*) navigation strategies follow their namesake: they allow you to "drill" into a project page or view from an overview page. They also allow you to proceed to the *next* option in a chain. Most of the time, these strategies will feel quite natural: for example, an operator clicks on a line or machine on an overview screen, or they have finished scanning a barcode and want to record the data using a "Save" button. You could use a button press to complete tasks like this, but any component or container can be configured to handle mouse (and touch) events. In fact, you may wish to create a custom view and/or style for this purpose, since the way that you navigate will be closely tied in with the needs of your project.

Clicking on Components or Containers

Very similar to adding navigation to a Button component, simple navigation can be added upon clicking (or touching) any component:



1. Drag in an instance of a desired component, view, or container.
2. Right-click on the component and select **Configure Events...**. Expand **Mouse Events**. Select **onClick**.
3. Click the **Add** ✚ icon to add a new action to the event. Select the navigation option that works best for your needs. See Navigation Actions above for details.
4. Add your script to navigate to the new page. You will often be passing a value into a re-usable view that uses indirection to show a specific tank, motor, etc.

Same-Page Navigation

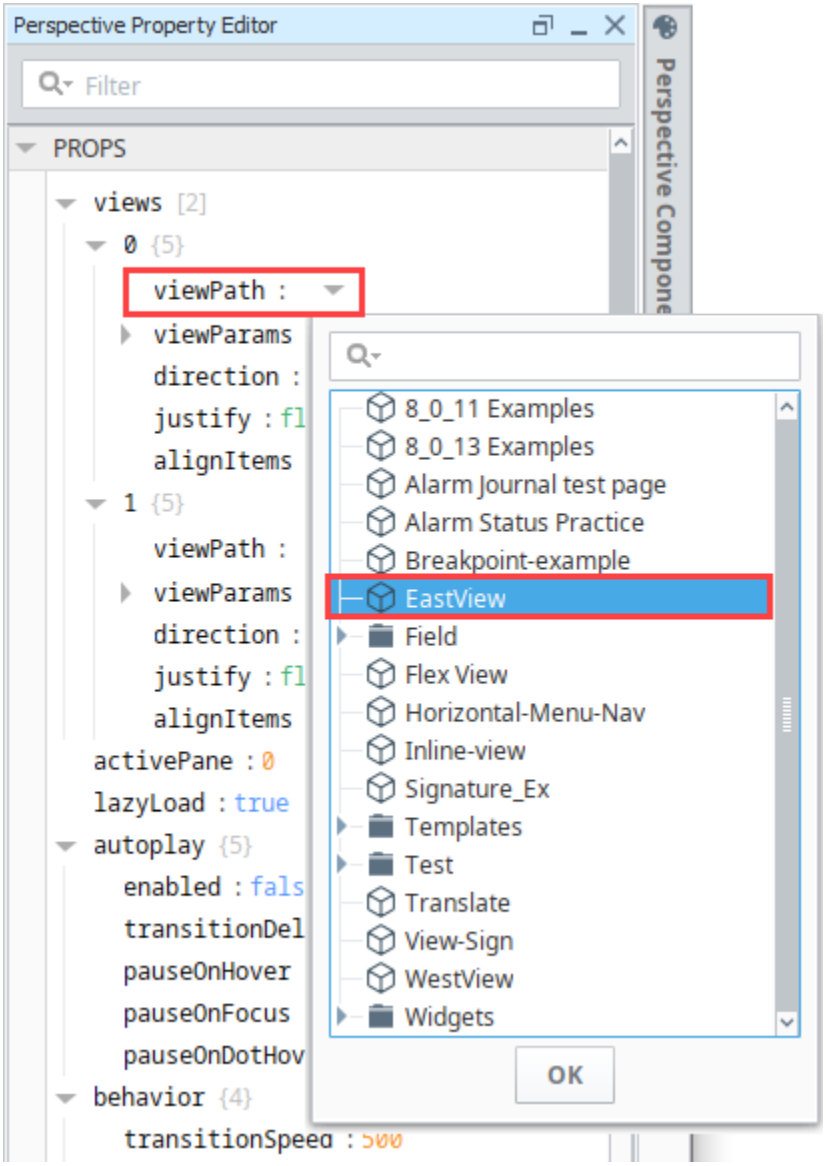
In this section we'll talk about a couple navigation strategies that explicitly *can't* take you to a new page of your project. They are good for navigating between views on the same page.

Side Scrolling / Carousel

Perspective offers strategies for dragging (or swiping) left and right as a way of navigating between views. The **Carousel** component is specifically for this strategy, which is perfect when working with several instances of the same view (including a dynamic number of them). Maybe your managers make a daily To Do list, and you'd like to scroll between them for different dates. Or you're looking for a way of collapsing a lot of content onto a small screen, and need a way of scrolling through it. Configuring side scrolling through a carousel is pretty straightforward:

1. Create the view you'd like to embed in a Carousel, and configure it as you'd like. It's probably a good idea to put something on the view that will distinguish it from other views on the Carousel (like a creation date, or distinct title).
2. Drag in a Carousel component, and position it.
3. Select the Carousel component. To add views to the carousel, click the **Add Array Element...** icon below the **views** property on the Carousel. Each created object has five properties, the most important of which are **viewPath** and **viewParams**. Select a view from

the dropdown, and configure any necessary view parameters in **viewParams**.

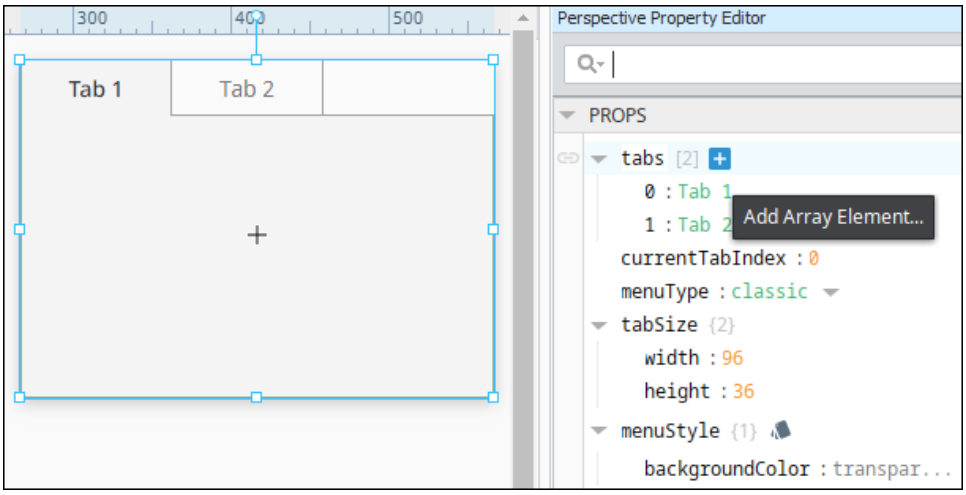


Tab Containers

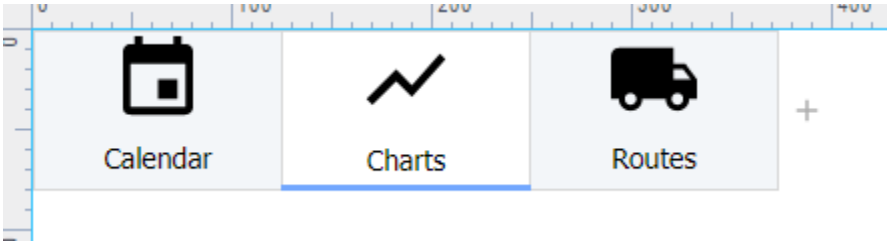
Tabs are an effective primary navigation strategy, particularly when you don't have many items to choose from. There's a caveat to this approach, however; the tab layout cycles between views, not pages. This means that, if you'd like to organize your project into pages (and you probably should), the tab layout doesn't quite do what we need in terms of tabbed top-level navigation.

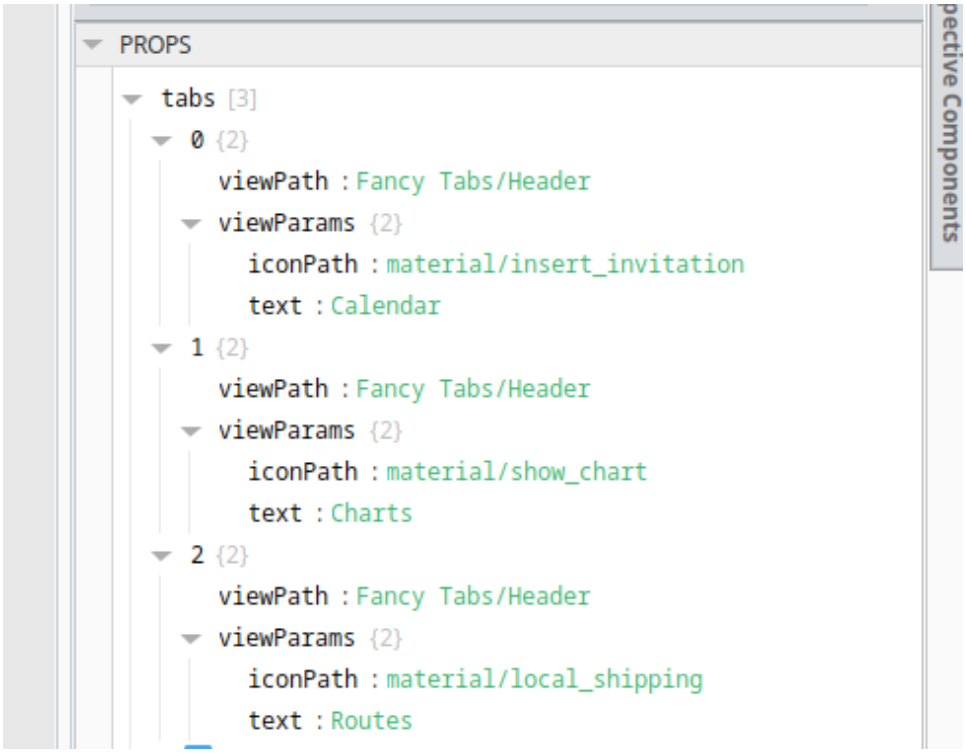
Tab Containers can be effective tools in designing complex single pages. In Perspective, they're easy to configure - you can use the Tab Container for simple drag-and-drop configuration; you don't even need to set up a docked view to control it. In these settings, the tab layout is perfect because it swaps between any views, containers, or components you provide. It's also easy to set up:

1. Drag a Tab Container into an existing view, and position it how you'd like.
2. Select the Tab Container. Add or remove tabs as needed by adjusting the **tabs** array. Note that if you're adding a tab, you'll want to click the **Add Array Element...** icon. From there, you have two options:
    a. Choose **Value** as the type. The string you enter will be displayed on the tab header.



b. If you feel like being extra fancy, you can nest a view of your choosing in the tab *header* (not to be confused with displaying a view in the tab itself). Choose **Object** as the type, then add a **Value** element to the new object. The element's key should be **viewPath**, and the value should be a path to a view to render in place of the typical tab header. You can include an additional **viewParams** object if parameters are needed. In the example below, we've added icons with the iconPath property.





3. Click on a Tab Header and drag in a view, container, or component to connect it to that tab.

Back Buttons

A Back button or reverse navigation generally refers to how a user might retrace their steps in an application, or move to a higher level page in the session. In Perspective, you will probably include "back" and "cancel" buttons and links in contexts where they seem appropriate (like navigating out of a view designed for a specific task). Since your project is being run in a browser, users will likely have access to the browser's back button, or a hardware back button on a phone. These buttons will typically navigate to the most recently visited *page* of your application. With this in mind, there are some good rules of thumb for developing browser-ready applications:

- Don't *depend* on the browser or phone's back buttons for project navigation. Many people aren't in the habit of using it.
- Don't assume your users *won't* use a back button. The back button is a valuable resource, and in most use cases, your project should gracefully handle navigation to any of its page URLs at any time.
- Break up your content into different pages when called for. If your entire project exists at the root page URL, an unfortunate use of the back button will leave a user outside their Perspective session, when maybe all they wanted was to return to a previous dialog.
- Don't make something a page if you don't want the back button to land you on it. For instance, if you have 50 PLCs and an operator needs to click through a page on each one, it may prove frustrating and disorienting to have to use the back button 50 times to navigate out of the list.

Navigating to Other Websites

Links

The **Link** component provides an easy navigation option when you want to invite users to view another network or internet resource from your project. The **target** property on the component dictates whether the page will open in the current tab or a new one. Tabs within the Horizontal Menu Component can also be set to navigate to a different website.

Related Topics ...

- Perspective Components