

# A Tutorial Guide to the Justinmind Prototyper 4

---



Last Revised: **September 20, 2010**

# Table of Contents

Preface .....	4
Tutorial for Beginners .....	4
Introduction .....	4
Getting started with the user interface .....	5
Menu Bar.....	5
Toolbar .....	6
User Interface Tab.....	6
Component Palette .....	6
Canvas or work area.....	6
List of screens.....	6
Selected Item Properties.....	6
Contents of the current screen.....	6
Progress tabs.....	6
Simulation button .....	7
Defining the contents of a screen.....	7
Importing external content to Prototyper .....	8
Interactive screen components .....	8
Clickable Wireframes .....	8
Interactive mockups.....	9
Share your prototype .....	10
Tutorial for Experienced Uses .....	11
Dynamic Content .....	11
Tabs .....	11
Dynamic Forms .....	12
Rich Interactions .....	12
Conditions .....	14
Variables.....	16
Templates.....	16
Masters .....	18
Widgets Libraries .....	19

Design Patterns .....	19
Interactive Images.....	19
Dynamically change the value of text .....	19
Tooltip .....	20
Custom Tooltips .....	20
Include a video from YouTube and Google Maps .....	20
Scroll bars.....	21
Conditional drop-downs .....	21
Jump between input fields using the Tab key .....	21
Tutorial for Advanced Users .....	22
Using the Dynamic Data Grid .....	22

# Preface

This document includes three tutorials: one for users who have not used the product, one for experienced users, and one for advanced users.

## Tutorial for Beginners

This tutorial is recommended for users who face the challenge of using Justinmind Prototyper for the first time.

The following sections explain how to use Justinmind Prototyper to draw and make clickable wireframes and interactive mockups. Finally, it describes how to export these screens to an interactive HTML prototype or in a document.

## Introduction

Justinmind Prototyper is a rapid prototyping tool designed to allow a user without programming skills to define how he/she wants a certain application. You can draw the screens of any type of application and enable interactivity. These interactions can be tested instantly with one click so you can test the changes immediately without writing a single line of code. These screens can be accompanied by text and diagrams and then generate a document that serves to clearly demonstrate an application.

The prototype can be exported to HTML format to simulate and provide feedback on the application that you plan to develop. In this way, you can create interactive prototypes and detail everything you want with the goal of capturing the feedback of the future users of this application and to avoid changes in critical phases of the project.

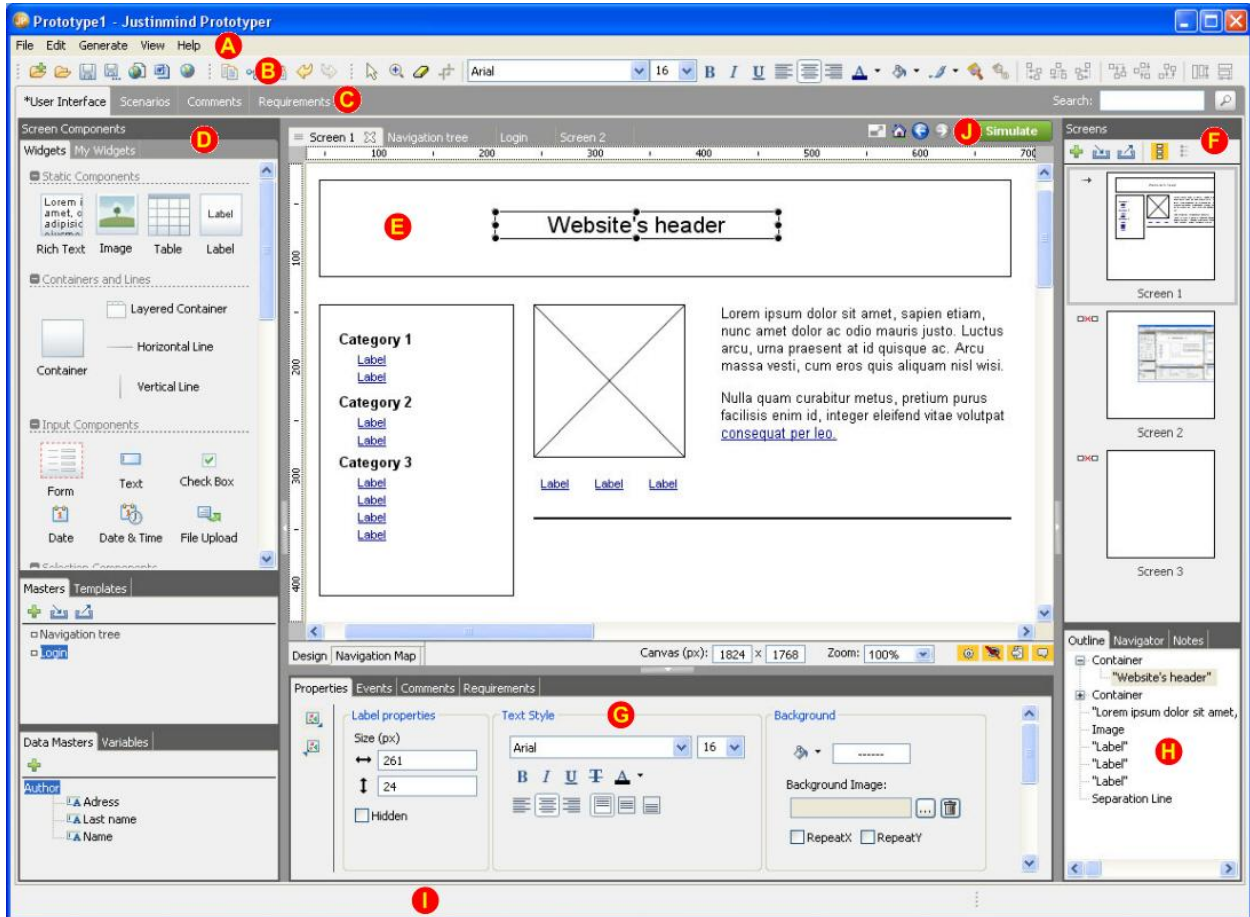
Prototypes made with Justinmind Prototype can be published using Justinmind Usernote. Usernote enables users to remotely test the usability of prototypes, take notes, provide feedback, and collaborate with you online. Collaborative development and rapid prototype refinement using Justinmind Usernote and Prototyper makes it easy to clarify requirements early in a project and to insure everyone rapidly agrees on the application's appearance and functionality.

This makes Justinmind Prototype with Usernote a unique and powerful tool for remote, online usability testing of prototypes throughout your project, saving you both time and money.

This tutorial teaches you how to use Justinmind Prototype to make wireframes, mockups, and interactive prototypes quickly and easily.

## Getting started with the user interface

The interface of Justinmind Prototyper makes it simple to do quick and easy creation of visual and interactive prototypes. It contains elements common to other drawing tools to make it even easier to use. We will now discuss the main sections in this tutorial.



**A:** Menu Bar, **B:** Toolbar, **C:** User Interface tab, **D:** Component Palette, **E:** Canvas or work area, **F:** List of screens, **G:** Properties of selected item, **H:** Content of the current screen, **I:** Status bar, **J:** Simulation button.

This tutorial only takes into account the sections described in the preceding paragraph, the remaining sections are explained in the tutorials for intermediate and advanced users.

### Menu Bar

The menu bar contains menus for performing tasks. The menus are organized by topic. For example, the menu contains commands to generate HTML, documents, or images.

## **Toolbar**

The toolbar contains options for quickly editing the component selected on the canvas. It also contains shortcuts to the most common options and editing tools.

## **User Interface Tab**

This tab includes everything that refers to the editing screens and interactions in the prototype. The other tabs are explained in other tutorials.

## **Component Palette**

The component palette contains all the elements that can be added to a screen. The components can be added to the canvas by dragging and dropping, or by clicking the mouse on the element and re-clicking on the position of the canvas as desired.

## **Canvas or work area**

The canvas shows the contents of the active screen and lets you add and modify components in the screen.

## **List of screens**

The screen list contains all the screens of the prototype under construction. To view the contents of a screen simply double-click on it in the list.

## **Selected Item Properties**

In this section you can edit the graphic style properties of the selected component. It also includes a section where you add comments to the same component. The remaining sections are explained in other tutorials.

## **Contents of the current screen**

This section lists the components that are part of the active screen on the canvas. Allows you to select and modify them individually as in the canvas.

## **Progress tabs**

The tabs are used to quickly change between the last opened screens. The selected tab indicates the active screen on the canvas.

## Simulation button

The simulation button allows to navigate and test the prototype as it would do an end user of the final application.

## Defining the contents of a screen

Drag and drop elements of the component palette to the canvas to create content on a screen. Once introduced into the canvas, the components can be resized and moved to organize within the screen. To move a component, simply select it and drag it to the area of the canvas you want. You can also move items with the arrow keys. Holding the shift key down while moving an item with the arrow keys will move the item faster.

Justinmind Prototyper includes an automatic alignment system using blue color guides that help you organize the contents of your screen. You can also apply a grid from the menu "View-> Grid." You can also create guides by clicking the mouse on the rules bars. To remove these guidelines you need to select them and press the "Delete" key.

The elements of a screen can be layered, so that some elements can be drawn on top of others. You can see the order of depth in the content view. To change the order of an element, click it with the right mouse button, or select that item in the content view and select the menu option "Order".

The image component allows you to indicate what areas will be occupied by images in your wireframe. You can substitute these representations by real images by selecting a file from the properties view. From this same view, you can also change the style properties (such as color or border) of the selected component.

To edit content on a different screen you have to double-click on the screen you want to edit from the list of screens. After double-clicking on the screen, the contents of this screen are loaded into the canvas. To create more screens in the prototype press the "+" above the screen list. To rename a screen, select it in the list, press the right mouse button and select "Edit" from the context menu. If you want to remove one of the screens in the select list, select it with right mouse button and select "Delete."

### **Important note: you can revert any action or remake it using the undo and redo options inside the "Edit" menu.**

A toolbar for the work area is located between the bottom of the canvas and the properties view. On one side there are two text fields that indicate the current resolution of the workspace. You can increase or decrease the resolution by directly editing these values. To the right of the canvas resolution is the zoom indicator. You can indicate the level of zoom through this drop-down menu.

You can also copy and paste parts of a screen on to the same or other screens using the Edit menu or keyboard shortcuts CTRL + C and CTRL + V. You can also duplicate screen elements by dragging while holding down the CTRL key or using the CTRL+D shortcut.

## Importing external content to Prototyper

Justinmind Prototyper provides many utilities to include external objects in your prototype. You can include text, images, and Web content such as videos, flash and Google maps.

To add an image from the file system just drag the image file to the canvas. You can also import the image from the properties view of an Image item dragged from the component palette. Another way to include images in a prototype is dragging and dropping directly from other graphical tools or even from Internet browsers. For example you can copy a selection from a Photoshop image and paste it in the prototype.

You can also import text from other applications by dragging and dropping or using the 'copy and paste'. Depending on the application, Justinmind Prototyper will import the text styles, if possible.

Displaying Internet content on a prototype is easy using the "Advanced Widgets" (they are located in the last group of widgets in the palette of components. For example, you can reserve a space in the active screen of the prototype to display a certain Internet address. To do this drag the URL component to the canvas and enter an Internet address in the properties view. You can also embed a video from YouTube or Google Maps by copying the HTML of these sites into the properties of the component HTML.

## Interactive screen components

Most components of the palette are interactive. In other words, once introduced into the canvas they can interact with the user pressing the button to simulate. One example is the input text component. If you drag this component to the canvas and click the button "Simulate" the simulation mode is activated. Similarly, all the components can be made interactive (if the user clicks on the text field he/she can enter a text as if it were a real application). Another interactive component is the drop-down menu. The values of that component are indicated in the properties view.

There are more interactive components, but two are special: the menu and the navigation tree. To create a menu, drag the menu component to the canvas and specify the size that will occupy the menu bar. You can then drag the menu items and place them in the desired position within the menu. The tree component works much like the menu, but in this case you have to drag tree nodes. The two components can be customized using the properties view.

All these components have implicit interactivity and they are activated by pressing the button "Simulate".

## Clickable Wireframes

If you have understood what was explained in previous paragraphs then you are ready to build easily clickable wireframes with Justinmind Prototyper. First, draw the contents of your wireframe using the figures of rectangle, text, label, and image. Then, mark the texts that are to

be clickable with a special style (typically blue and underlined). Finally, drag the items you want to navigate to a screen to the screen that is going to be displayed. Prototyper will automatically create a link. At this point, pressing the button "Simulate" will enable interaction with items on the screen and/or simulate navigation between different screens of the prototype.

Tip: if you want to indicate that only one piece of a rich text component is a link, place an "image map" over the area that occupies that piece of text and drag it onto the screen you want to navigate to. The "image maps" are not displayed when you press the button to simulate the interactions but remain in the area they are occupying.

As you define navigation between different screens on your prototype you can get lost on how the sitemap is being built. Justinmind Prototyper offers several options to browse the contents of a prototype. The first ones are the "browser" actions "home screen," "back", and "next" which simulate the behavior of a web browser and are located to the left of the button "Simulate ". At this point we should clarify the concept of "initial screen." In any prototype there is an initial screen that serves to introduce the user who is testing the prototype to other screens. This screen always comes first in the list of displays and is highlighted with an arrow. You can change the initial screen with the option to "Mark as initial screen" from the context menu from the list of screens.

Another way to see the structure of a prototype is to change the current screen view from the "Design" to "Navigation" view. To change to the mode "Navigation", press the "Navigation" tab located between the bottom of the canvas and the properties view at the left of the screen. In this view screens are displayed as a sitemap from the active screen. Clicking on the active screen displays the first level of navigation. Clicking on the screens at that level unfolds another level of navigation, indicating to which screens you can navigate from the selected screen. The navigation map is updated every time you create a link in the design view.

Menu items and the nodes of a tree can also navigate to other screens. All you have to do is drag onto the screen you want to display when you click on them. You can also drag the screen over the item and the effect is the same. In fact, any screen element can be dragged to a screen of the list and become clickable.

## **Interactive mockups**

Every process of defining an application includes a graphical design phase. At this stage the stakeholders suggest several styles and designs of the graphical interfaces of the application, although there are cases in which the design is marked by corporate style guides. The designs are usually performed with specialized graphic design tools such as Photoshop and usually the prototypes are just a collection of images made with that tools. The problem is that these images are static, i.e. you cannot interact with them, and can lose sight of the navigation between those screens. These proposals are often called graphic mockups.

Justinmind Prototyper offers a simple way to make these images interactive. First add the images of the screens in various prototype screens (remember you can drag image files directly or you can copy and paste content from other design programs.) Then drag the image map component on parts of the image representing buttons or anything that causes a change in screen when you press on it. Drag the image maps to the screens in the list you want to display. Do the same with the rest of screens and you can simulate navigation through these images.

You can also place form fields on the form fields drawn on the image, remove the border and adjust the size and font size so that that piece of image "comes alive" when you click the button to simulate.

The rest of tutorials explains how to define other types of interactions other than "click-> show another screen" and also teaches how to display dynamic content within the same screen. Remember that images can also be used in such cases.

## Share your prototype

The prototype is a vehicle of communication between different people to define an application. The closer the prototype is to the final application the clearer it would be to everyone involved in a project to know how that application would be when it's finished. As the prototype is a way of explaining how it will be an application, it's important to show the prototype to as many key people as possible within a project so they can give their opinion. The first objective of a prototype is to show how the application will be when it's finished in a way everybody can understand. The second objective is to detect changes in the definition before you start building the application. Justinmind Prototyper offers several ways to demonstrate and distribute the prototype with key people in a project.

The first way to demonstrate a prototype is using the Justinmind Prototyper simulation. This can be done using the Simulate (F5) and full screen (F11) buttons in the Prototyper application. This has the advantage of being able to apply changes on the spot and validate them immediately. On the downside, this also requires you to be onsite and to demonstrate in person.

Another way to provide a prototype is to export it to HTML format. This format can be opened from any Web browser, making it easy to distribute to key users. Once distributed, users can interactively try it out themselves, and send their opinions. The advantage of this system is that users can test the prototype and make contributions whenever they want. The disadvantage is that, first, we have to prepare the prototype and guide the user on what has to be done and, secondly, the user's spam filter will likely strip the attached HTML code.

There is still another way for users to review and test a prototype: Justinmind Usernote. This service allows you to publish a prototype accessible from the Internet and invite key users to see it using their Internet browsers. Users can also annotate on the prototype in a collaborative way, so all comments are centralized. Another option is to integrate this prototype accessible from the Internet with remote testing tools and perform user tests on the prototype.

You can also export all the information of the prototype in a specification document to be integrated in the different document processes of an organization or just to have a document that can be signed.

# Tutorial for Experienced Uses

This tutorial is aimed at users who have already used Justinmind Prototyper and seek insight into how to prototype with high interactivity. Those who have never used Justinmind Prototyper should first read the [Tutorial for Beginners](#).

## Dynamic Content

In this section we will explain in detail the capabilities of Justinmind Prototyper to simulate dynamic content in a screen. Dynamic content is content that changes in response to user actions, excluding things that cause the changing of screens.

The following illustrates a number of features and characteristics which you may have included in your application and which you may want to demonstrate interactively in your prototype. The following examples primarily leverage the dynamic panel component, but the concepts in these examples should be applicable to other dynamic elements of your prototype.

## Tabs

One of the ways used to organize large amounts of information on one screen is to place information in different "tabs". Each tab displays different content when the user clicks on it. The dynamic panel component is particularly useful for this case. This component allows you to group content into multiple layers and decide what is being viewed at any time.

To create tabs, drag the dynamic panel component to the canvas. Once the component is on the canvas it shows the area reserved for the currently active panel, the panel or layer name, and an icon of "+" to add more panels. Click the "+" icon repeatedly; once for each tab you want to simulate. Place in each panel the contents of the respective tabs by dragging components directly above the panel (note: to select a panel you must select the component of dynamic panels and then the gray label corresponding to the panel or layer you want to select). Then draw the tabs — in this scenario we will use labels and give them the form of tab.

Set the align properties of the labels center-aligned vertically and horizontally. After that indicate the text will have an edge and customize it by indicating the lower edge is not visible. Finally, we round off the edges a bit to give it a more tab-like look. Once the first tab is finished, you can copy and create another two. So far we have prepared the tabs on one side and the content you have to show in each tab.

Now we will indicate that pressing each tab will reveal the content corresponding to that tab. Select a tab and then select the events view next to the Properties view. This view shows all the interactions that can be defined between the user and the selected item when you run the simulation. If you click on the Add button, a dialog opens to specify what will happen when the user clicks on that tab. What we want to happen is to display the panel with the contents of that tab, so select the option Show / Hide, select the panel, and indicate that what we want to do is to show it (note: no need to indicate that other panels have to hide because only one panel can be visible at a time). Do the same with the other two tabs indicating the panels to be displayed.

Finally, if you click on the Simulate button you will see that content is displayed differently depending on the selected tab.

Tip: if you also want to highlight which tab is the currently selected tab, you can move the tabs into the panels and change the style from each panel.

## Dynamic Forms

In some cases, the data entry forms in an application can have many fields. There are several ways to organize the fields so as to not overload the user. One of those ways is to show only the fields necessary and give the user the option of displaying the fields that are optional. Here is how to simulate this type of form with Justinmind Prototyper.

We will use the same idea with the tabs, but more simplified. First we will draw some fields that we consider essential. Then drag the dynamic panel component to the canvas and create two panels or layers. In the first panel we will place a label titled "Show optional fields" and in the second panel another label that is "Hide optional fields" and below it add a few form input fields.

Define an event in the label "Show Optional Fields", so that clicking on that label displays the panel containing the optional fields. In this panel, define an event on the label "Hide optional fields" showing the other panel. It's that simple.

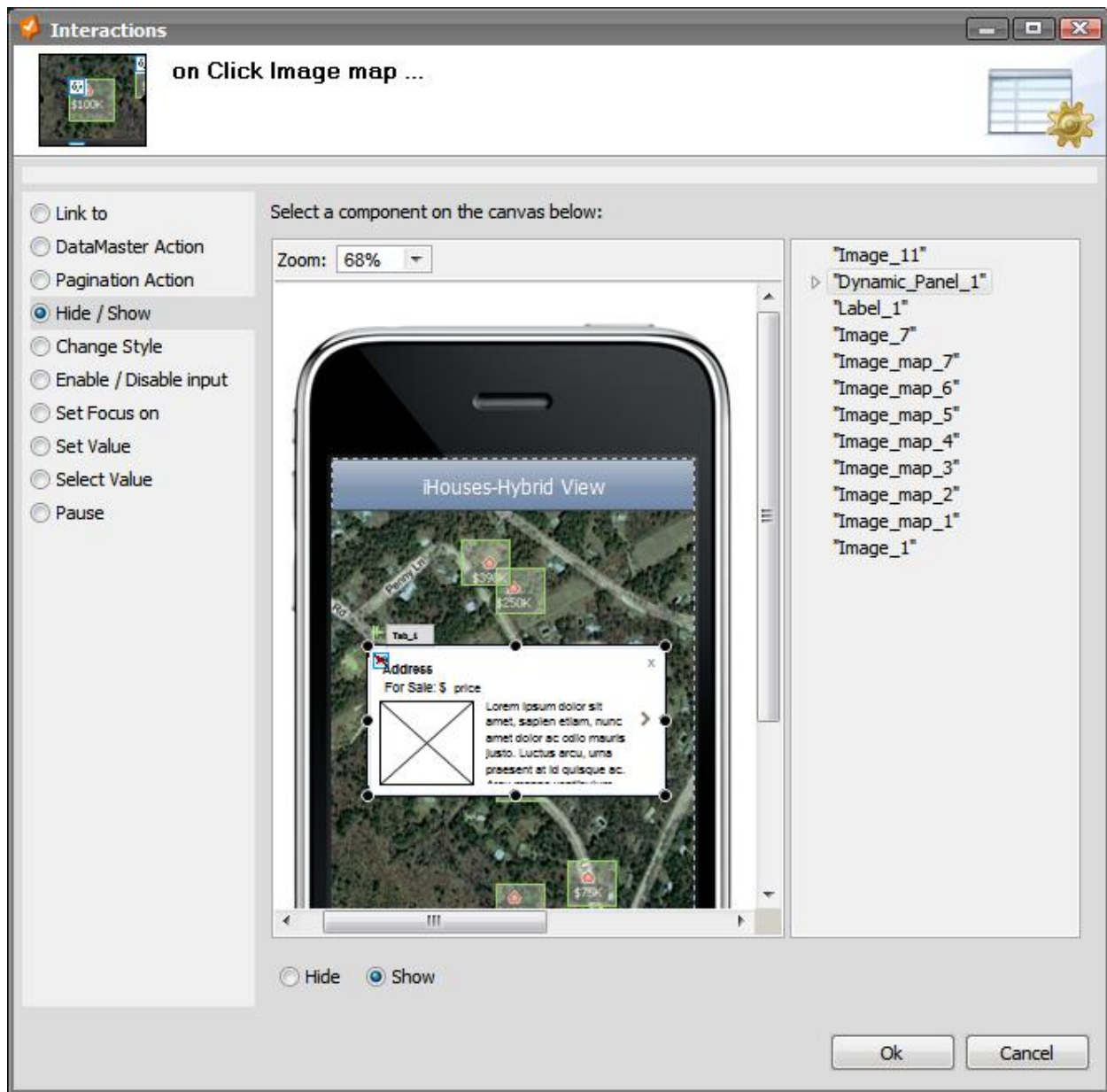
The [Tutorial for Advanced Users](#) explains how to make dynamic content that pushes the rest of the content using the layout property of dynamic panel.

## Rich Interactions

We have seen in this tutorial some of the interactions that can be simulated by Justinmind Prototyper. Now we will go into more detail on what specific events and actions can be simulated.

To define an event, use the Events tab next to the Properties tab in the view section at the bottom of the screen. To create an event, first you have to choose which user action will cause the behavior you want, and then to choose what you want to happen. To choose which user action, you have a user actions menu on the left. It is important to know that a component can react to more than one user actions, so you can define a button, for example, to react to both a button click and a hover action. The user actions menu also shows the number of actions that are defined for each trigger.

After selecting the user action, you can create another interaction by pressing the "Add interaction" button. When pressed, a dialog box appears in which we can configure what happens when the user performs the selected action. On the left side of this dialogue is a list of the different options we can simulate.



- **Link to:** Change the screen being viewed by the selected screen. You can also show an URL (Internet address). It can also be configured to display the new screen in a pop up.
- **Data Master Action:** These are actions related to the simulation of dynamic data lists and functional forms. These are explained in detail in the [Tutorial for Advanced Users](#).
- **Pagination Action:** Another action related to the simulation of dynamic data lists and is beyond the scope of this tutorial.
- **Hide / Show:** Hide or Show existing content in the current screen.
- **Change Style:** Change the properties of color, border, background, etc. a component of the current screen.

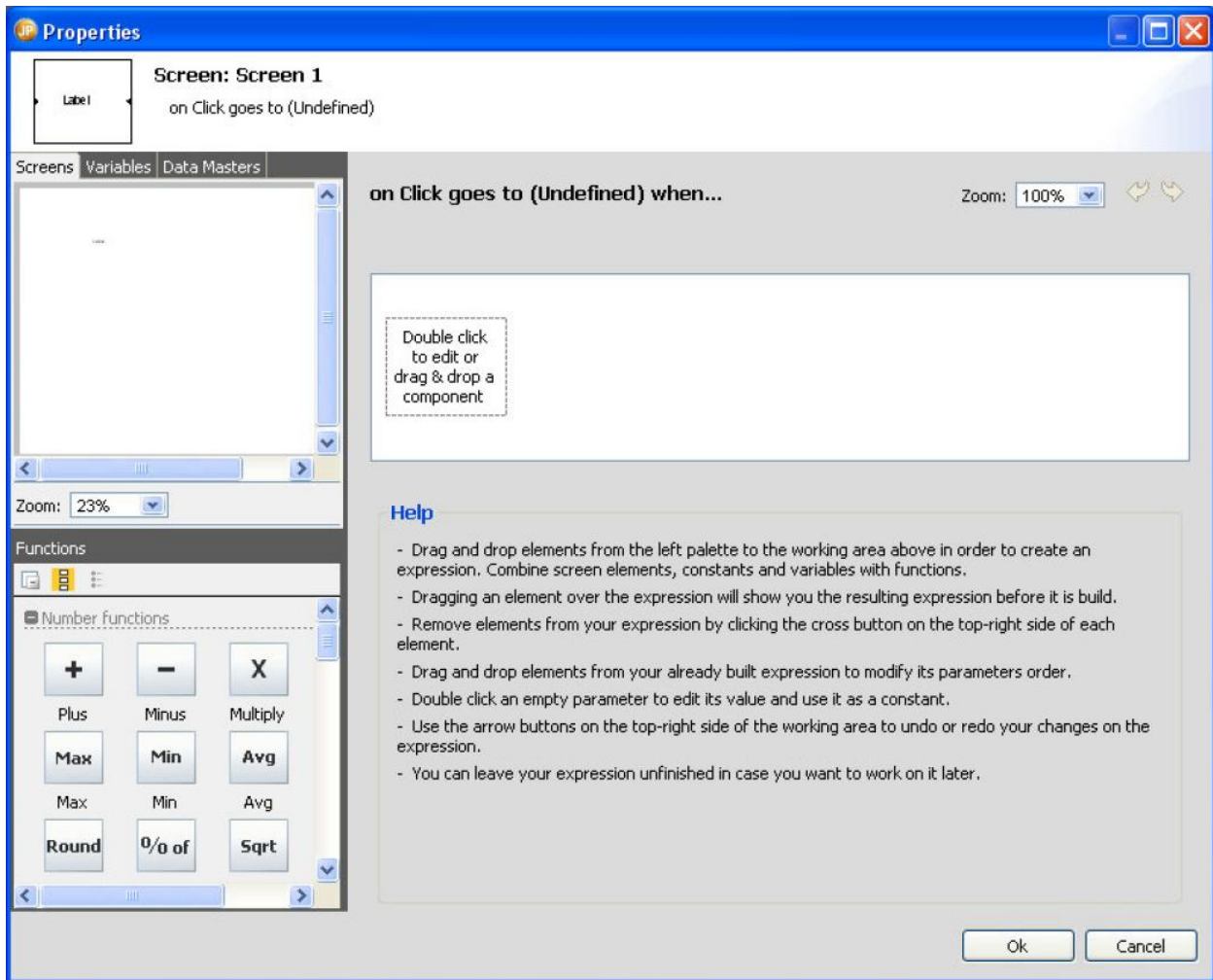
- **Enable / Disable input:** Toggle input fields of forms between being editable and being disabled (non-editable).
- **Set focus on:** The screen focuses on the selected item and activates it if it is a form input field.
- **Set value:** Change the value of a label, or variable form field by showing the user.
- **Select value:** Select the values entered by the user — only valid on form input fields of type selection (drop-down lists, checkboxes groups, etc).
- **Pause:** Wait the time specified before running any other option. This action can be combined with other actions to simulate animations, for example, a progress bar.

This dialog box also gives the option of adding a condition, but that part will be explained in the next section. Press "OK" after defining your action. The selected action will then appear in the Events tab. A single user action may trigger multiple actions in sequence. For example, we can add another action to run after the one just defined by pressing the arrow button next to the word "do." You can also add more actions using the button "Add interaction." The actions are executed from top to bottom and from left to right. We will return on this topic at the [Tutorial for Advanced Users](#) when discussing how to simulate animations.

## Conditions

Justinmind Prototyper can add conditions to the events to only run at certain times of the simulation. For example, you might want to simulate a form of "log in" to only allow browsing to the next screen if the user and password are correct and if not display an error message.

To create this form of "log in", first we draw it on a screen by dragging labels, form fields and buttons. Once drawn, drag the button "Log in" to the Welcome screen. A navigation event will be created automatically in the events tab. If you look above the navigate action you will see "Add condition". Clicking this option opens a dialog box to build special conditions.



The Conditions Properties dialog contains an editing area marked in white in which you can build the condition. The elements are divided into components and functions. Components can range from text form elements or variables of the screens, data fields master's degrees, or master data. All these components can be dragged and dropped into the holes in the editing area. Functions are operations that can be performed on those components to build the condition.

So following the example of the log in, build a condition to verify that the "user" equals "admin" and the "password" equals "1234." First drag the "&" function into the editing area. In the first hole of this function drag another function "=" and do the same with the other hole in the "&". Drag the field "user" of the screen to the first hole of the first function "=" (note: if you can't select the fields on the screen you can use the zoom feature.) On the second hole of the first function we write the word "admin". Drag the password field on the first hole of the second function and write in the last gap "1234." Press the "ok" button and you will see the condition in the events tab.

After creating the condition we see that beneath the navigation event has appeared an option "else" to indicate what happens if that condition is not met. Now we will simulate that an error message is display if the condition is not met. First, we must create the error message using the

component of "Label". Drag this component to the screen and write the error message you want. Mark the text component text as "hidden" using the Properties pane. Then select the log in button and click on the option "else" on the events tab. Select the option "Show / Hide", select the error text, and choose the option "Show." To end, press the button "ok", and press the button to simulate to verify that we have built the simulation of the log in form. You can make as complicated conditions as you like and you chain them in the form "if 'x' do 'and' if not"... The conditions can also be copied and pasted into other events including events from other components.

## Variables

The variables system can store data during the simulation of a prototype for use in other parts of the prototype. This is similar to the idea of "global variables" and is like containers that can store data on one screen and use it on another.

For example, we could create a registration form that leads us to a welcome screen and in this screen display the username you entered in the field "user." First, we draw out the registration form using components from the palette. Then, we will create a variable from the tab of "Variables" below the component palette. Drag the input field where you enter the user name to the variable indicating that when the user exits that screen what the user typed in this field of the input is saved in the variable. Then we will open another screen and insert two text labels. In one of the text labels write "Welcome" and drag the variable onto the other text. In this way we indicated that when the screen is loaded the text of the label is replaced by the value of the variable at that time. If you press the button to simulate from the first screen will see how to pass data from one screen to another.

Prototyper will automatically create an event when you drag the "user" field and drop it above the variable. If we look more closely at that event, we see that it is a "on Page unload", and what it does is to change the value of the variable for the content of this field. This is not the only way to initialize a variable; any action of "Set Value" associated with an event can change the value of a variable. To this end, the dialogue event must select the "Variables" button next to the Screens tab and select the variable that we want to change the value.

The variables can also be used as an element in the conditions. If we open the dialogue of expressions we see that, besides being able to drag items from the screen, you can also drag variables.

## Templates

The template system is a useful tool for rapid prototyping and, more importantly, to make changes globally. Templates are also used to define the content and interactions common to several screens in one place, and to define styles (colors, border types, fonts, etc..) that the elements will use as default in screens.

A "template" consists of a layout and content you desire to be consistent across a number of screens. Additionally, one or more areas are defined where content specific to each page will reside. You can also think of a "template" as being a "common background" shared between

screens, where each screen's content is superimposed on top of the contents of the shared template. The diagram below illustrates what we have just explained:



The image above shows a website where you can see that the header and side navigation bar are common to all screens; however, the core content of each screen varies. If we were prototyping this site using Justinmind Prototyper, it would be best to place the header and the navigation bar on a template and draw the core content in different screens of the prototype.

The tab "Templates" is located below the screen components palette and groups all the templates available in a prototype. By default, this group is composed of a single template. We can see its contents by double-clicking on the name of the template. The contents of the template will open in the workspace, just as if it were a normal screen of the prototype.

The contents of a template are defined in the same way as that of a screen; by dragging components onto the canvas. For example, you can drag an image of a logo to the top left of the template. If we open the screens, we see the logo appears in the same position in all of them. If we return to the template and make a link to that logo, we can also note that this link works on each screen, so not only is the content shared, but also the events.

A prototype can have several templates that are created by pressing the "+" tab of the templates. To indicate that a screen uses a particular template you have to click the screen with the right button and select "Edit." Then select the option "Template" and select the template you want to use for that screen.

Each template defines a default style for each type of screen component. If you drag a label into a template, the screen component may look different in that template than it does in another. These default styles are indicated by pressing the right button on the template and selecting "default styles." You can also specify the default style is the one from an already drawn element. To do this, simply select that item and click the option "Assign as default style" represented by the icon in the Item Properties tab. When pressed, the properties of the already drawn element you selected will become the default style applied thereafter to all the new components of the same type in that template. Just above this button is another button that does the reverse effect, i.e. changing the style properties of the current element to the default in the template.

Templates, like so many other components within Justinmind Prototyper can be exported and imported into other prototypes to make creating new prototypes faster.

Tip: How can one make tabs which change the active screen but also highlighted the tab referring to the current screen? Simple. First, all the tabs are drawn as if they were 'disabled' in the template and appropriate links are made. Then, on each screen, draw up the tab on that screen as a tab style 'selected'. All you need to do to verify the effect is as desired, is to press the 'Simulate' button.

## Masters

The masters are groups of screen elements that can be used throughout the prototype. They serve to propagate changes globally. Thus, changing the contents of a master, changes the contents on all those screens of the prototype where the master resides.

To create a new master, press the "+" button located on the Masters tab, just below the list of components and next to the Template tab. Once you press the button and enter a name the master opens on the canvas.

In the master, you can drag and drop any of the components available in the palette, just as you would when creating a screen. You can also define events, provided that they act only on components of the master. Once finished, the master is ready for use in different screens. To include a master on a screen, simply drag it from the list of masters in the area of the screen you want. Once in place you will see that the size of the master is set to its content.

The masters can be dragged to all screens, and even more than once on the same screen. Once you have placed the masters, if you want to make a change in any of the items contained in the master you have to open it by double-clicking on the list of masters or double-clicking on the master on the screen. Doing so, the content of the Master will open on the canvas and you can make changes. Once the changes are made the changes will be reflected on all screens where you had included the master. This way you can make global changes in common parts of the prototype with one change. There is also the option of making a master cease propagating changes to a particular screen. To do this, we select the master on the desired screen and open the context menu. Then select the option "Break master", so that it stops being a master and becomes a separate copy of the contents of the master. As in the case of templates, masters can also be exported and imported to share with other users or for use in other prototypes.

NOTE: When you delete a master in the master list, all the occurrences of that master are removed from all screens which included it. Deleting an occurrence of a master on a screen does not affect other occurrences of the master in the prototype.

This brings up the question of when one should use Masters and when one should use Templates. The templates are often used to define content which is common to multiple screens, always appearing in the same position and never on top of a screen element. Templates are usually for the headers, footers, and main navigation. The masters are also usually used to define content which is common to multiple screens, but the content can appear in different positions depending on the screen. Masters can also be used to define common content that can appear above the screen content, such as a drop-down navigation menu. Masters are often used to make boxes of links, search engines, or log in forms.

## Widgets Libraries

The widget libraries allow you to save groups of screen elements to use in other screens or other prototypes. They resemble the masters, but in this case a change in one component of the library does not affect the components that have already been inserted into the screen. They are ideal to capture style guides or to prototype some types of applications. For example, in the extras section at the Justinmind web site there are libraries for prototyping different types of interfaces for mobile devices.

The tab "My Widgets" located on the component palette lists all the components out of your current Justinmind Prototyper libraries. Import an existing library by pressing the Import button, located next to the button '+' in the tab "My Widgets", and select the .jpl file containing the elements of the library. The elements of this library will appear in the list of "My Widgets" and can be dragged to the various screens of the prototype. Once imported, a library will be available for all prototypes being done with Justinmind Prototyper.

## Design Patterns

The following examples describe how to create various design patterns that can be simulated using Justinmind Prototyper.

### Interactive Images

In this example, we will use the widget called "image map" to define events on areas of an image.

First, we include an image (in this case, the Justinmind logo), and then drag the widget "image map" over the letter "d" of the word "Justinmind" in the image that we included. We alter the size of image map so that it covers just that letter. Then we add a component rich text beside the image of the logo. Select the image map and create an event "on click hide the text next to the image". Press the button to simulate, and you should find that if you click on the letter "d" the text appears. This is one way to animate image areas simply and make them interactive.

### Dynamically change the value of text

In this case, we will simulate a text entity which when pressed allows us to edit it directly and have the value of the text replaced by the edited text.

First, we add a dynamic panel and then add a text component to the panel (note: when you drag a component onto a "dynamic panel", the component is inserted into the layer you are currently viewing). Then, select the dynamic panel and create a new layer. Insert a text input field and a button into this new layer. We now have all the elements on the screen, so let's create the interactions.

Select the button on layer 2 and click the option to "add interaction" in the events tab. Select the action show / hide, and indicate that we want to show layer 1 and click "ok." The event you just

created will appear in the events tab. Now press the arrow next to the word "do" in the events tab in order to show the options menu. Select the option "Create action" to allow us to concatenate other behaviors to the one we have defined. The dialog to choose the action will now pop up. Select the action to "change value", select layer 1 in the dynamic panel and within that layer selected text item. Then select the value by pressing "Calculate". A dialogue will be displayed to build calculated expressions. In the Screens tab, select the layer two in the dynamic panel and drag the text input element to the top in the dashed box. Press "ok", and return to dialogue events. Click "ok" again. We have just defined that when the user presses the button, what you typed in the text input element is going to be written in the text of layer 1. Finally, select the layer 1 and select the text item that we have inserted into it. Go to the events tab, and add an interaction for showing layer 2 when click on the text. If you press the button to simulate will see that, indeed, if the user clicks on the text, an input element with a button will appear that, when filled in and submitted, will replace the original text with the entered text.

## Tooltip

Creating the simulated displaying of a tooltip for a given element is as simple as selecting the item and indicating the text that will appear in the tooltip in the item's properties. If no text is defined on the property, no tooltip will appear.

## Custom Tooltips

In this case, we use the event "on mouse over" to simulate a custom tooltip. First, draw the contents of the tooltip; e.g. we could do it by bringing together a rectangle and a rich text component. Select the two and then select the option group and mark it as a "hidden" so that the group is not shown when you press the "Simulate" button. Then drag in any item, for example an image. Select the image, and click the option "on click" in the events tab. A list of user interactions will be displayed. Select the option to "on mouse over." Then, we add a new interaction that shows the group, and then press the "Simulate" button to verify that when the mouse cursor is over the image the group consisting of the rectangle and the text rich is shown.

## Include a video from YouTube and Google Maps

In this example, we will use the widget "HTML" in the widgets group called "web" to include a YouTube video and a map from Google Maps in a prototype. First, drag the widget to the screen. Then, open the YouTube page in your browser and right-click the video you want to insert. Select the context menu option to "Copy embed html". Next, return to Justinmind Prototyper, select the component and paste the HTML code from YouTube into the properties tab. Press the save button within that tab and note the video preview is now in your screen. Resize the HTML component to display all of the video and press the simulate button to test.

You can do the same with Google Maps, indicating a web address of Google maps by clicking the option "link" and copy the code in the HTML component as we did with the video on YouTube.

## Scroll bars

Scroll bars are a property of the layers of a dynamic panel. To make a scroll area simply drag a dynamic panel component into the properties tab and indicate where it is a horizontal or vertical scroll bar. When you enter the content within that layer, scroll bars are updated automatically.

## Conditional drop-downs

In this example, we will make one drop-down's content change depending on what the user has selected in another drop-down.

First, drag two drop-down menu elements to the screen. Select one of the two and click the Properties tab. Then click the pencil icon next to the drop-down "value". A screen will appear to allow us to edit the values of this drop-down. Change the default values by double-clicking on each of them and write names of countries. Press "ok", and select the events tab. Click on the icon "on click" and in the list of possible events, select the event "on Change". Then click the option "Add Interaction" and choose to "Set Value" in the action list on the left. Select the second drop-down menu and add values of cities of the countries in the other drop-downs (for example, if USA was country added in the first drop-down you can add a few cities like New York, DC and Chicago). Then click on the option of adding a condition, and define a condition so the drop-down will only have the cities related to the appropriate country. Drag the first drop-down (the countries) to the dotted box that says "Double-click to edit ....". Then drag the "=" located on the functions tab "General". Finally, we double-click the new box that has appeared and enter "USA". Press "Ok", and "ok" again. The description of the interaction we have just described should appear in the events tab. If we click on the text that says "else if ..." we can define the cases for the rest of the other countries as we did with the first. Finally, pressing the simulate button should allow you to select the country and have the appropriate cities appear in the cities drop-down.

## Jump between input fields using the Tab key

In forms with multiple input fields it is very common for the user to jump from field to field using the Tab key. This behavior is simulated automatically in the exported HTML from Justinmind Prototyper, and the order is defined by the depth of the elements on the screen. Now let's see how to define the tab order in a more specific way.

First, drag several text input fields to the screen. Then, select the first one and click on Events tab. Press where it says "click" with the mouse, and select the option "key pressed." Then, press the option "Add Interaction", which will display the screen to define actions. Unlike other cases, next to the button "Add condition" will be an input field to detail exactly which key will trigger the event. By default, any key pressed within the input will cause the action to trigger, but the Tab key is a control key. To specify the Tab key, we first press the pencil icon and then the Tab key. Then select the action of "Set focus on" and select another text input field. Click "OK" and press the button to simulate. If we write something in the first entry field and then press the Tab key, the cursor should jump to the next field.

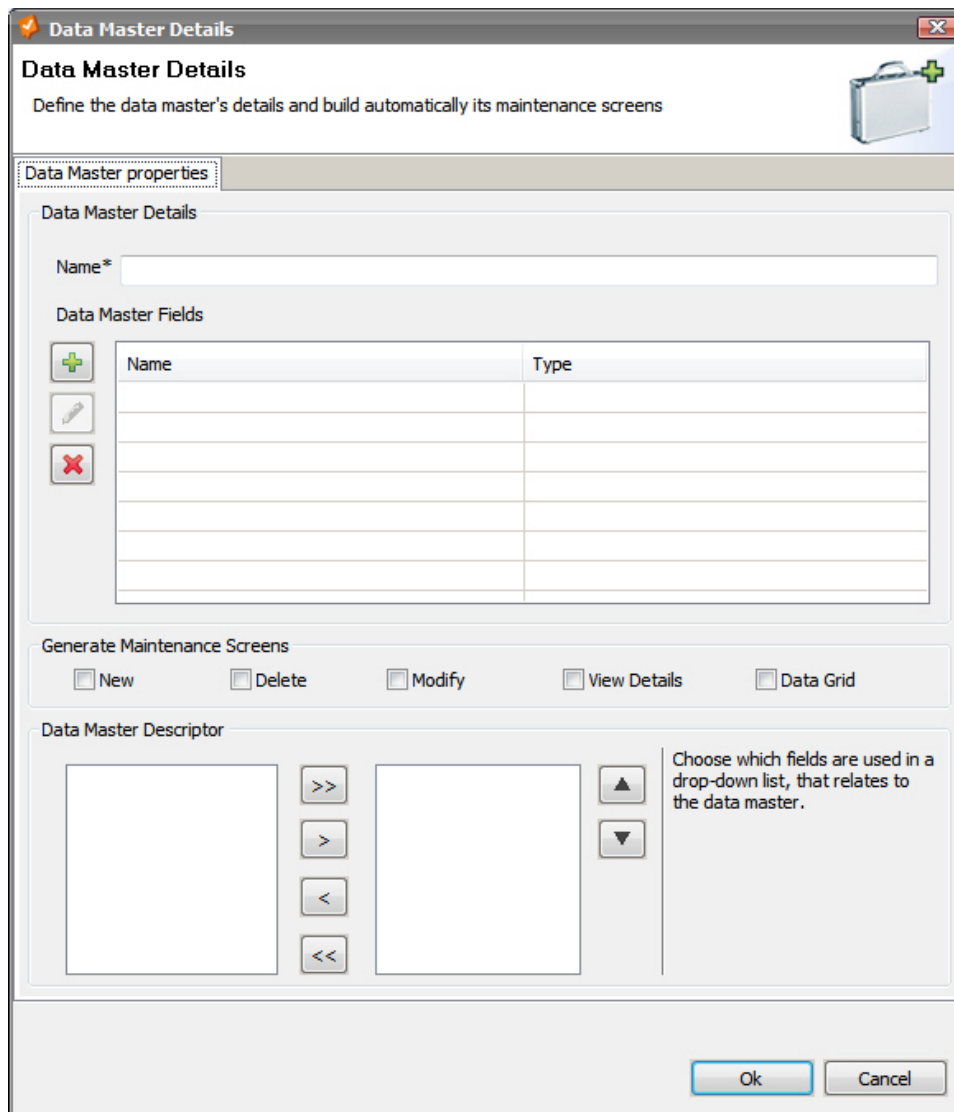
# Tutorial for Advanced Users

This section explores the various simulation options of Justinmind Prototyper when using data grids, dynamic lists, and forms populated with simulated real data.

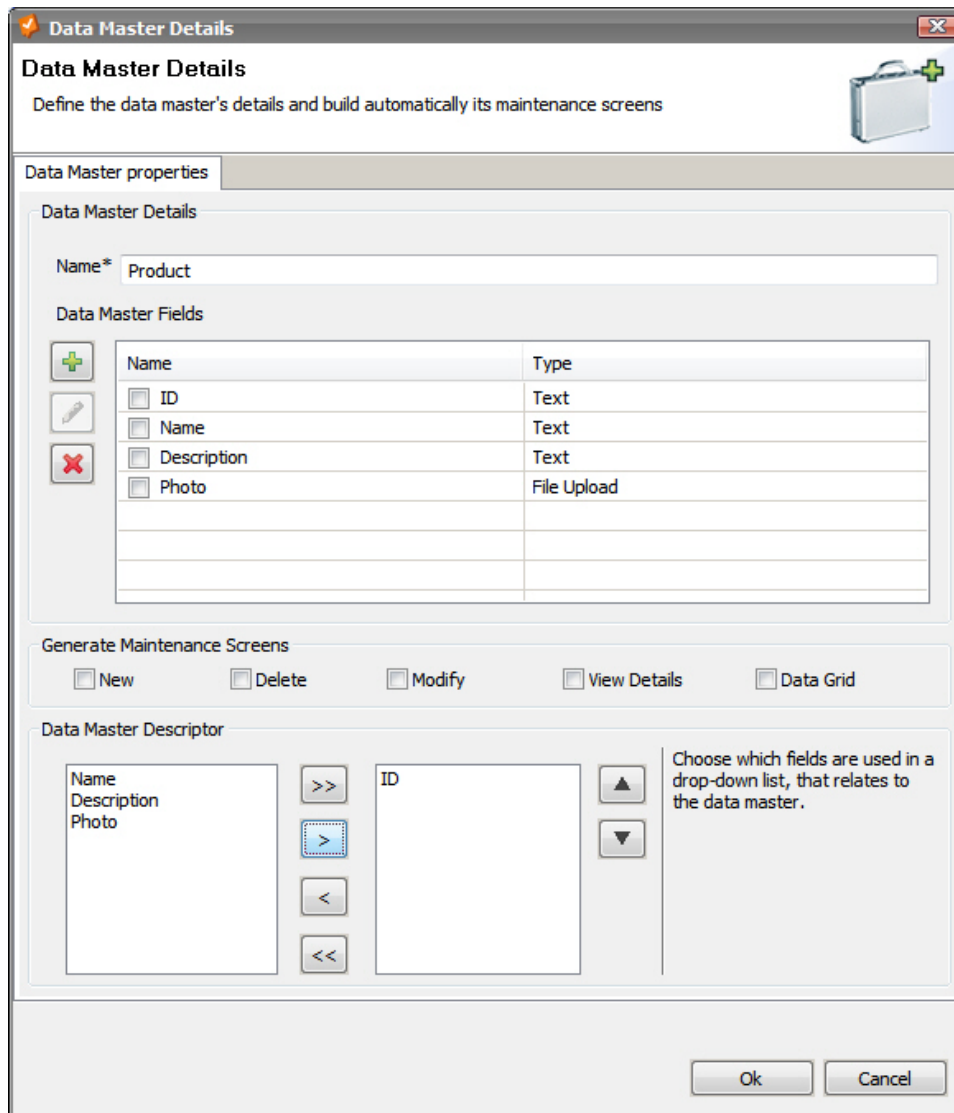
## Using the Dynamic Data Grid

In this chapter we will learn how to simulate structured content, either in the form of tables or lists, and populate that content dynamically. We will explain about data masters and the data grid component, such as adding search and pagination to these lists of data. We will also see how to prepare a collection of test data.

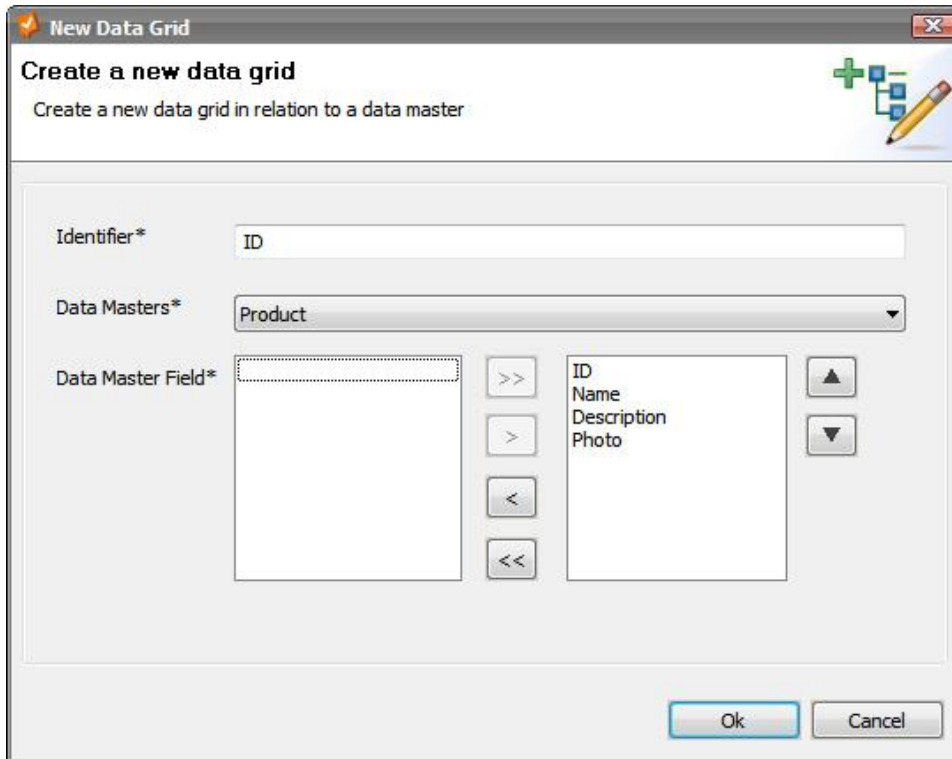
We start by creating a data master by pressing the '+' button in the Data Masters tab at the bottom left of the screen. The following dialog will appear:



In this example, we will simulate a list of products, so we will name the Data Master "Product". For each product in our product list, we want to show an identifier code, a name, a description, and a photo. We add each of these fields by pressing the "+" button just below where we have the Name of the data master. After adding each Data Master Fields, the Type will default to "Text." Click the type "Text" on the row that holds the attribute "Photo" and note the drop-down menu with several types available. Select the type "File Upload" because, as we shall see, the type affects how that information is represented in lists of data and forms. Click on "ID" and the ">" in the Data Master Descriptor. Finally, press the "OK" button.



The list in the Data Masters tab should now include the Product we just created. Now, click the Data Grid element in the list of components. A screen will appear to provide information about that table of data. Enter "ID" for the identifier, select the Data Master to be used as a data source (in our case we chose the only one, the Product), and select what attributes from that Data Master will be used as columns in the list, in our case we select all attributes.



Press OK and the program shows the size of that table. Position the outline, and click somewhere on the screen to set the location, and notice a data grid is drawn with a dummy row. If you press the button to simulate the table you will see three example rows.

ID	Name	Description	Photo
[Product.ID]	[Product.Name]	[Product.Description]	

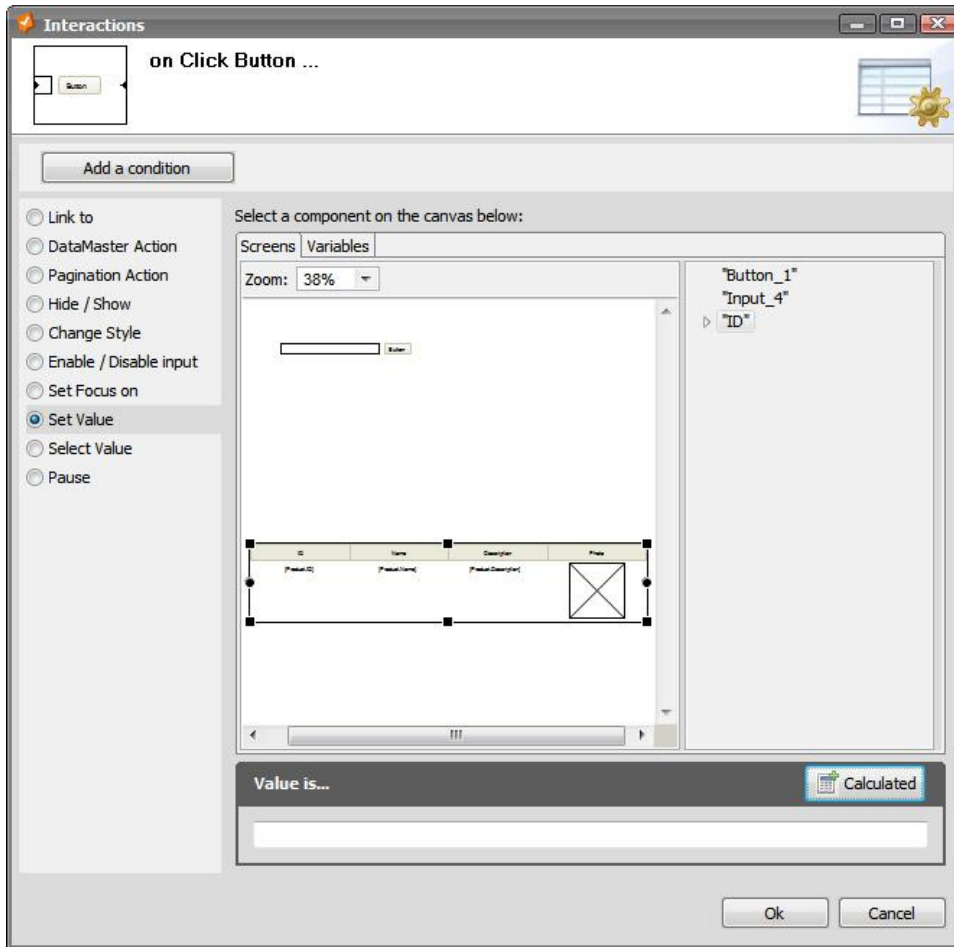
Let's change these example rows with more representative data. We double-click the product data master within the Data Masters tab and again we see the same screen as when we were creating it. However, now there is a new tab at the top that says "View and edit instances." Click it and you will see a table where each column corresponds to a data master attribute. You can define a set of test data that will show all data tables that use this data master. Click the cells of the table to change the dummy data.



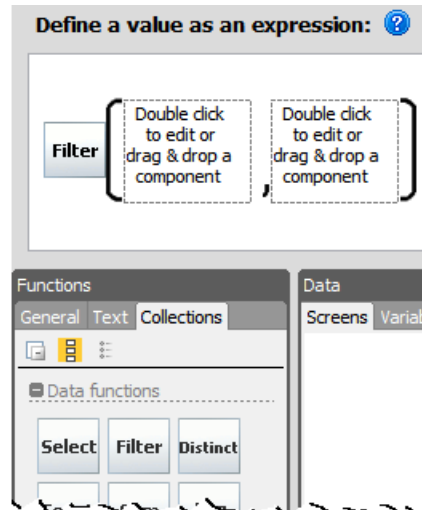
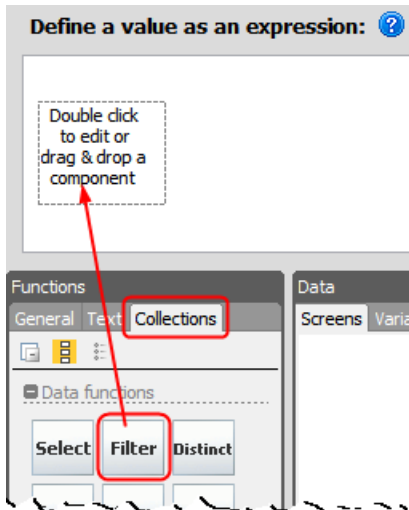
ID	Name	Description	Photo
id 1	john	sample 1	13 Verdana
id 2	jane	sample 2	18 Verdana
id 3	jim	sample text	18 Verdana

Data tables can be customized to form structured data lists. Continuing with the example, we could move the information from the table cells for product code, name, and description inside the cell showing the product photo, make that cell larger to fit all that information and delete the remaining columns. You can also select the whole list and define the color of even rows / odd.

Justinmind Prototyper contains an entire collection of functions to perform calculations on data masters and data tables. Let's use one of these functions to simulate a search box on the product list. First, draw the search box by adding a text input field and a button. Then press the button and select the events tab. Press the button "Add Interaction" and select the action "Set value." Select the data table (important, you should select the entire table, not a cell or a row, you'll know because the selection groups the row and the header) and click on the button "Calculated."

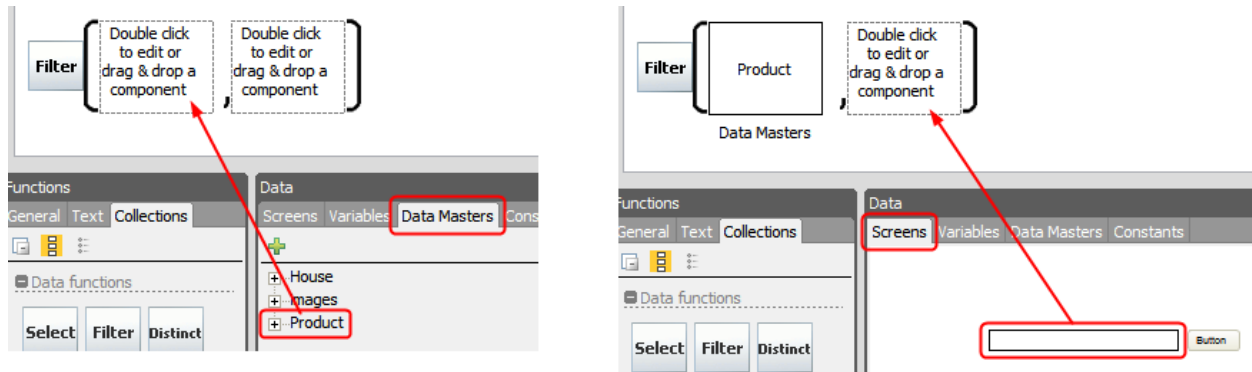


In the Expression Builder, click on the "Collections" Functions tab, and drag function "Filter" to the top.



Now we need to define the data source we want to filter. In this case, we want to do a search on all products. So, select the Data Master tab and drag the data master "Product" to the first

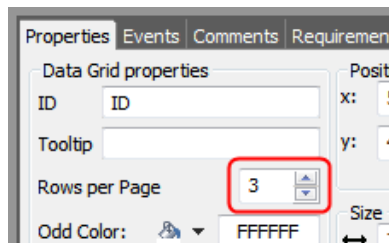
component slot. The second component slot is the rule that will use the filter function to decide whether a value is filtered or not. In this second slot drag in the text entry field located in Screens tab to the search button.



Press "OK" and "OK" again, and press the button to simulate. We now have a fully operational simulated search/filter function.

Sometimes the number of results in a table of data is so large that you would like to split into several pages. Here is how to simulate this type of paging using the components of "Index" and "Summary" and the list actions.

First, you have to prepare the data table to use paging functions. Select the data table of products we are using as an example and click the Properties tab. In the property "Rows per page" indicate the maximum number of rows to display per page (0 means it will not limit the number of rows and therefore not using the paging).



This example will show only three rows per page. Drag a component of "Index" to the screen and link it with the data table using the property "Data Grid" in the properties tab. The index automatically shows the number of pages that will have the list based on the number of rows per page. If your table has 1 to 3 rows, then the index will be "1". If it has 4 to 6 rows, then the index will be "1 2". This index is dynamic so the search simulation also updates the index.

The "Summary" is another dynamic text, and similar to the index it is related to a data table. It is configured like the index, dragging the component and selecting the property data grid in the properties tab. This text shows the number of results that are in total on all listing pages and the values that are being displayed at that time. For example with 3 items and a Rows per page of 2, it would say "3 items found, display 1 to 2".

Finally, we simulate the actions to see "next page", "previous page", "last page", and "first page". First, drag four buttons, and label them as text "next", "previous", "first", and "last." Each one is going to run a different paging action on the data table. Let's do an example with the

action "see next page", but you can apply the same method to the other buttons. Select the button "next" and press the tab events. Press the button to "Add Interaction" and select the option "page action." Press the button that says "next page" and open the Expression Builder that we saw earlier to calculate automatic conditions. Drag the list on the screen in the hole of the expression and press "OK". We return to click "OK" in the dialogue of events and test that we really are changing the values of the list each time you press the button "next". You should create a few values in the data master for a better view of the paging, summary, and index simulation.

To add a column to a table of data, select the table, click the right mouse button, and choose the option "New" column. A new empty column will appear to the right of the list. To indicate that some data master attribute has to display in this new column you only have to drag the attribute to the cell. You can also change the order of the columns simply by selecting them from the top of the header (the cursor changes to an arrow pointing down) and moving horizontally on the list.

This is the end of the tutorials. You are now ready to start using Justinmind Prototyper for your prototyping projects.