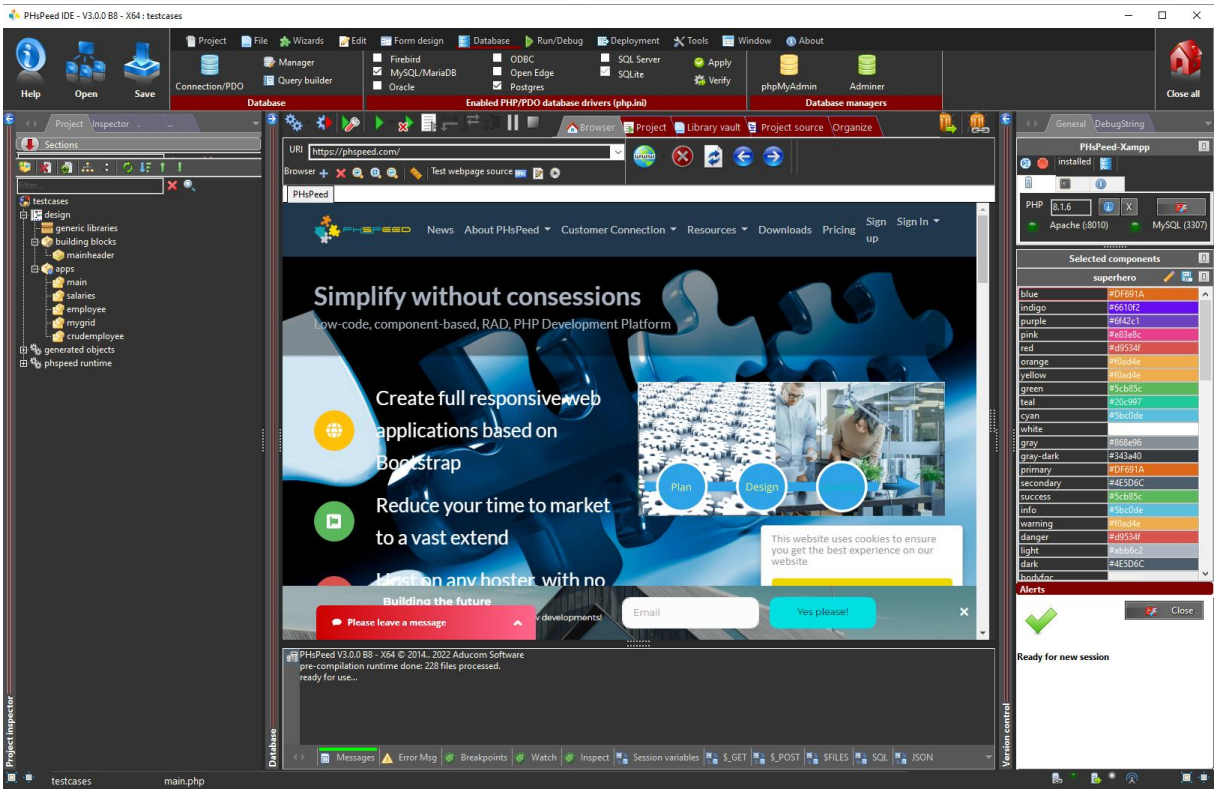
PHsPeed Introduction

Layout your forms (Bootstrap rules)

5

PHsPeed: Getting Started



Welcome to the world of PHsPeed! This manual is designed to introduce you to our low-code PHP development tool and help you work faster and smarter, aligning with the modern approach to web application development.

In the previous documents, we discussed the fundamental features of the Integrated Development Environment (IDE), created the first basic PHsPeed project demonstrating the 'hello world' example, and showed you how to leverage the application wizards to expedite your development process. You also know by now how to create a form manually, and wrote your first PHP code in an ajax event.

In this manual, we will delve into the process of designing forms and ensuring their responsiveness using PHsPeed. The first important concept to understand is that PHsPeed utilizes the Bootstrap grid system. This grid system is designed in a table-like manner, featuring a variable number of rows with columns. These rows and columns are contained within a designated container, which can be exemplified by using a Panel component.

When you add a Panel component to your form, you have the ability to include rows within it. Each row has the flexibility to accommodate a maximum of 12 columns. The key to achieving responsiveness lies in specifying how these columns should behave and break into new rows based on the device on which the form is being displayed.

In summary, PHsPeed leverages the Bootstrap grid system, allowing you to create responsive forms by organizing content into rows and columns within containers like the Panel component. By defining column behaviors, you can ensure optimal form display across various devices.

Bootstrap Grid System

The Bootstrap Grid System plays a crucial role in creating responsive layouts. To fully comprehend

Bootstrap, it is essential to grasp how this system functions. The Grid is essentially composed of arrangements of Rows and Columns contained within one or more Containers.

The rules governing the proper use of the Grid are as follows:

- Columns must always be the direct child elements of a Row.
- Rows are solely meant to contain Columns and should not have any other elements within them.
- Rows should be placed within a Container (Like the cPanel component).

These rules are of utmost importance. The seamless interaction between Rows and Columns is fundamental to the Grid's functionality, and it is imperative never to have one without the other.

Failing to adhere to these three simple Grid rules can lead to undesirable consequences. It is essential to strictly follow these guidelines to avoid potential issues and ensure the effective operation of the Bootstrap Grid System.

The rule of 12

The logical question is, why did Bootstrap choose 12? The main reasons for choosing 12 columns are as follows:

Divisibility: 12 is a highly divisible number. It can be evenly divided by 1, 2, 3, 4, 6, and 12. This allows for greater flexibility in creating various layouts and arrangements within the grid system. Different column configurations can be easily achieved by dividing 12 into different fractions.

Easy Percentage Calculation: Since 12 can be divided by 100% (for example, 12 columns of 8.33% each), it makes percentage-based calculations easier for responsive design. This is important when creating layouts that adapt to different screen sizes and resolutions.

Simple Fraction Representation: When dividing the 12 columns, the fractions are relatively straightforward to work with. For instance, 1/2, 1/3, 1/4, 1/6, and 1/12 can be easily represented in whole numbers or simple decimals.

Standardization: By using a common grid system like the 12-column layout, designers and developers can work with standardized guidelines and easily collaborate. It provides a consistent framework for designing and coding layouts, making it easier for teams to understand and build upon each other's work.

While the use of 12 columns is a convention, it's worth noting that you can change these settings in the original bootstrap CSS. But it is highly discouraged.

Device context

The secret of a good responsive application is to apply the rule of 12 into the different types of displays. Bootstrap 4[1] uses 5 different device classes:

[1] Bootstrap 5 is under development and supports 6 different device classes.

Device class

Property	Value	
deviceclassdesktopxl	3	Extra large >= 1200px
deviceclassdesktop	3	Large > 992px
deviceclasstablet	6	Medium > 768px
deviceclassphone	6	Small > 576px
deviceclassmobile	12	Extra small < 576px

As an example, you have a row with two columns and have set the device class "tablet" to 6. This means that on screens larger than 768 pixels (considered as tablets or larger devices), the two-column layout will be displayed. Each column will take up half of the available width (6 out of the 12 columns). All the components within these columns will automatically adapt to the available size, adjusting their width accordingly.

However, when the screen size becomes smaller than 768 pixels (considered as smaller devices such as mobile phones), the two columns will be displayed as two rows. Each column will now occupy the full width of its row, one below the other. This is known as a stacked or "vertical" layout for smaller screens, which makes the content more user-friendly on mobile devices.

The responsive behavior you described is achieved by using Bootstrap's grid classes, such as "col-sm-6" (for screens larger than 768 pixels) and "col-xs-12" (for screens smaller than 768 pixels). The "col-sm-6" class specifies that the column should take up 6 out of 12 columns on small and larger screens, while "col-xs-12" ensures that the column takes up the full width on extra-small screens (less than 768 pixels).

To clarify, setting the device class "tablet" is not a standard feature in Bootstrap. Instead, it requires responsive grid classes like "col-sm-6" and "col-xs-12" to define the layout behavior for different devices. Bootstrap automatically applies the appropriate classes based on the screen size, making it easy to create responsive designs for all devices without the need to specify each device individually. PHsPeed will generate these classes, based upon the device class settings.

Padding and Margin.

In Bootstrap, components are designed to be "Bootstrap aware," meaning they are built to work seamlessly with the Bootstrap framework. These components typically come with properties that allow you to control padding and margin, which help define the outer and inner spacing of the component.

- **Padding**
  Padding is the space between the content of the component and its border. It creates an inner space around the content, separating it from the border of the component. Padding can be specified for each side of the component (top, right, bottom, left), or you can set a uniform padding for all sides. This helps to control the spacing inside the component and achieve a visually appealing layout.

- **Margin**
  Margin is the space outside the component, separating it from other elements on the page. It defines the gap between the component and its neighboring elements. Like padding, margin can also be specified for each side of the component or set uniformly for all sides.

By adjusting the padding and margin properties, you can control the spacing and positioning of Bootstrap components effectively. This ensures that your layout looks well-organized and consistent across different screen sizes and devices.

PHsPeed: Best Practices for Applying Padding and Margins

In PHsPeed, it is recommended to adhere to specific best practices when applying padding and margins to create well-structured and visually appealing forms. To achieve optimal results, follow the guidelines outlined below:

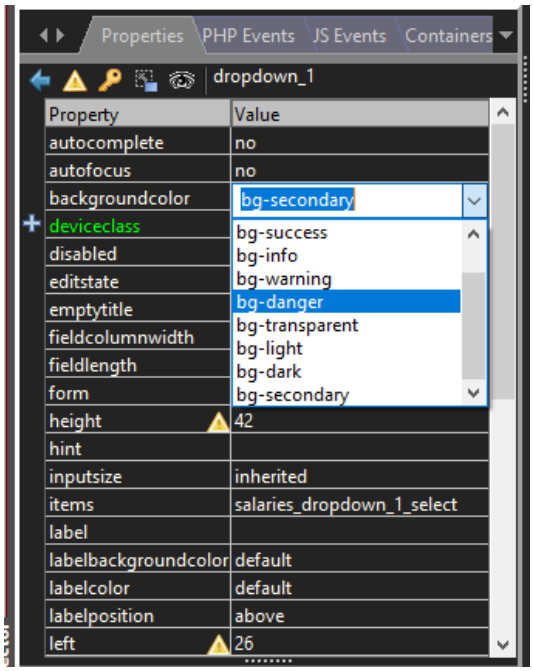
- **Apply Padding and Margins to Rows and Columns:**
  - Emphasize applying padding and margins primarily to rows and columns in your form layout.
  - Doing so ensures consistent spacing and alignment for multiple form elements within a container.
  - Utilizing the grid system effectively helps maintain the integrity of your form's structure.

- **Avoid Direct Application to Fields:**
  - Refrain from directly applying padding and margins to individual fields, such as edit fields and labels.
  - Applying these properties directly to fields can lead to additional spacing between elements, potentially causing the form to appear wider than desired.
  - This approach might result in an uneven and unbalanced layout.

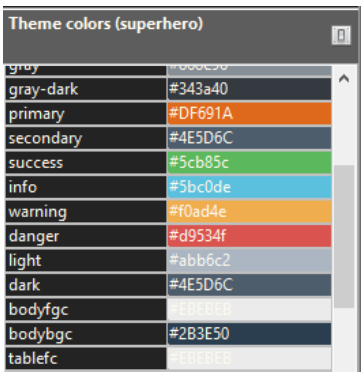
By adhering to these best practices in PHsPeed, you can achieve harmonious and consistent form designs, mitigating any unintentional stretching or distortion due to excessive spacing between form elements. Following these guidelines will result in visually cohesive and user-friendly forms that adapt well to various devices and screen sizes.

Other layout properties.

Besides the padding and margin, components can have other properties to maintain the layout. In the most simple form, you can apply colors to the edit font, background, labels, set or unset borders etc. These properties can be accessed using the property editor. If you want to change the font color of an edit field, select it with the mouse, and the property editor will show the properties.



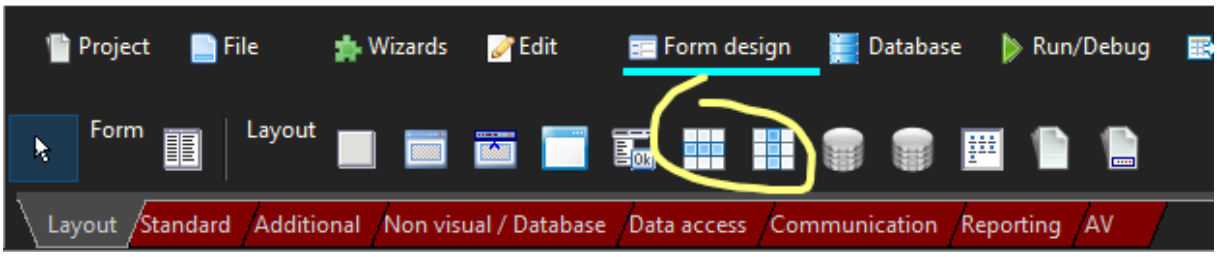
Instead of using color hex codes, like #cc0000, PHsPeed uses the Bootstrap colors. Success, info, warning, etc. are *symbolic* colors that translate to a certain color that is defined by the choice of the Bootstrap theme. Depending on the chosen theme, Bootstrap will use the colors of the theme. Note: some components are based upon JQueryUI. They do not make use of the Bootstrap colors, but still



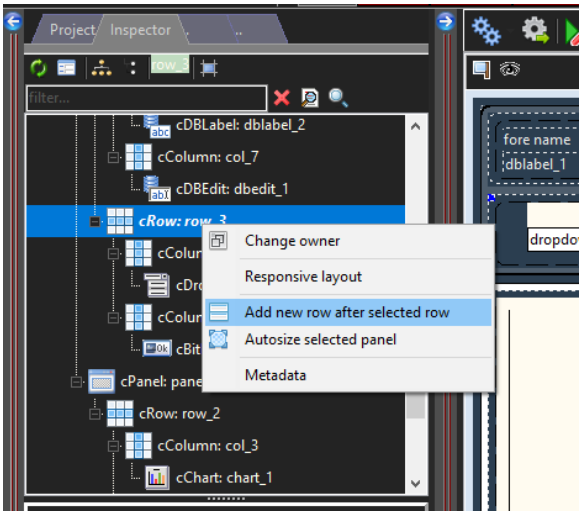
require the hex code. In the current PHsPeed version this is rare. The symbolic colors for the selected theme are shown in the IDE right bottom corner:

Layout in action:

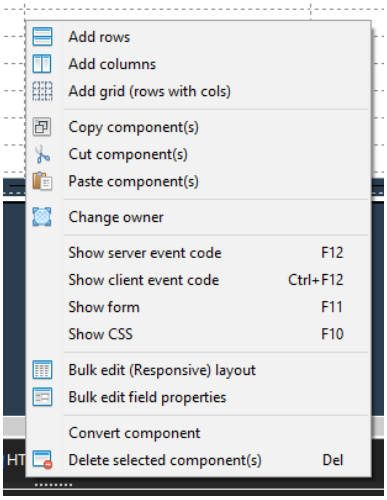
In many cases you will have used the wizards to generate your forms. In situations where you have applied new fields, or require additional functionality it might be necessary to add rows into an existing layout. In that case you can use the form designer to add rows and columns. It is easy to add new row to a form, and drag it into the required position.



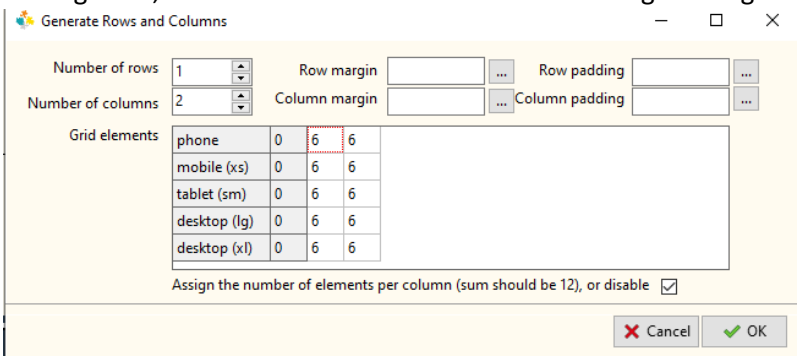
Another approach is to add the row to a selected row. This menu can be found if you select a row and then use the right mouse button to open the popup menu.



It is also possible to add a group of rows with columns in a single step. If you right click the mouse in your form then you will find the following options:



Adding rows, columns or both are then executed using a dialog:



You can set the device classes, margin and padding. It is also possible to ingore, and apply the settings as soon as you have completed your form.

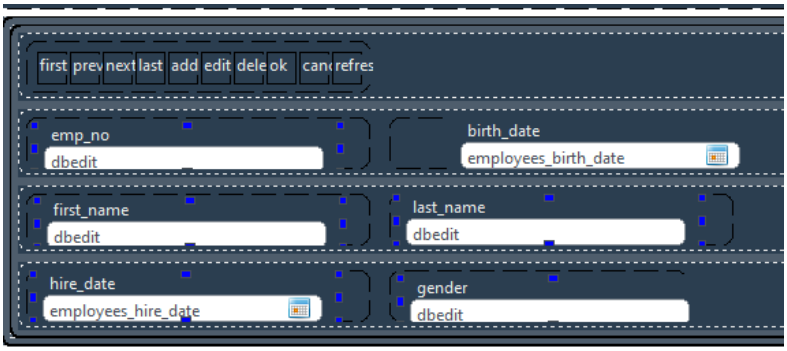
This function is helpful if you want to create a form from scratch without the help of a wizard. You can add new rows and columns at any time.

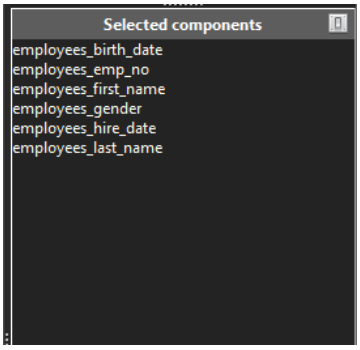
Bulk changing (layout) properties.

In many cases you have changes to apply to more fields, and it can be time consuming to select each individual field, change the property and save. PHsPeed has many features to speed up this process.

Multi select fields

This option allows you to select a field, while holding the shift key. In the right ‘selected components’ list, the fields are populated, and the property editor will adapt by showing only the overlapping properties. Changing a property will than apply the change for all selected components. This way you can easily change all label colors of the form fields.





Never change properties like label, as setting this label will then appear above all fields. Only change similar properties like, coloring, margins, paddings etc.