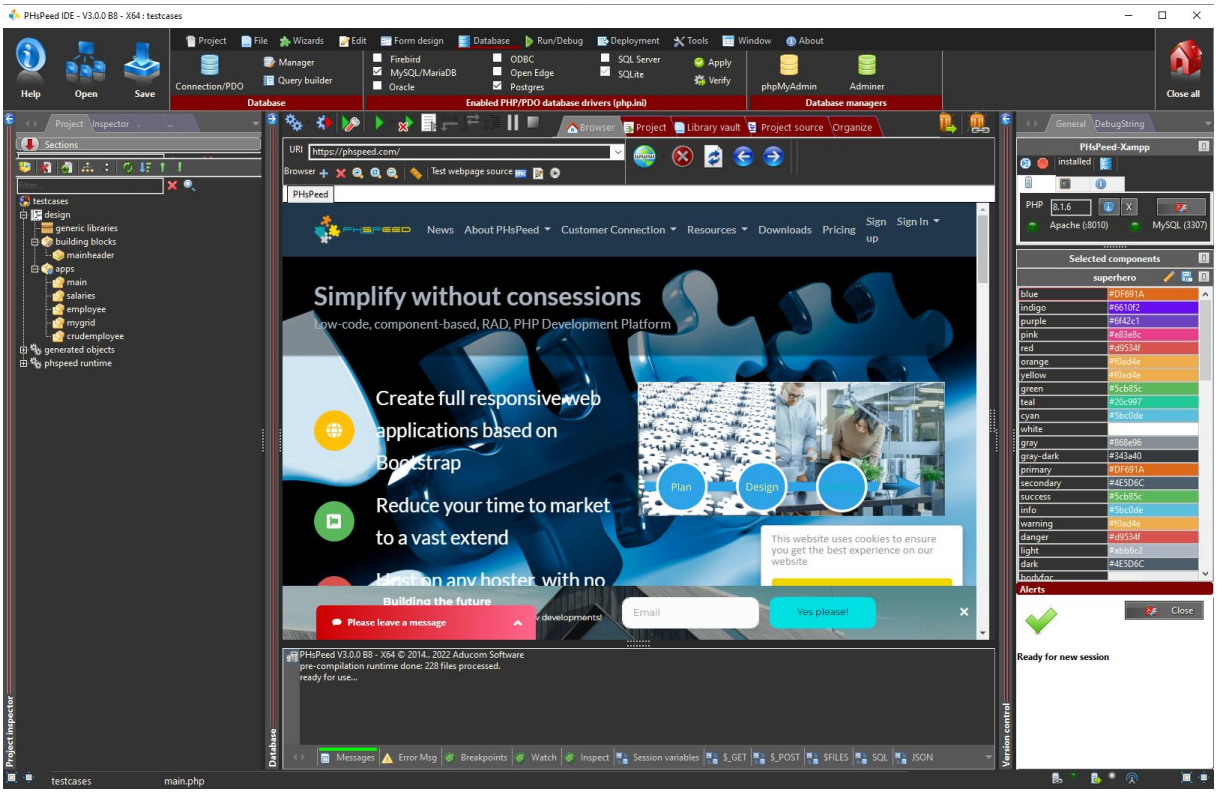
PHsPeed Introduction

“Hello world”

2

PHsPeed: Getting Started



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

In the previous document, we discussed the fundamental features of the Integrated Development Environment (IDE). Now, in this document, we will delve into the process of creating your first applications. We will begin with the quintessential "hello world" program and gradually move on to developing more advanced database-aware applications, utilizing the internal wizards provided by the IDE.

The purpose of this document is to guide you through the steps involved in building your initial applications. We will provide clear instructions and examples to help you grasp the concepts and gain hands-on experience with the IDE.

Starting with the classic "hello world" program, you will learn how to write a simple application that displays the phrase "hello world" on the screen. This introductory exercise will familiarize you with the basic structure of an application and the essential components required to run it successfully.

We will explore the IDE's capabilities in developing database-aware applications as we progress. The internal wizards integrated into the IDE will prove invaluable in streamlining the process of creating such applications. These wizards provide pre-built templates and automation tools, allowing you to generate code and establish connections with databases quickly.

By following the instructions and utilizing the internal wizards, you will gain a solid foundation in application development using the IDE. This document will equip you with the necessary knowledge and skills to build your own applications, from simple "hello world" programs to sophisticated database-driven applications.

Let's get started on this exciting journey of application development using the IDE!

First concept: Database connection

the default database definition in PHsPeed. However, if you intend to use a different database, you must create your own definition accordingly. It is worth noting that in PHsPeed, the database definitions can be reused across multiple projects, eliminating the need to define them repeatedly.

There are two important points to consider in this regard:

- IDE Connection to the Database:
  The IDE needs to establish a connection with your database in order to retrieve its metadata. If you are running the database locally, it is likely that the connection can be established without any additional steps. However, there is a possibility that you might need to install drivers to enable the connection. Please refer to the documentation or resources specific to your database to ensure that the required drivers are installed properly.

- PHP Connection to the Database:
  PHP utilizes PDO (PHP Data Objects) to connect to databases. In most cases, the necessary PDO drivers are already installed. However, if you are working with SQL Server, it may be necessary to download and install the drivers from the official Microsoft website. This step ensures that PHP can connect with the SQL Server database seamlessly.

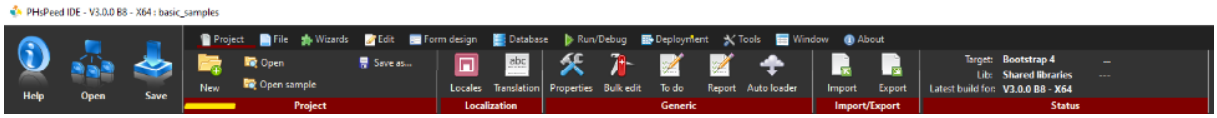
By ensuring that the IDE and PHP can connect to your database, you will have the necessary infrastructure in place to retrieve metadata and interact with the database effectively. Review the connectivity requirements and make any necessary installations or configurations to ensure a smooth development experience.

Please consult the relevant documentation and resources provided by your database vendor, PHsPeed, and PHP to obtain detailed instructions tailored to your specific database setup.

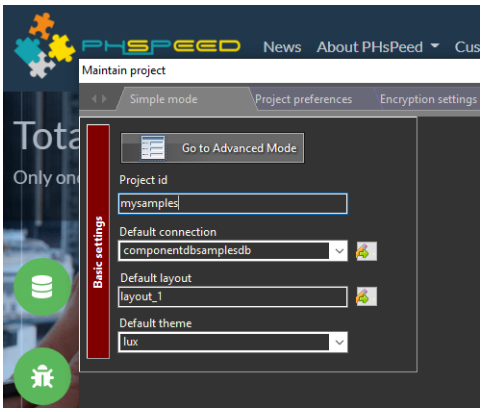
The first program: 'Hello world'

The following section provides instructions for creating a simple program, which does not require a database connection but covers essential basics. This sample program will demonstrate important concepts. Please note that you need to create a simple form to display the data. The initial step in any new project is to create one. To proceed with this process, follow the steps outlined below:

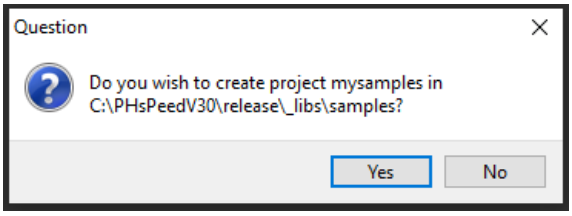
- Open the IDE and navigate to the project tab. Locate the project tab in the IDE's user interface.
- Once you have found the project tab, select the "New" option. This action will prompt the IDE to initiate the project creation process



- In the following screen, click to enter your project name, i.e. 'mysamples'



This will create a new project, including a database connection to the samples database. We will use this later.



- The IDE will now open, allowing you to design your form.

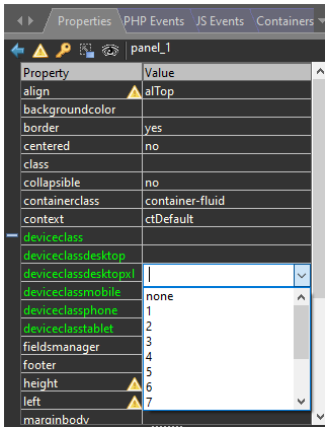
Second concept: the Bootstrap Grid system.

PHsPeed makes use of the Bootstrap Grid system to layout your forms. You don't have to incorporate the Bootstrap CSS and JavaScript files in your HTML document, as PHsPeed includes the necessary files by default.

Container: Wrap your grid layout inside a container. The container provides a fixed-width layout and centers it horizontally on the page. There are two types of containers: .container and .container-fluid. Use .container for a fixed-width container and .container-fluid for a full-width container. In PHsPeed, by default, a design container is presented in the form designer (called a Panel[1]).

Rows: Inside the container, create a row using the .row class. A row acts as a horizontal grouping of columns. All columns should be placed inside a row to ensure proper alignment and spacing.

Columns: Within the row, divide the available horizontal space into 12 columns. Bootstrap uses a grid system with a 12-column layout. To create columns, use the .col-* classes, where * represents the number of columns a specific element should occupy. For example, .col-6 will make an element occupy six columns, while .col-12 will make it occupy the full width of the row. In PHsPeed, we use the 'device class property' to specify the widths, so you don't have to worry about tags.



Column Sizing: You can create responsive layouts by combining column classes. For example, you can use .col-6 .col-md-4 .col-lg-3 to define a column that occupies six columns by default, four columns on medium-sized screens and three columns on large screens. This allows your layout to adjust dynamically based on the device screen size. In PHsPeed, you can do so by entering the values into the desired device class.

Nesting: You can nest rows and columns to create more complex layouts. Simply place a new row inside a column to create a nested grid system. Remember to follow the container → row → column structure to maintain proper alignment.

[1] PHsPeed was originally designed for Bootstrap 3 where panels were used as containers. Currently they are 'cards'. But we maintained the term 'panel'.

That's a brief overview of how the Bootstrap 4 grid system works. By utilizing the grid classes, you can easily create responsive and flexible layouts for your web pages. For more advanced options and customization, refer to the official Bootstrap documentation.

Setting up the form

In PHsPeed, when you create a project and open it, you will see an empty form with some default components. These components include:

- **Root**
  This component defines the type of PHP application you are building. In this case, it is a regular application.

- **DBConnection**
  This component is generated by default but can be deleted if you don't need to connect to a database. To delete it, click on the component, then click on the "Delete" button and confirm the deletion. If your sample project doesn't require a database, you can safely remove this component. (Which is the case in this example).

- **Form**
  This component serves as the container for all your form fields. You will add your form fields and design elements inside this component.

- **Panel**
  This component acts as an initial placeholder for your design. You can add rows and columns inside the panel to structure your form layout. Panels help you organize and group related fields or design elements together.

To start designing your form, you can begin by adding rows and columns inside the panel component. This will allow you to arrange your form fields and other design elements in a structured manner.

Concept 3: Component based design

PHsPeed is a component-based development tool. When using the form designer in PHsPeed, you will see a list of tabs that contain various components. These components are categorized into three types: layout components, visual components, and non-visual components.

Layout components: These components are used to design the layout of your application. Examples include panels, rows, and columns. They help you structure and organize the visual elements of your application's user interface.

Visual components: These components are used to define the input and output of HTML forms in your application. They provide the user interface elements such as buttons, text fields, dropdowns, checkboxes, and more. Visual components allow users to interact with your application and input data.

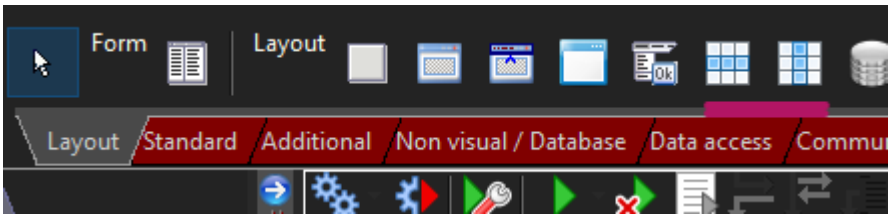
Non-visual components: These components are primarily functions or resources that do not have a visual representation in the user interface. They serve various purposes, such as handling database operations, executing queries, managing connections to databases, and performing other backend tasks.

By utilizing these three types of components, PHsPeed allows you to develop applications with a visual form designer and a wide range of functionality. You can design the layout, define user input and output, and implement various backend operations to create robust and interactive web applications.

Layout of the form

In the basic "hello-world" project, the focus is on creating a simple form that displays a single line of text. To achieve this, you can follow these steps using the layout toolbar:

- Locate the layout toolbar: Look for the toolbar in your development environment that contains the various layout components you can use to design your form



- Find the "row" component: In the layout toolbar, search for the component representing a row. This component allows you to organize and structure the elements within your form horizontally.

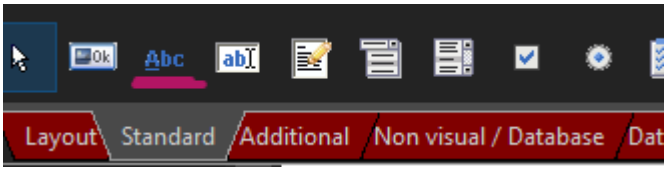
- Add the row component: Once you have located the "row" component in the layout toolbar, click on it to select it. Then, click within your form (or the designated panel) to place the row component in that location. This step adds the row component to your form layout.

- Select the "column" component: After adding the row component, search for the component representing a column in the layout toolbar. The column component is used to organize elements vertically within a row.

- Add the column component: Similar to the previous step, click on the column component in the layout toolbar to select it. Then, click within the row component you added earlier to place the column component inside the row. This step adds the column component to the row layout.

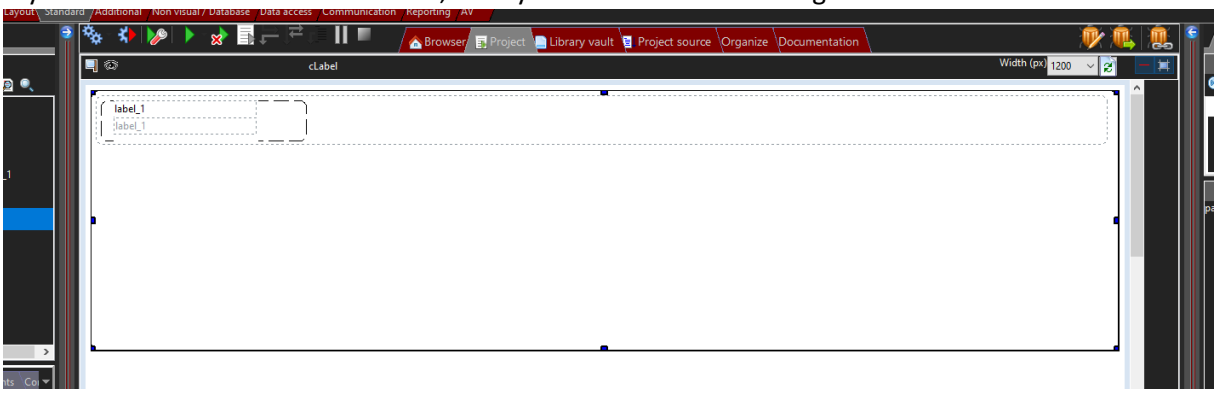
Visual components to the form

- Add a label component: Within the ‘standard tab’, locate the component representing a label. This component will display the line of text in your "hello-world" project.



- Place the text component in the column: Click on the component in the layout toolbar to select it. Then, click within the column component to place the text component inside it.

If you have followed the instructions, then you should see something like this:



- Customize the label component: With the label component selected, you can modify its properties, such as changing the displayed text or adjusting the formatting options.

Concept: Components have properties

In the context of software development, component properties refer to the characteristics or attributes of a software component that define its behavior, appearance, or functionality. These properties provide a way to configure and customize the component based on specific requirements.

Here are a few key points about component properties:

- Configurability: Properties allow you to configure various aspects of a component, such as its size, color, position, or visibility. By changing these properties, you can modify the component's behavior or appearance to suit different needs.

- Data binding: Properties can be bound to data sources, enabling dynamic updates based on changes in the underlying data. For example, a data grid component may have properties like "dataSource" or "columns" that can be bound to a dataset, allowing the grid to display and manipulate the data accordingly.

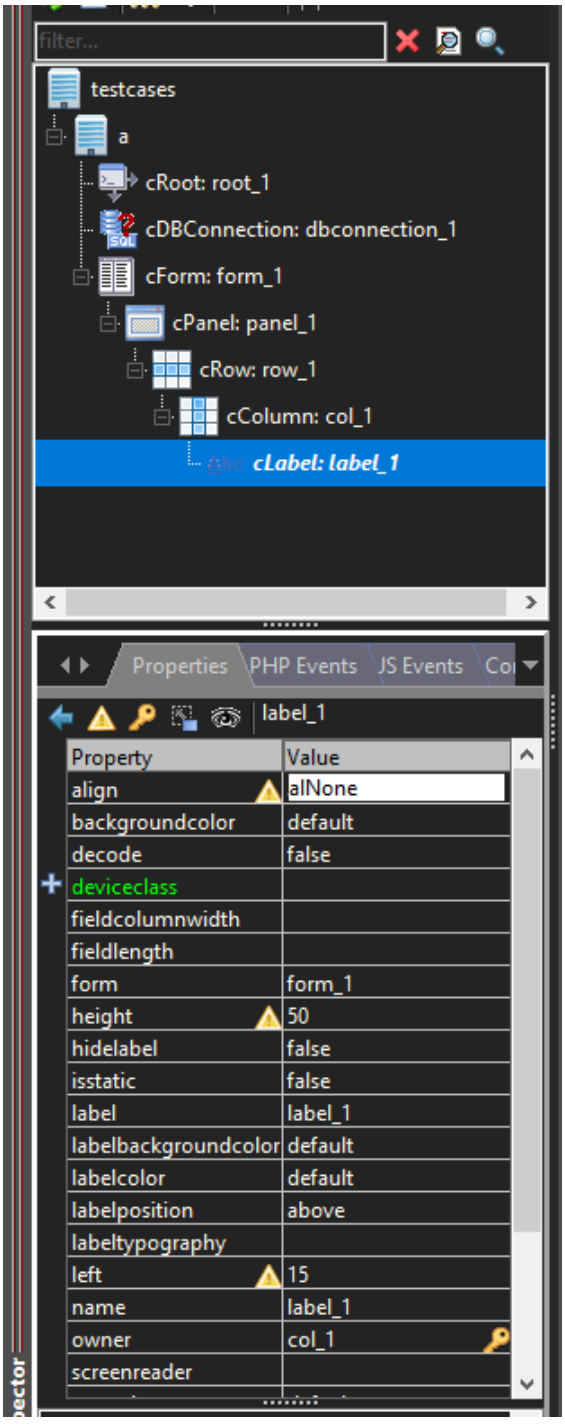
- Accessibility: Properties can be used to specify accessibility-related information, such as the component's label, role, or keyboard navigation behavior. These properties help ensure that the component is usable and understandable for users with disabilities.

- Events and callbacks: Components often expose properties that represent event handlers or callbacks. These properties allow you to associate custom logic or behavior with specific events, such as a button's "onClick" property, which triggers a function when the button is clicked.

- Default values: Each property can have a default value, which is used when the property is not explicitly set. Default values provide a starting point and ensure that the component behaves predictably even if certain properties are not specified.

- Validation and type checking: Properties can be validated to enforce constraints on their values. This includes checking the type, range, or format of the property value to prevent invalid or unexpected input.

The properties of a component can be accessed and modified through the property editor. When you select a component, its corresponding properties will be displayed in the editor for you to interact with and make changes.

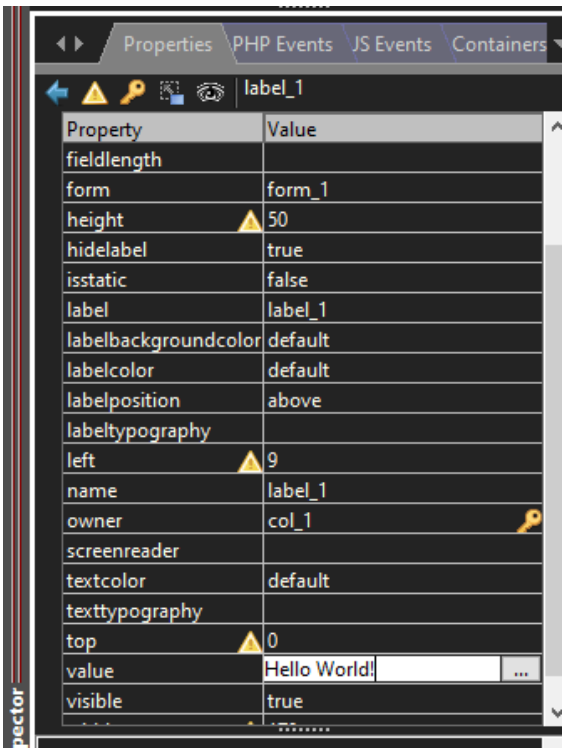


Some properties hold greater significance than others. When it comes to visual components, the "label" and "value" properties take precedence. The label refers to the descriptive text displayed before or above the field, while the value represents the field's actual content. It's worth noting that the value cannot be modified through input in a label field.

Select the label component and change the properties as follows:

Hide label: set to true
Value: set to ‘hello world’

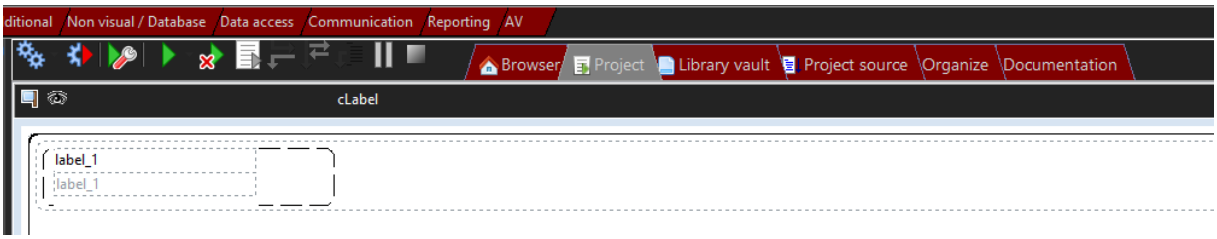
Now you are ready to generate and run your first PHsPeed project!



Concept: generating and running applications

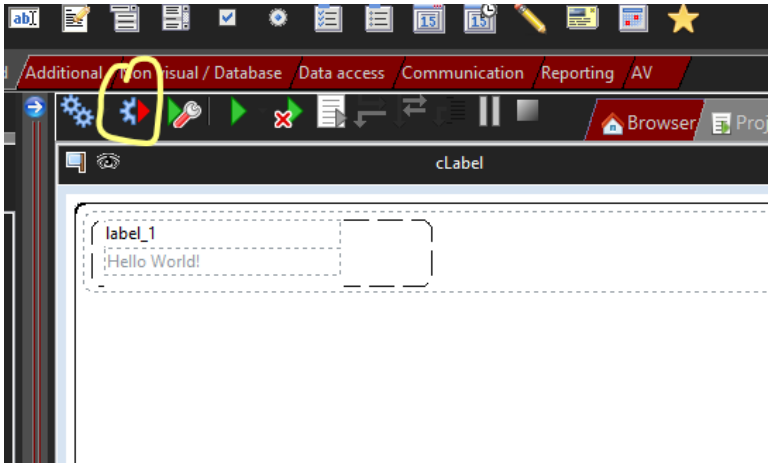
PHsPeed includes an integrated Apache webserver. To enable application execution, PHsPeed requires code generation from your specifications and subsequent deployment to the internal webserver.

In scenarios where you are working on a complete project, generating the entire code might not be ideal. In such cases, you may prefer to generate code specifically for the modified applications. PHsPeed offers a convenient toolbar that allows you to generate and execute the code as needed.

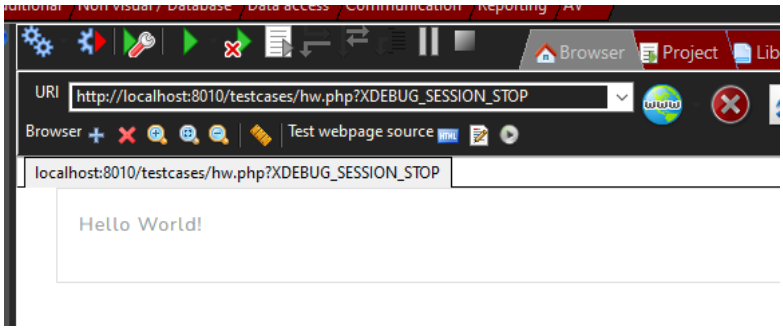


Within this document, I will focus on providing an overview of the first two buttons found on the toolbar:

Double gear icon:	Generates the code for the modified application(s).
Half gear icon with a red arrow:	Generates and executes the code specifically for the selected application.



Click on the Half gear icon to generate and run the code:



The result should look like this:

Congratulations! You created your first application in PHsPeed!