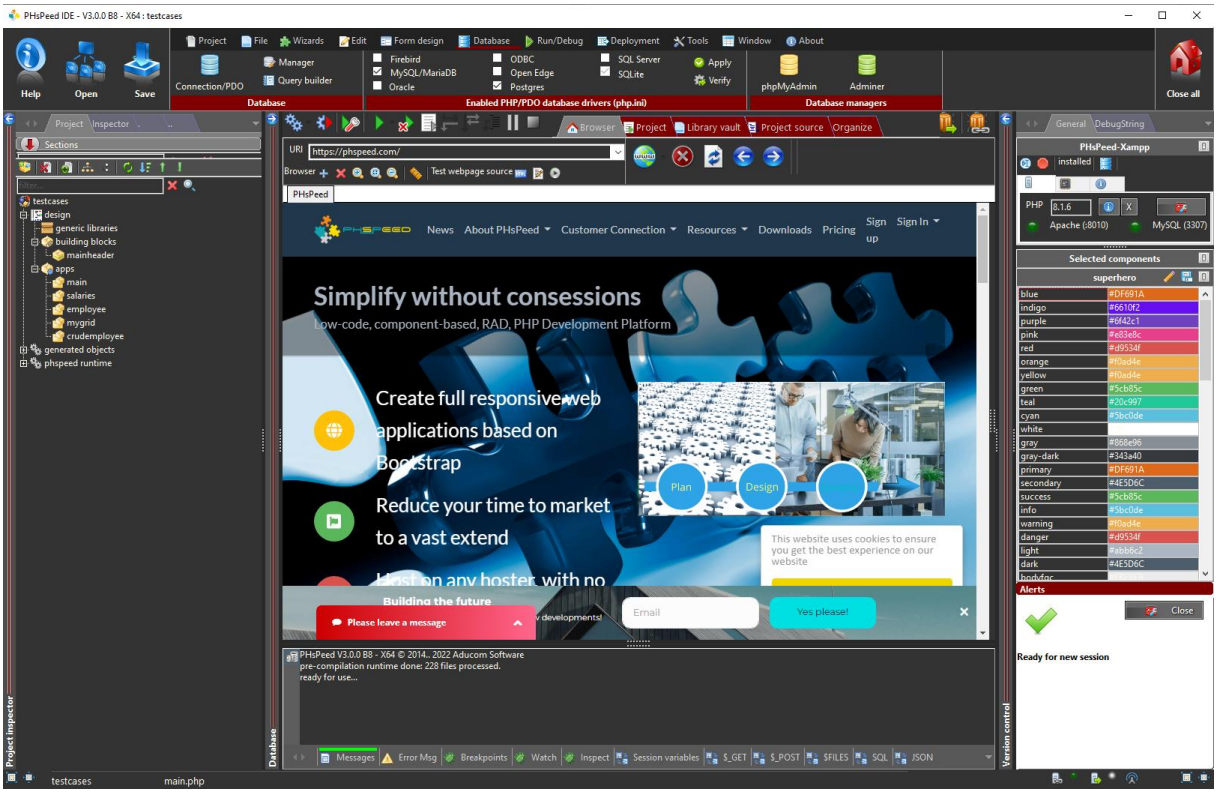
PHsPeed Introduction

Deploying your applications

8

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. We also explained the bootstrap grid system to lay out your forms. You have a basic understanding about events, integrating custom code, and the application logic flow. You also know the basic principles of debugging your application. At some point you are ready with your application, all is working well and want to deploy it on the internet or intranet.

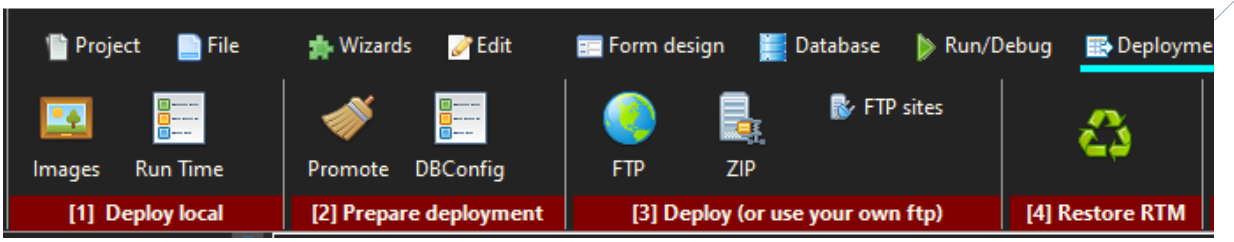
Deploying your application

This manual aims to provide you with comprehensive insights how to deploy your application on the web.

Introduction.

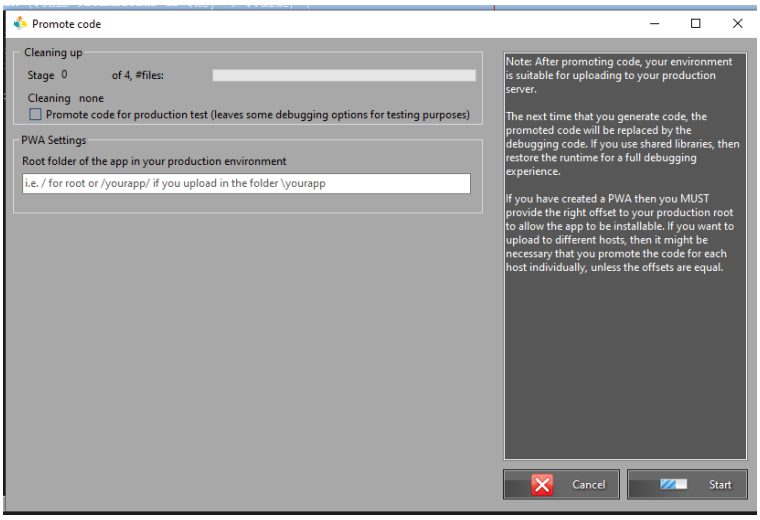
When PHsPeed generates your application, then it will also generate code that enables the IDE to interact with the application. In test mode, you can use properties to set database credentials, that – most likely – will differ from your production environment. Also, in your production environment, you don't want to have some kind of readable configuration files that exposes all kinds of data.

Deploying your application is supported in the Deployment Tab of the IDE



Promote code

The first step to take is to promote your code. This will remove all the debugging code and IDE related stuff. It will also remove comment lines and empty lines to improve speed. After all, production site is not meant for debugging and readability. By default it will also disable error reporting. This might cause blank screens when you have issues, but it is considered to be the most safe way by not exposing error messages to potential hackers. For your first test, you can consider to leave some debugging options open, and when all works well in production, do a redeploy to disable error messages. For a standard deploy, leave the PWA (progressive web application) settings open. Note: PHsPeed will also promote the runtime library.



Now you have a few options to move the promoted code to your webserver. The first one is to use the embedded FTP option. You can setup your FTP server and upload from the IDE. As the embedded FTP client is very basic, you might consider using a dedicated FTP client like FileZilla. This client also allows you to deploy using a secure channel.

When PHsPeed generates code then it will do so in the embedded WebServer:

[drive:]/phspeed[V31]/xampp/htdocs/[project]/

Shared libraries vs Private libraries.

By default PHsPeed uses shared libraries. That means that you can create projects that all use the same runtime. It is evident that it saves a lot of space, as the runtime is big. The other option is to use private libraries, and that requires you to deploy the runtime initially for your application. It will then be an integral part of your project.

Deploying shared libraries.

The shared libraries are in [drive:]/phspeed[V31]/xampp/htdocs/_libs

Initially you must copy this folder to your webserver above the root of your projects:

htdocs

_libs
Project 1
Project 2

If you use a folder for your projects then a logical structure would be:

htdocs
[apps]
_libs
Project 1
Project 2

Deploying the runtime only need to be done once. Whenever a new runtime comes out you have to redeploy.

Deploying private libraries.

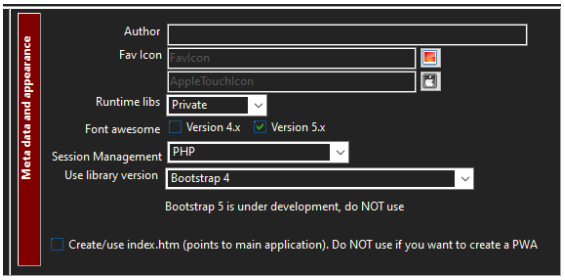
You do not have to take special steps to deploy the private libraries as they are part of you full application. If you want to host more projects on the same server, then be aware of the fact that each project is containing a private version of the libraries and thus takes quite some more webspace.

Considerations

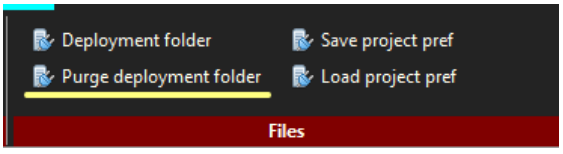
The private libraries were the original way of deploying applications. But because of the webspace the private libraries were introduced. Both methods have pro's and con's:

Shared Libraries	Private Libraries
Only one runtime, limits required webspace	Every project contains the runtime occupying webspace
As there is only one runtime, upgrading is a quick process	Each project needs to be re-uploaded on new versions of the runtime
Before upgrading the runtime, applications must have been tested for functionality as the upload has influence on each project	You can test, upgrade and deploy applications one-by-one.

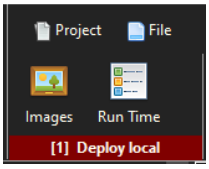
Switching between shared and private libraries.



It is possible to switch between the use of shared and private libraries. In the project properties you can set the required option. After setting the option it is important to purge the deployment environment. This option is part of the deployment tab.



If you have changed to private libraries, then deploy the images and runtime.

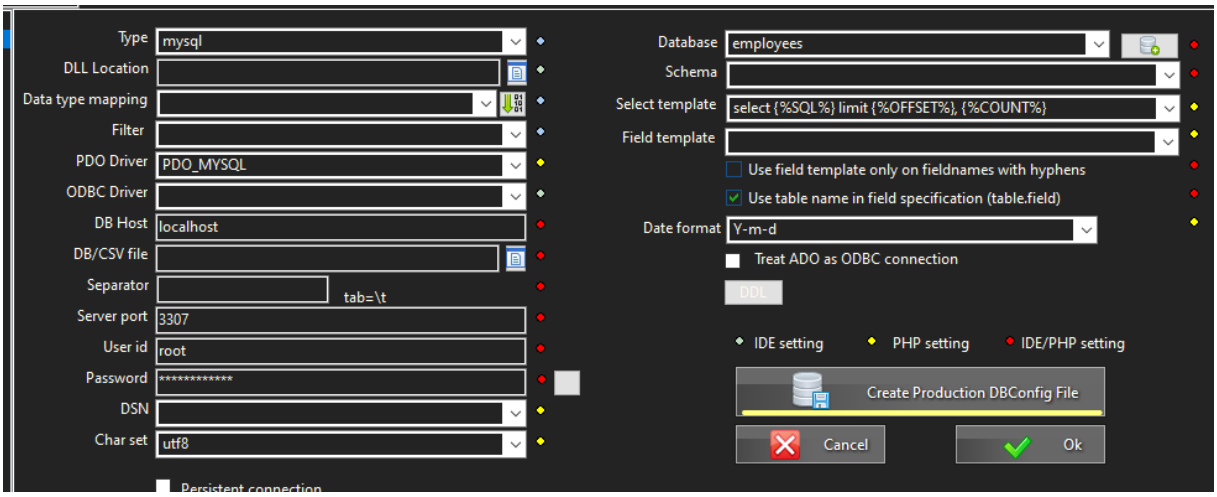


If you have changed to shared libraries, deploy the images.

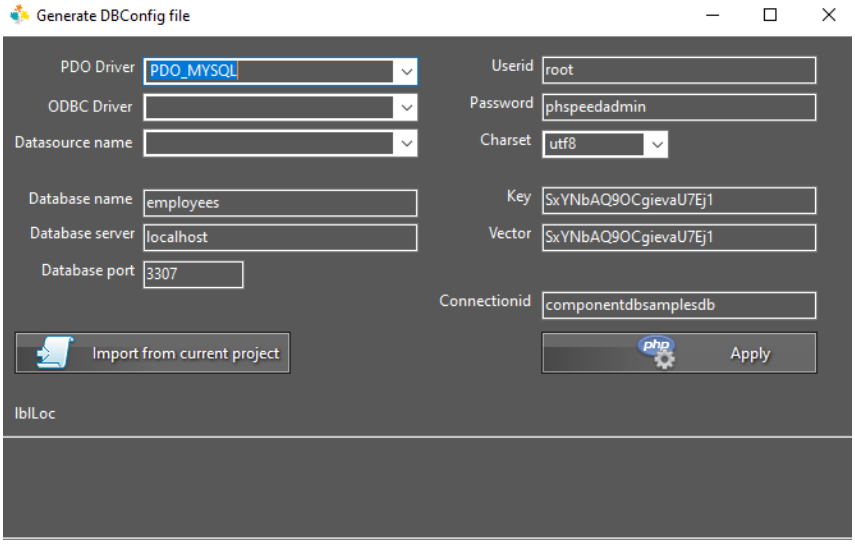
Next, fully regenerate your applications and promote code. Note: it is handy to use the purge deployment folder. If you have worked on your project and have deleted modules, then the generated code will remain in the deployment folder. A purge is always a great way to get rid of orphan files and other left-overs.

Setting database credentials

After deploying your application, you cannot make use of standard config files or the set properties (Except if you have intercepted the login procedure of course). Instead, you need to create an encrypted configuration file. The most easy way to create this configuration file is to use the Database configuration (database->connection/pdo).

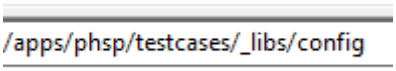


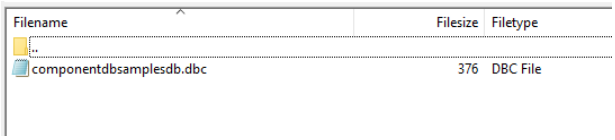
This will open the generate dbconfig file.



Enter the data that is required in the production environment! The key and vector are retrieved from the IDE settings. If you have deployed your application with different settings then you can change this. The encryption data for the application and encrypted configuration file must match, obviously.

Click apply to generate the encrypted file. This is stored in your project in the _libs/config folder:





It is also possible to create a config application that allows you to set the credentials in the production environment. Same goes for Email credentials, although most developers will put this data in a database table.

Index.php / index.htm

By default PHsPeed uses a main as the starting point. You can change that in the project settings. The idea is that using index might not always be a good idea from a security perspective. However, there are situations where you might just want to use index. One option is to start with index.php instead of main.php. The other option is to have a index.htm created. This option can be set in the project properties.