Debugging with a PHP Web Application Debug Configuration

In this debugging mode, IntelliJ IDEA fully controls the debugging process: it launches the application, opens the browser, and activates the debugging engine according to a *PHP Web Application debug configuration*.

A *PHP Web Application* debug configuration tells IntelliJ IDEA where the PHP application to debug is deployed, the URL address to access the starting page of the application, the browser to open the starting page in, and the correspondence between files and folders deployed on the server and their local copies (*mappings*).

Mappings are necessary for the following reason. During the debugging session on the local or remote server, the server side of the debugger tells IntelliJ IDEA the name of the currently processed file and the number of the line to be processed. IntelliJ IDEA opens the local copy of this file and indicates the line with the provided number. This behaviour is enabled by specifying correspondence between files and folders on the server and their local copies. This correspondence is called *mapping*.

You can also specify the scripts requests to which you want IntelliJ IDEA to ignore during debugging. This approach can be useful, when your application contains scripts that use AJAX. Suppose you have a `menu-ajax-script.php` that "reloads" a part of your web page. This script works properly so you do not need to debug it. However, this script is still requested during the debugging session. To have incoming connections to this script ignored, add the `menu-ajax-script.php` script to the *skipped paths* list. You can also group such scripts into folders and add these folders to the "ignore list".

On this page:

Preparing the debugging engine

Before you start debugging, make sure that you have a debugging engine installed and configured properly. IntelliJ IDEA supports debugging with two most popular tools: XDebug and Zend Debugger. These tools cannot be used simultaneously because they block each other. To avoid this problem, you need to update the corresponding sections in the `php.ini` file. To find out which `php.ini` file is active, create and run a test file with `phpinfo()`, then search for the *Loaded Configuration File*.

For more information on configuring debugging engines, see Configuring XDebug, Configuring Zend Debugger, http://confluence.jetbrains.com/display/PhpStorm/Xdebug+Installation+Guide, and http://confluence.jetbrains.com/display/PhpStorm/Zend+Debugger+Installation+Guide.

Setting the breakpoints

Breakpoints are source code markers used to trigger actions during a debugging session. Typically, the purpose behind setting a breakpoint is to suspend program execution to allow you to examine program data. However, IntelliJ IDEA can use breakpoints as triggers for a variety of different actions. Breakpoints can be set at any time during the debugging process. Your breakpoints don't affect your source files directly, but the breakpoints and their settings are saved with your IntelliJ IDEA project so you can reuse them across debugging sessions.

1. Place the caret on the desired line of the source code.

   Breakpoints can be set in the PHP context inside `*.php`, `*.html`, and files of other types. Only executable lines are valid locations for line breakpoints. Comments, declarations, and empty lines are not valid locations for the

2. Do one of the following:
   - Click the left gutter area at a line where you want to toggle a breakpoint.
   - On the main menu, choose **Run | Toggle Line Breakpoint**.
   - Press `Ctrl+F8`.

Creating a PHP Web Application debug configuration

1. Open the Run/Debug Configuration dialog box by doing one of the following:
   - On the main menu, choose **Run | Edit Configurations**.
   - Press `Shift+Alt+F10`, then press `0` to display the **Edit Configuration** dialog box or select the configuration from the pop-up window and press `F4`.
2. Click + on the toolbar or press `Insert`. From the drop-down list, select the **PHP Web Application** configuration type. The PHP Web Application dialog box opens.
3. Specify the configuration name.
4. Choose the applicable debug server configuration from the **Server** drop-down list or click the **Browse** button ... and define a debug server configuration in the Servers dialog box that opens.
5. In the **Start URL** text pox, type the server path to the file that implements the application starting page. Specify the path relative to the server configuration root.

   > The read-only field below shows the URL address of the application starting page. The URL address is composed dynamically as you type.

6. Specify the browser to open the application in. Choose a configured browser from the **Browser** drop-down list or click the **Browse** button ... and specify another browser in the Web Browsers dialog box that opens.
7. To have the debugging engine stop as soon as connection between it and IntelliJ IDEA is established (instead of running automatically until the first breakpoint is reached), select **Stop at first line** check box.

Defining a debug server configuration

1. Open the Servers dialog box by doing one of the following:
   - Choose **File | Settings** for Windows and Linux or **IntelliJ IDEA | Preferences** for OS X. Then click **Servers** under the **PHP** node.
   - In the Run/Debug Configuration: PHP Web Application dialog box, click the **Browse** button  next to the **Server** drop-down list.
2. Specify the server configuration name.
3. Specify the host where the application is run and the port to access it.
4. From the **Debugger** drop-down list, choose the debugging engine to use.
5. Specify how the IntelliJ IDEA will set up a correspondence between files on the server and their local copies. Based on these mappings, IntelliJ IDEA will open local copies of currently processed files.
   - To have IntelliJ IDEA suggest mappings itself, clear the **Use path mappings** check box. When you start a debugging session, IntelliJ IDEA will try to detect the local copies of the application files on the server. The suggestions are displayed in a dialog box where IntelliJ IDEA asks you to confirm or edit suggested mappings.

     > IntelliJ IDEA detects mappings 100% correctly if the application is deployed to the server root and the folder structure on the server is identical with the folder structure under the project root.

   - To specify correspondence between files on the server and their local copies manually, select the **Use path mappings** check box and map files and folders on the server to their local copies.

Initiating a debugging session and examining the suspended program

1. To start debugging, click the **Debug** button  on the toolbar.
2. As soon as the debugger suspends on reaching the first breakpoint, examine the application by analyzing *frames*. A *frame* corresponds to an active method or function call and stores the local variables of the called method or function, the arguments to it, and the code context that enables expression evaluation. All currently active frames are displayed on the **Frames** pane of the Debug tool window. where you can switch between them and analyze the information stored therein in the **Variables** and **Watches** panes. For more details, see the section Examining Suspended Program.
3. Continue running the program and examine its frames as soon it is suspended.
   - To control the program execution manually, step through the code using the commands under the **Run** menu or toolbar buttons: **Step Into** (`F7`), **Step Out** (`Shift+F8`), **Step Over** (`F8`), and others. For more details, see Stepping Through the Program.
   - To have the program run automatically up to the next breakpoint, resume the session by choosing **Run | Resume Program** or pressing `F9`

Specifying scripts to skip requests to

This approach can be useful, when your application contains scripts that use AJAX. Suppose you have a `menu-ajax-script.php` that "reloads" a part of your web page. This script works properly so you do not need to debug it. However, this script is still requested during the debugging session. To have incoming connections to this script ignored, add the `menu-ajax-script.php` script to the *skipped paths* list. You can also group such scripts into folders and add these folders to the "ignore list".

1. Open the Project Settings, click **Debug** under the **PHP** node, then click **Skipped Paths**.
2. On the Skipped Paths page that opens, configure an "ignore list" of scripts and folders with scripts not to be invoked if IntelliJ IDEA receives incoming connections to them.
   - To add a new entry to the list, click the **Add** button + or press `Alt+Insert`. Then click the **Browse** button ... and in the dialog box that opens choose the file or folder to skip connections to.
   - To remove an entry from the list, select it and click the **Remove** button — or press `Alt+Delete`. The script will be now executed upon receiving requests to it.
3. To have IntelliJ IDEA inform you every time it receives a request to a script to be skipped, select the **Notify about skipped paths** check box.

See Also

Procedures:

- PHP Debugging Session
- Creating and Editing Run/Debug Configurations
- Debugging PHP Applications
- Debugging

Reference:

- Run/Debug Configuration: PHP Web Application
- Servers

Web Resources:

- Developer Community