PHP Script Debugging

- Introduction
- Attaching a PHP Interpreter
    - Attaching a PHP INI
- The Script Debugger
- Script Debug/Run Sessions

Introduction

The Studio supports two types of PHP debugging:

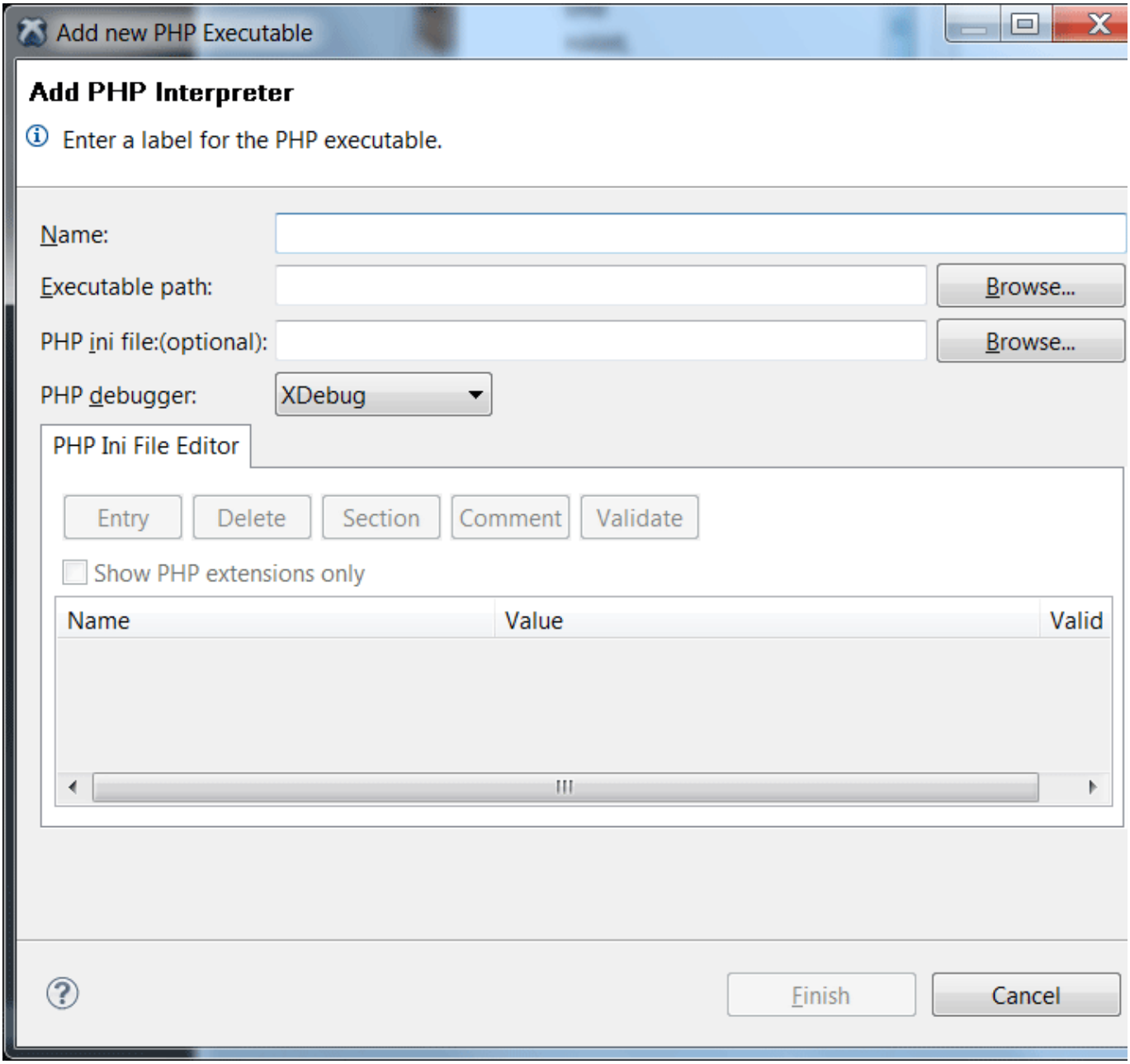
1. Internal debugging (for example, Script debug without a server-side involvement).
2. Web Page debugging that uses a debugger extension (either Zend or XDebug) installed on a server.

This page will guide you through the setup process and the debugging process when a single script is debugged.
The single-script mode is a quick way to debug through the functionality of a single PHP file. In case you are debugging a PHP application, you will find that a '*Web Page*' debugging is the way to go.

Attaching a PHP Interpreter

A PHP Script debug session uses a local PHP interpreter and a debugger extension defined in its php.ini.
In order to attach an interpreter:

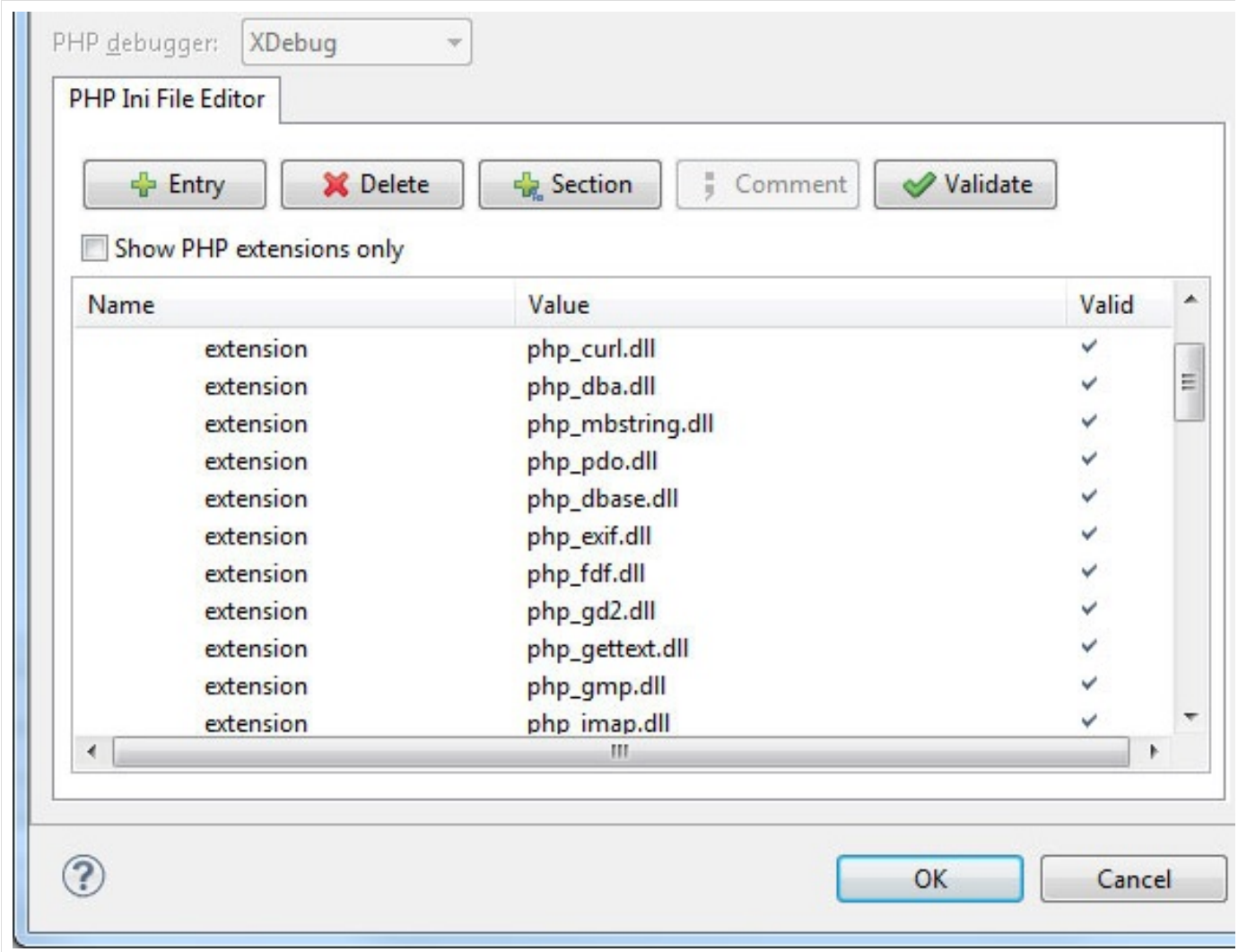
1. Open the Studio's *Preferences* page and locate *Aptana Studio -> Editors -> PHP -> Debug -> PHP Interpreters*.
2. Click to add a new interpreter and browse for the PHP executable location on your disk.
3. The php.ini location is optional in that dialog, and when not specified, the system will look for one under the executable path.



Attaching a PHP INI

When you attach a php.ini location to the interpreter settings, you will be able to perform some extension related actions:

- Add/remove an entry
- Add/remove a section
- Comment an entry
- Validate the settings



A little bit on the validation process:

When you hit the *Validate* button, the PHP interpreter will try to execute the current php.ini settings and collect the errors/warnings.

This validation can be used to determine which of the extensions triggered warnings, or even triggered fatal errors and could not be loaded. At the end of the validation, question marks next to the extensions will be replaced with appropriate icons (Error, Warning, or OK).

Hovering over the extension line displays the error/warning message that the PHP process outputted when it was validating.
In general, any fatal extension should be commented out to allow the PHP process function currently.

> ⚠ This process is no a "100% bullet-proof", and in some cases where the PHP process fails to load, a deeper investigation might be needed to determine the cause.

The Script Debugger

The Studio debugger works with XDebug and ZendDebugger extensions. Both can be attached via the INI editor we described above.
Once you have a PHP interpreter set up correctly, you can attach a debugger extension to the php.ini.

1. Open the Studio preferences at *Aptana Studio -> Editors -> PHP -> Debug -> PHP Interpreters*.
2. Click to edit your previously defined PHP interpreter.
3. Set the type of *PHP Debugger* in the dialog dropdown. You can choose between XDebug and ZendDebugger.
4. Edit the ini section to add your debugger extension by setting the extension name as *zend_extension_ts* and the value as your debugger .dll or .so extension (Note that *zend_extension* should be used for non-thread-safe extensions).
5. Click to validate your PHP ini to see if the extension was loaded correctly.

Once it's all set-up, you are ready for a script *Debug* (or *Run*) session.

Script Debug/Run Sessions

A session can be initiated in several different ways.

Option 1:

1. Right-click the editor area and select 'Debug As'->'PHP Script'.
2. You will be prompted to move to the debug perspective - Click yes.
3. In that perspective, you will see the debug stack, variables, and breakpoints views. You can also add more views like the *Expressions* vie w to track any Watch expressions you add.
4. From here, you can do a step into, step over and step return by clicking the buttons on the stack view or by using F5/F6/F7 keys.
5. You can set more breakpoints as you debug, or even run to a line that you mark (right-click + 'Run to line' action or Ctrl+R).

Option 2:
Click *Run* in the application menu bar and then click *Debug Configurations...*. Create a new *PHP Script* debug configuration. Make sure that the selected debugger-type match the type that was set for the selected PHP executable, then launch the session.

The same applies to a *Run* session that only runs the script and output to the console.