Go! AOP documentation Documentation

Release 0.0.1-alpha

Alexander Lisachenko

Oct 28, 2017

Table of Contents

CHAPTER 1

Introduction to AOP

Glossary

Aspect

A modularization of a concern that cuts across multiple objects. Logging, caching, transaction management are good examples of a crosscutting concern in PHP applications. Go! defines aspects as regular classes implemented empty Aspect interface and annotated with the @Aspect annotation.

Join point

A point during the execution of a script, such as the execution of a method or property access.

Advice

Action taken by an aspect at a particular join point. There are different types of advice: @Around, @Before and @After advice.

Pointcut

A regular expression that matches join points. Advice is associated with a pointcut expression and runs at any join point matched by the pointcut (for example, the execution of a method with a certain name).

Introduction (Also known as an inter-type declaration)

Go! allows you to introduce new interfaces (and a corresponding implementation with trait) to any user-defined class. For example, you could use an introduction to make all Data Transfer Objects implement an Serializable interface, to simplify persistence.

Weaving

Linking aspects with other application types or objects to create an advised object. This can be done at any time: compile time, load time, or at runtime. Go! performs weaving at runtime and doesn't require any additional steps to transform the source code.

Types of advices

Before advice

Advice that executes before a join point, but which does not have the ability to prevent execution flow proceeding to the join point (unless it throws an exception).

After returning advice

Advice to be executed after a join point completes normally: for example, if a method returns without throwing an exception.

After throwing advice

Advice to be executed if a method exits by throwing an exception.

After (finally) advice

Advice to be executed regardless of the means by which a join point exits (normal or exceptional return).

Around advice

Advice that surrounds a join point such as a method invocation. This is the most powerful kind of advice. Around advice can perform custom behavior before and after the method invocation. It is also responsible for choosing whether to proceed to the join point or to shortcut the advised method execution by returning its own return value or throwing an exception.

The motivation for Go! (and likewise for aspect-oriented programming) is the realization that there are issues or concerns that are not well captured by traditional programming methodologies. Consider the problem of enforcing a security policy in some application. By its nature, security cuts across many of the natural units of modularity of the application. Moreover, the security policy must be uniformly applied to any additions as the application evolves. And the security policy that is being applied might itself evolve. Capturing concerns like a security policy in a disciplined way is difficult and error-prone in a traditional programming language.

Concerns like security cut across the natural units of modularity. For PHP the natural unit of modularity is the class. But in PHP crosscutting concerns are not easily turned into classes precisely because they cut across classes, and so these aren't reusable, they can't be refined or inherited, they are spread through out the program in an undisciplined way, in short, they are difficult to work with.

CHAPTER 2

Installation

Install Go! AOP framework on any PHP project

Install Go! AOP on Symfony framework

Install Go! AOP on Laravel framework

Install Go! AOP on Zend framework

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index