GUICE AOP

krótki przegląd

Guice jako framework do DI oferuje podobnie jak Spring możliwość wkorzystania AOP. Nie jest to pełne AOP znane z AspectJ tylko jak to nazywają sami twórcy „przerywanie metod". AOP w Guice nie jest projektowane z myślą o codziennym programowaniu tyko o twórcac bibliotek.

Podstawą działania AOP w guice są tz. matchery. Do zaaplikowania aspektu potrzeba dwóch: do zdefiniowania klasy docelowej oraz dla metod w tej klasie. Dla uproszczenia do podstawowych scenariuszy stworzono fabryki.

Gdy matchery dopasują się do wywołania danej metody w danej klasie wywoływany jest pasujący MethodInterceptor. Jest to odpowiednik rady around w AspectJ. Może wywoływać swoją logikę przed, po metodzie, może mapować parametry czy zwracaną wartość, może decydować czy dana metoda zostanie wywołana.

Przykład, metody nie działające w weekend

Zdefniujmy sobie adnotację:

```
@Retention(RetentionPolicy.RUNTIME) @Target(ElementType.METHOD)
@interface NotOnWeekends {}
```

oraz klasę:

```
public class RealBillingService implements BillingService {
  @NotOnWeekends
  public Receipt chargeOrder(PizzaOrder order, CreditCard creditCard) {
    ...
  }
}
```

teraz zaimplementujmy sobię interceptor:

```
public class WeekendBlocker implements MethodInterceptor {
  public Object invoke(MethodInvocation invocation) throws Throwable {
    Calendar today = new GregorianCalendar();
    if (today.getDisplayName(DAY_OF_WEEK, LONG, ENGLISH).startsWith("S")) {
      throw new IllegalStateException(
          invocation.getMethod().getName() + " not allowed on weekends!");
    }
    return invocation.proceed();
  }
}
```

Teraz skonfigurujmuy to:

```
public class NotOnWeekendsModule extends AbstractModule {
  protected void configure() {
    bindInterceptor(Matchers.any(), Matchers.annotatedWith(NotOnWeekends.class),
        new WeekendBlocker());
  }
}
```

i po wywołaniu chargeOrder dostaniemy w weekend:

```
Exception in thread "main" java.lang.IllegalStateException: chargeOrder not allowed on weekends!
        at com.publicobject.pizza.WeekendBlocker.invoke(WeekendBlocker.java:65)
        at com.google.inject.internal.InterceptorStackCallback.intercept(...)
        at com.publicobject.pizza.RealBillingService$$EnhancerByGuice$
$49ed77ce.chargeOrder(<generated>)
        at com.publicobject.pizza.WeekendExample.main(WeekendExample.java:47)}}}
```

Ograniczenia:

Guice korzysta z generacji bytecode'u, daltego jego użycię jest ograniczone do wspieranych środowisk (np. Android nie ma takich możliwości).

- Klasy muszą być publiczne albo prywatne dla pakietu
- Klasy nie mogą być finalne
- Metody muszą być publiczne albo prywatne dla pakietu
- Metody nie mogą być finalne
- Instancję muszą być tworzone przez Gucice'a poprzez adnotację `@Inject` korzystając z bezargumentowego konstruktora.

Więcej: http://code.google.com/p/google-guice/wiki/AOP