AoP i Dependency Injection

Technologie obiektowe i komponentowe

Krzysztof Romanowski

Czerwiec, 2013

Dlaczego AoP i DI

- Łatwa implementacja
- Naturalne połączenie
- Lekkie frameworki

Sping AOP

- 2 sposoby deklarowania
  - @AspectJ
  - Schema-based (xml)
- Sematycznie wzorowany na AspectJ
- Oparty na czystej Javie (nie wymaga specjalnej kompilacji ani dostępu do classloaderów)
- Ograniczony (celem aspektów są tylko wywołania metod)

Spring AoP cd.

- Wykorzystuj mechanizm proxy
- Zaleca programowanie do interfejsu
- Stworzony jako dodatek do Springa pozwalający wstrzykiwać np. tranzakcyjność nie jako framework AoP
- Integracja z AspectJ

Spring AoP

- Zalety
  - Lekki
  - Napisany w czystej Javie
  - Nie ma dużych wymagań
- Wady
  - Nie jest to pełny framework
  - Nie działa poza DI
- Zastosowanie
  - Ulepszanie aplikacji w springu
  - Dodawanie „poziomej” modularności

Tranzakcyjność w Springu

- Korzysta z AspectJ
- Nie wymaga całości Springa
- Przykład dobrej biblioteki AoP

Guice AoP

- Podobny jak Spring AoP
- Jeszcze uproszczony (tylko odpowiednik rady around)
- Szereg ograniczeń
- Zastosowanie
  - Podobnie jak Spring AoP tylko w mniejszej skali

Podsumowanie

- AoP jako dodatek do DI znacznie rozszerza jego możliwości
- Ograniczenia narzucane przez frameworki wspierają czytelnieszy kod
- Mniejsze możliwości niż AspectJ
- AoP jako cegiełka do budowania bibliotek nie aplikacji