Seasar Conference
2007 Spring

いまさら人には聞けない DI×AOP入門

2007.5.27

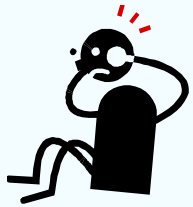
エスエムジー株式会社
小森 裕介 (komori@smg.co.jp)

はじめまして！

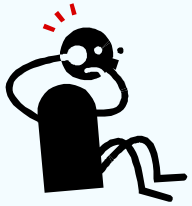
- 名前：小森 裕介
- **Blog**：http://d.hatena.ne.jp/y-komori/ 「こもりん日記」
- 所属：エスエムジー株式会社 (http://www.smg.co.jp)
- 主な仕事：
  - ✓ Javaによる集中監視フレームワーク設計・開発
  - ✓ Webアプリケーションシステムの設計・開発
  - ✓ 技術コンサルティング
  - ✓ 教育・各種執筆活動
    - » 日経ソフトウエア「とことん作って覚える! Java入門」連載
    - » 「なぜ、あなたはJavaでオブジェクト指向開発ができないのか」

- **Seasar2とのかかわり**
  - ✓ S2Containerコミッタ、S2JMSコミッタ、S2JFaceコミッタ

巷でやたらと耳にするDI×AOP・・・



なんとなく良さそうなんだけど、
なにが良いのかはっきりわからない・・・



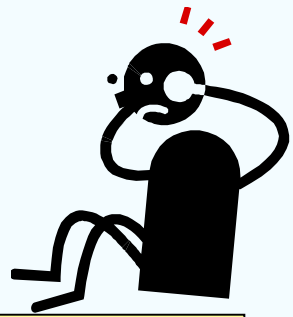
自分の開発現場にも導入したいのだけど、
上司や同僚を納得させる自信がない・・・

そんなアナタに

DI×AOPのオイシイところをお伝えします!

DI登場前夜

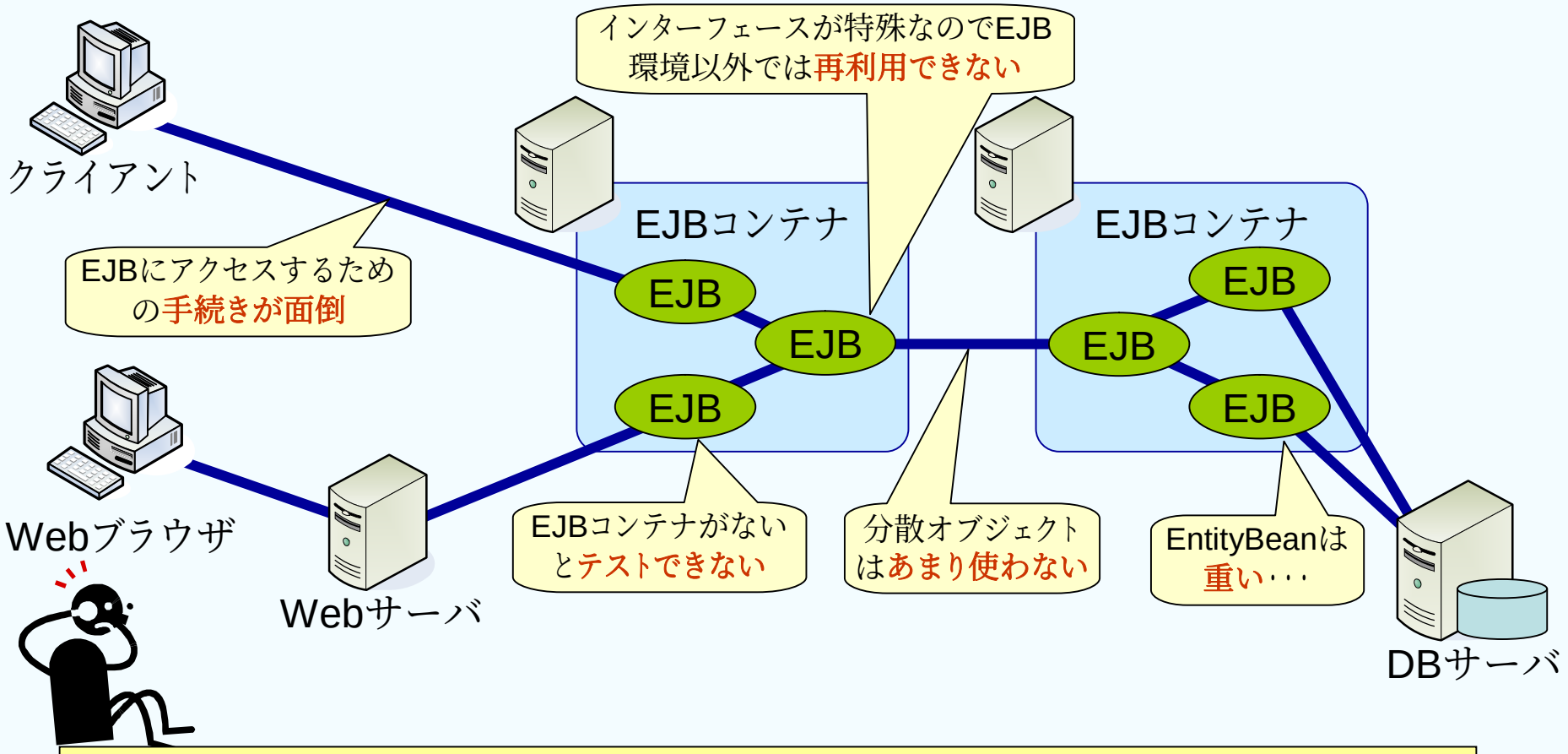
- SunはEJB2.0の利用を推進していた
  - ✓ プログラムをコンポーネント化して再利用できるようにするための技術
  - ✓ 思想自体は間違っていなかった
  - ✓ 問題点
    - » おそい　コードの記述量が多く開発に時間がかかる
    - » たかい　高価なAPサーバの導入が必要
    - » まずい　仕様が難しすぎて理解が大変



EJBはあまり利用されずStruts等、OSSの利用が広まった

EJB2.0の問題点

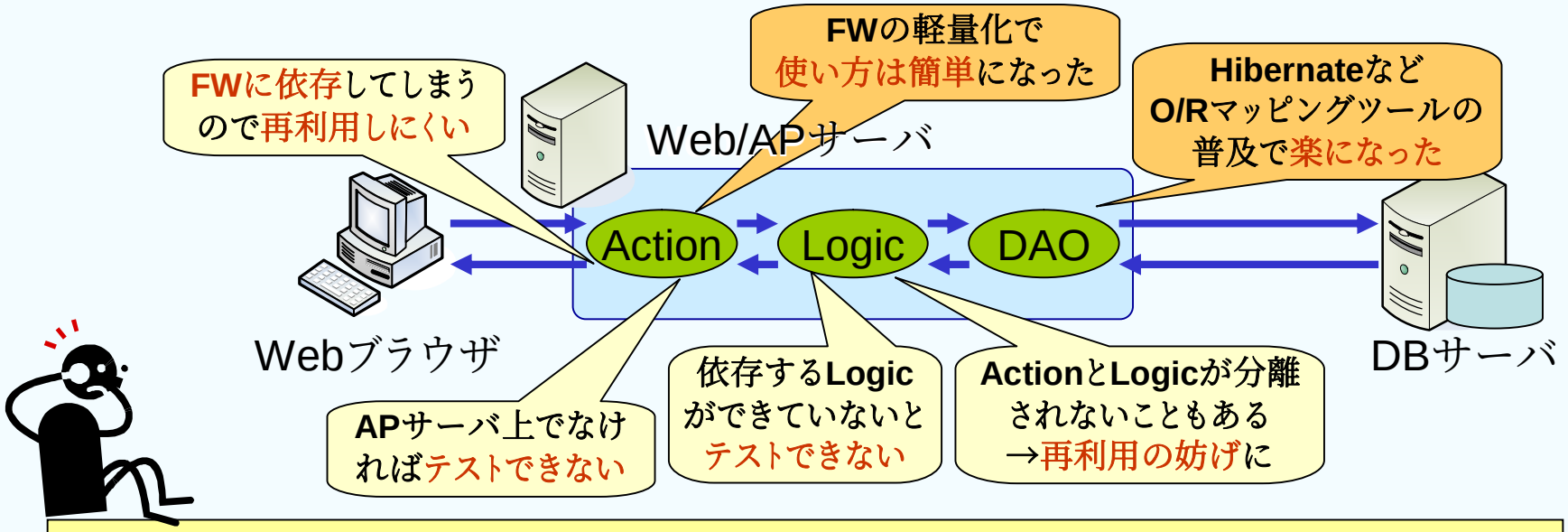
- 具体的には、なにが問題なの？



もっと簡単にコンポーネント技術を利用する方法はないの？

なぜこの状況が放置されたのか？

- Javaのエンタープライズ利用は Webアプリケーションが主流となった
  - → Struts をはじめとする「WebアプリケーションFW」が台頭。
    中・小規模のシステム開発では、
    EJBの提供する機能がなくてもあまり困らなかった



本質的な問題は未解決。大規模システムでは影響が顕著に。

DIコンテナの登場

- DIコンテナ(Seasar2)の特徴を簡単に言えば・・・
  - はやい
    - ✓ ソースコードの記述量が少なくてすむ
  - やすい
    - ✓ 高価なAPサーバは不要！
    - ✓ オープンソースなので導入コストはゼロ！
  - うまい
    - ✓ 使い方が簡単！
    - ✓ うまく利用すれば様々なところへ応用できる！

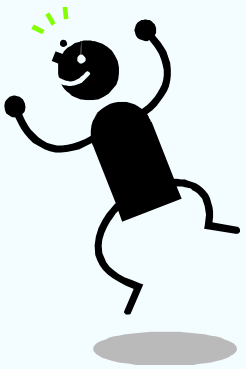
DIコンテナで何が楽になるのか

■生産性の向上

- ✓ コンポーネント間の依存関係が減るため、作業を分担しやすい
- ✓ 依存コンポーネントを準備するためのコードが不要になるため、コーディング量が減る

■保守・拡張性の向上

- ✓ コンポーネント間の依存関係を設定ファイルで記述するため、機能拡張時に他コンポーネントへ与える影響が少ない



■テスト効率の向上

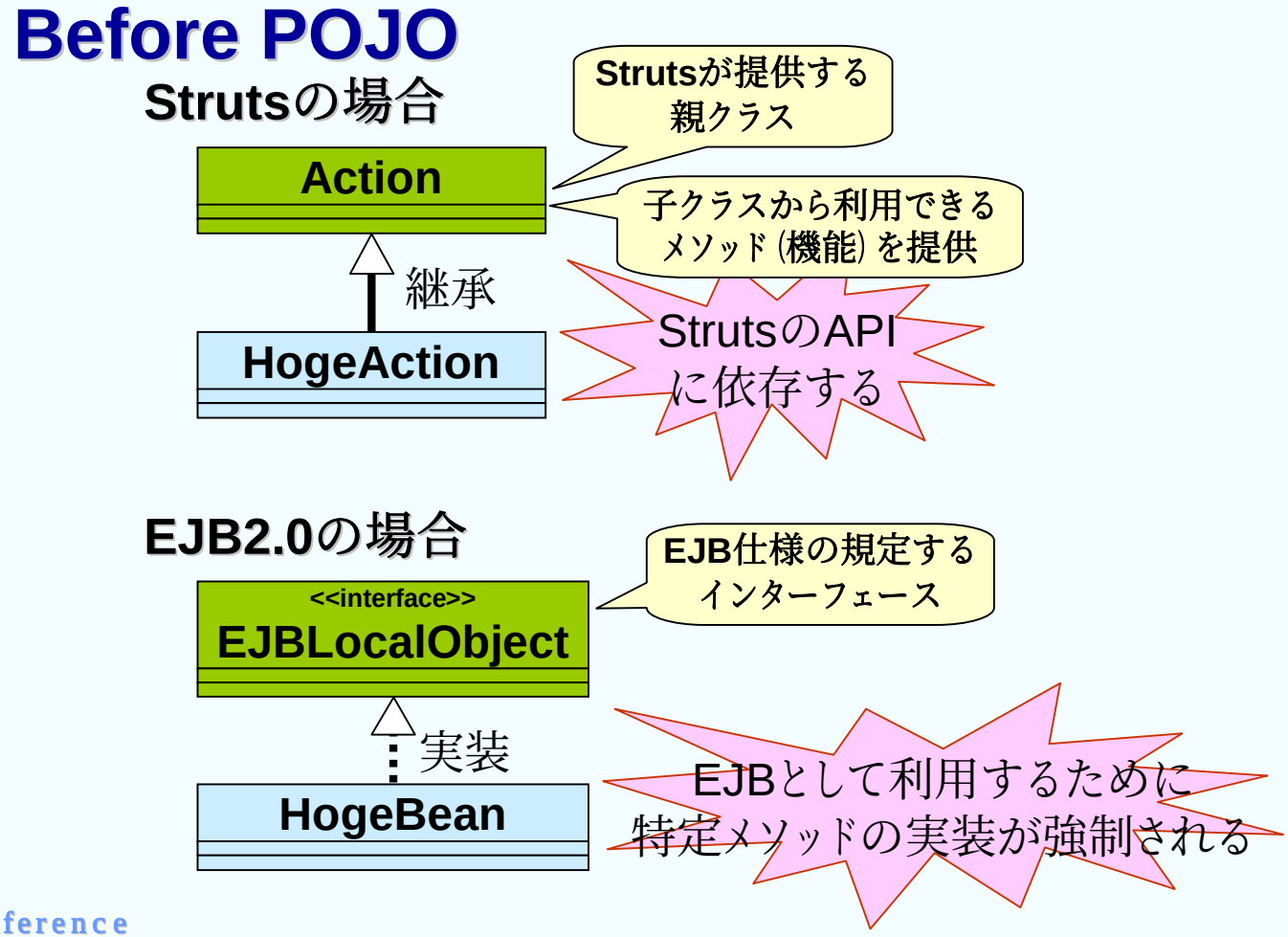
- ✓ 依存コンポーネントの完成を待たずにテストできる

■学習コストの低減

- ✓ コンポーネントが単純になるため、開発担当者が理解する範囲が少なくてすむ

DIという考え方 - 革新その1 - BeforePOJO

- POJO (Plain Old Java Object) という考え方
  - ✓ FW特有のI/F・親クラスを持たない、「昔ながらのただのJavaオブジェクト」



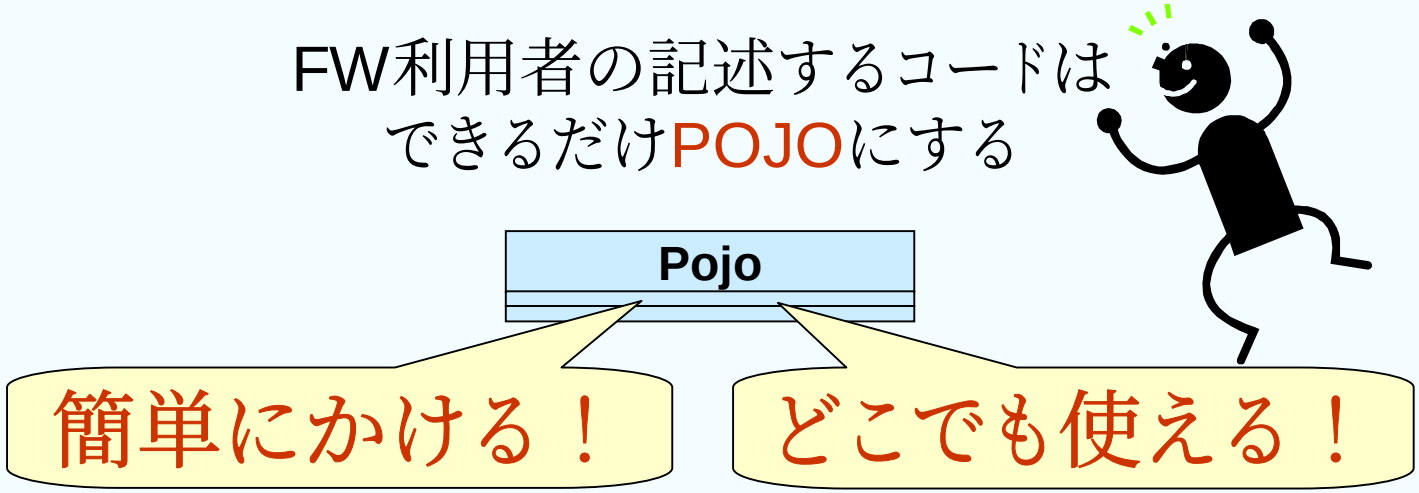
DIという考え方 - 革新その1 - AfterPOJO

- POJO (Plain Old Java Object) という考え方
  - ✓ FW特有のI/F・親クラスを持たない、「昔ながらのただのJavaオブジェクト」

AfterPOJO

POJOベースの設計

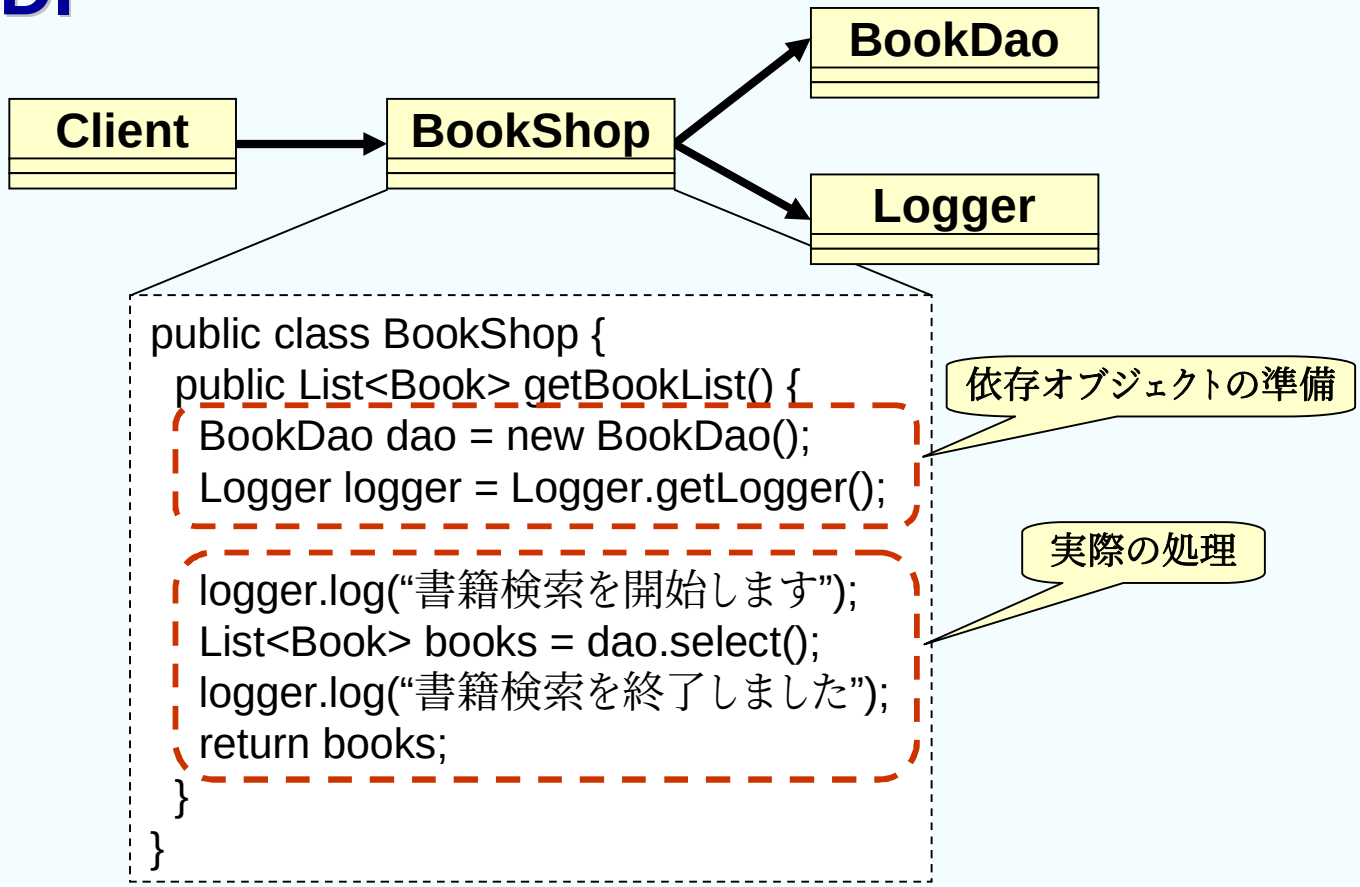
FW利用者の記述するコードは
できるだけPOJOにする



DIという考え方 - 革新その2 - BeforeDI

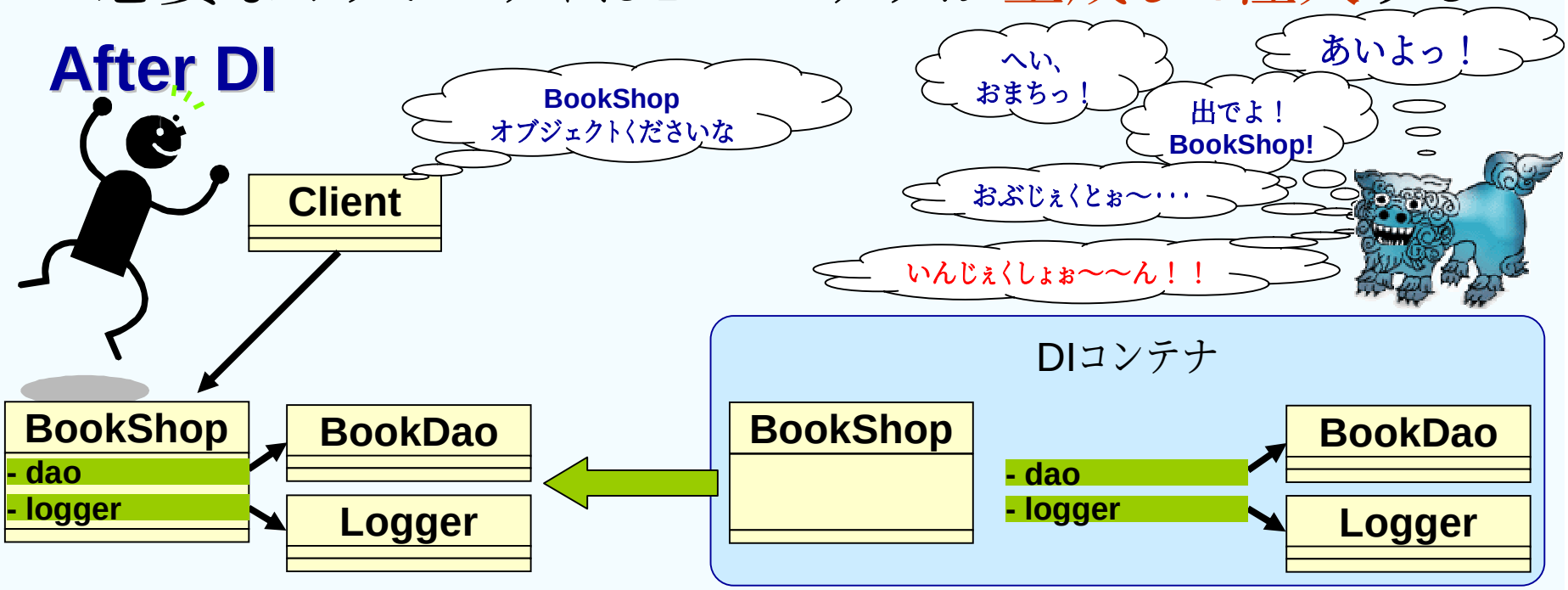
- 依存性注入(Dependency Injection)という考え方
  - ✓ 必要なオブジェクトはDIコンテナが生成して注入する

Before DI



DIという考え方 - 革新その2 - AfterDI

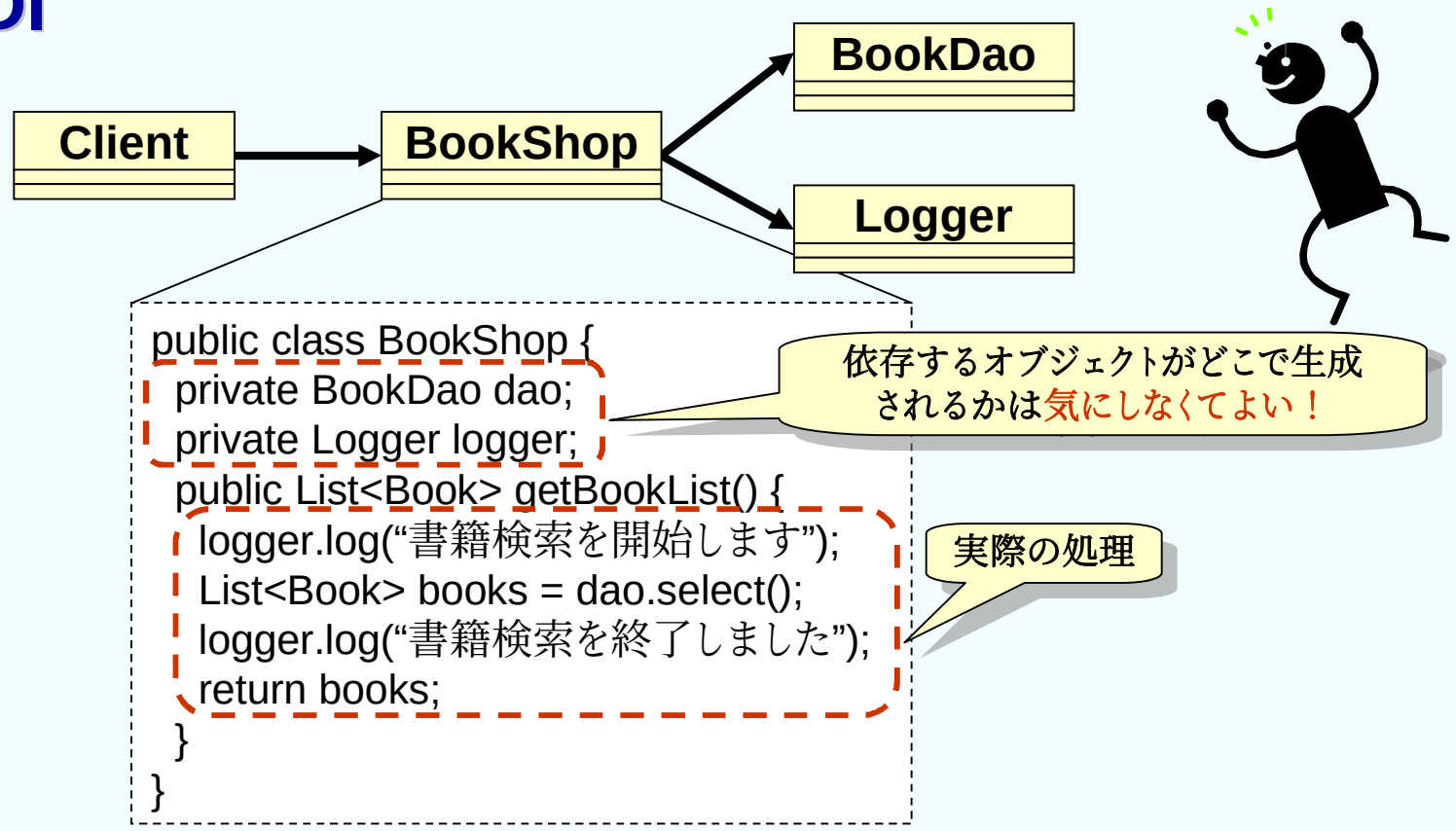
- 依存性注入 (Dependency Injection) という考え方
  - ✓ 必要なオブジェクトはDIコンテナが生成して注入する



DIという考え方 - 革新その2 - AfterDI

- 依存性注入 (Dependency Injection) という考え方
  - ✓ 必要なオブジェクトはDIコンテナが生成して注入する

After DI



DIコンテナによる生産性の向上(1)

- 生産性を向上するには、開発を分担すること
  - ✓ 開発を分担するには、インターフェースと実装を分離すること

なかなか実現できなかった・・・

依存先オブジェクトを生成する際、実装を意識せざるを得なかったから

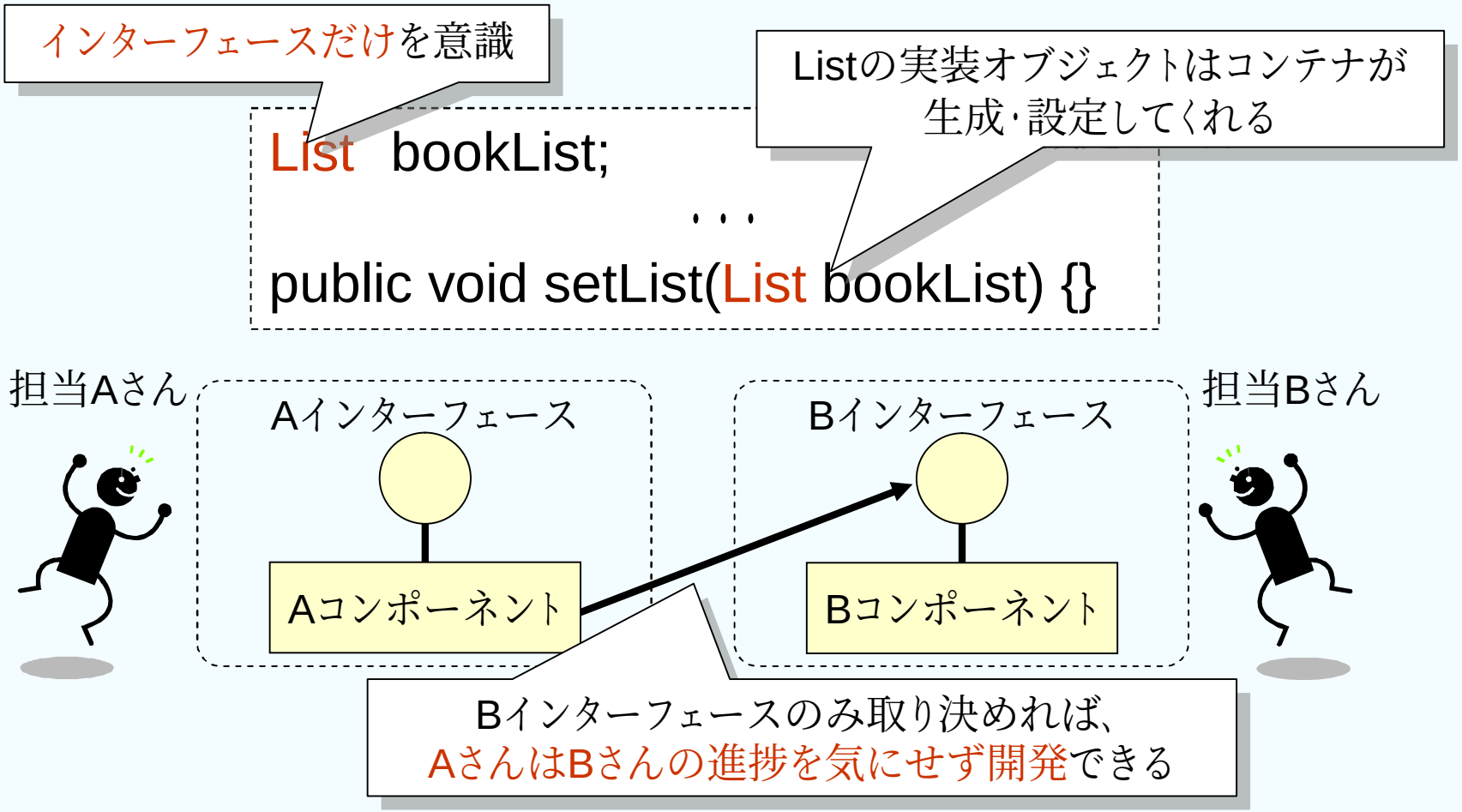
```
List   bookList = new ArrayList();
```

せっかくインターフェースだけを
意識しても・・・

オブジェクト生成の際、
どの実装クラスにするかを意識する

DIコンテナによる生産性の向上（2）

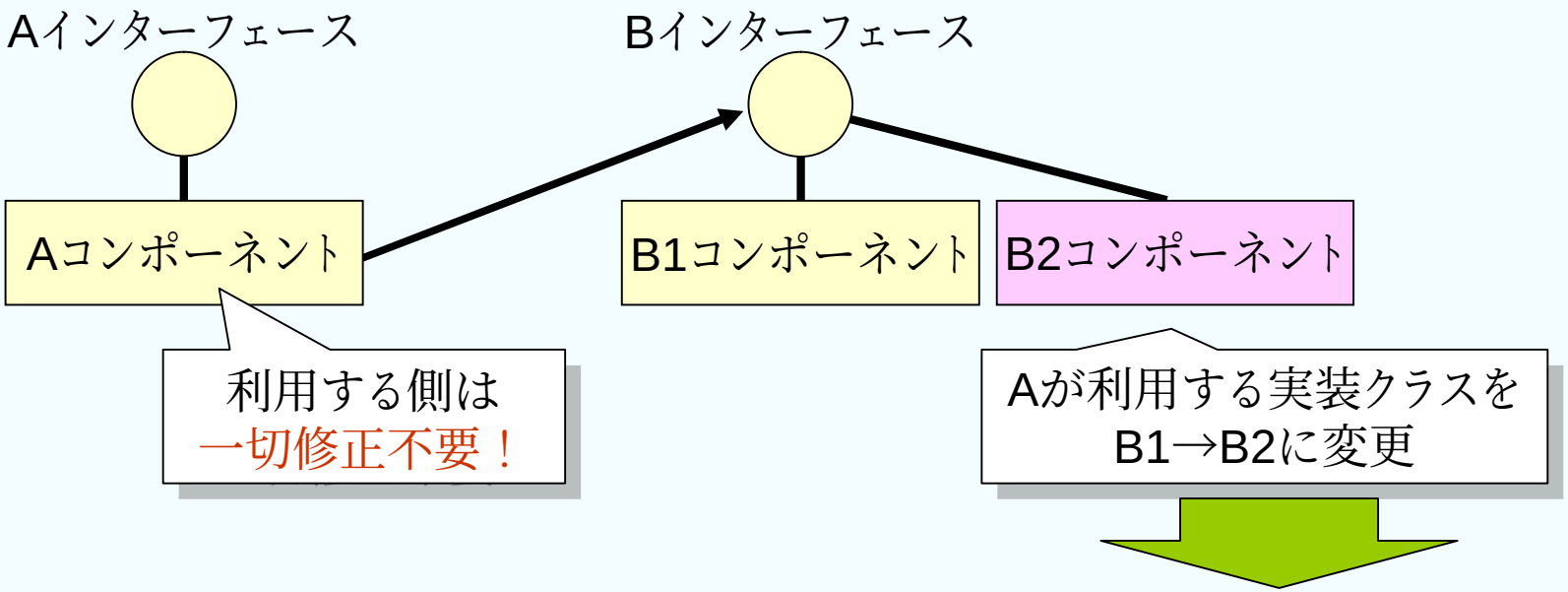
■DIでインターフェースと実装が完全分離できる



開発分担しやすく、生産性向上！

DIコンテナによる拡張性の向上

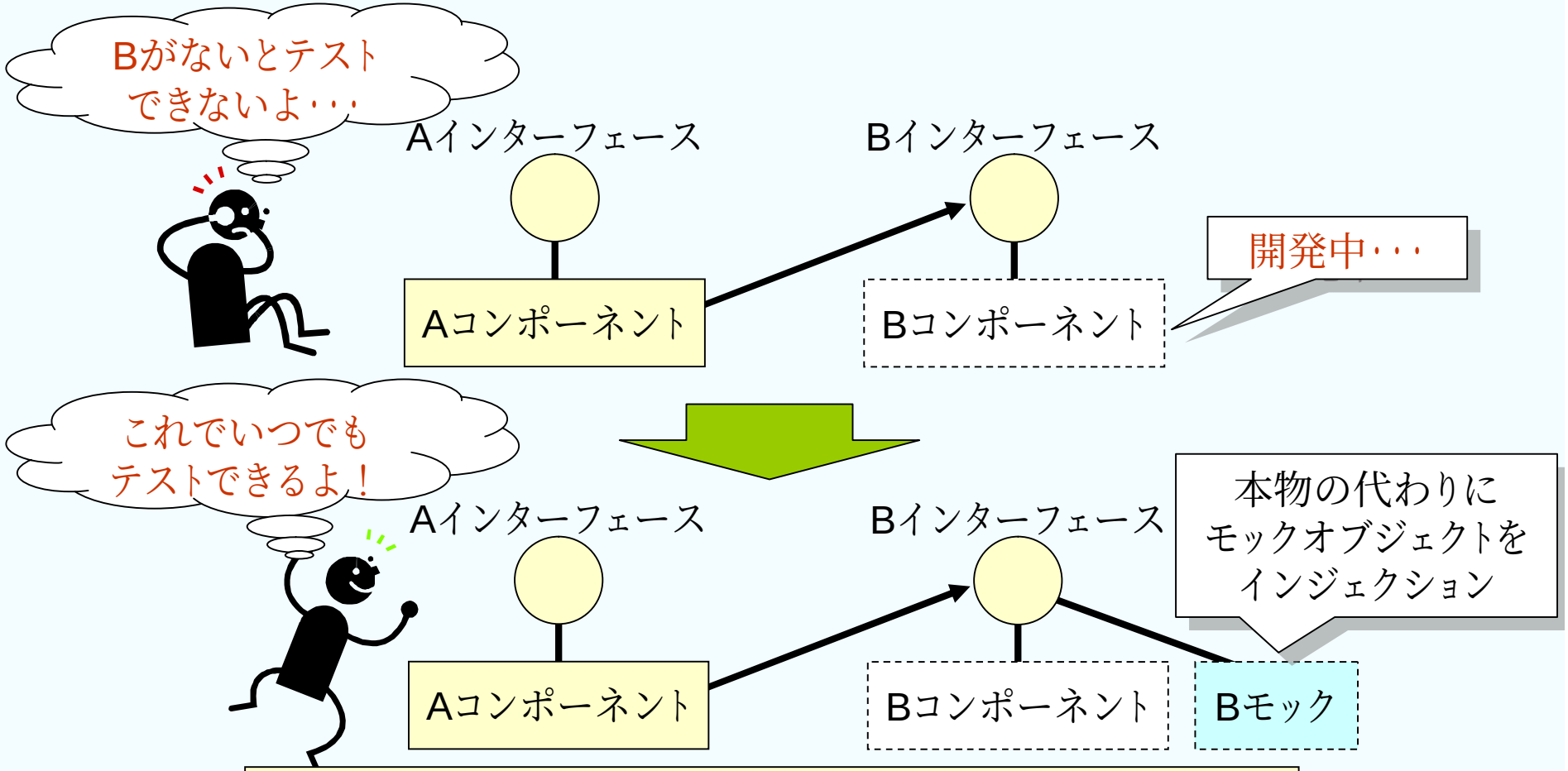
- 拡張や保守で実装クラスを変更する場合、依存元コードを一切修正せずに変更できる



最小の影響範囲で修正、拡張性向上！

DIコンテナによるテスト効率の向上

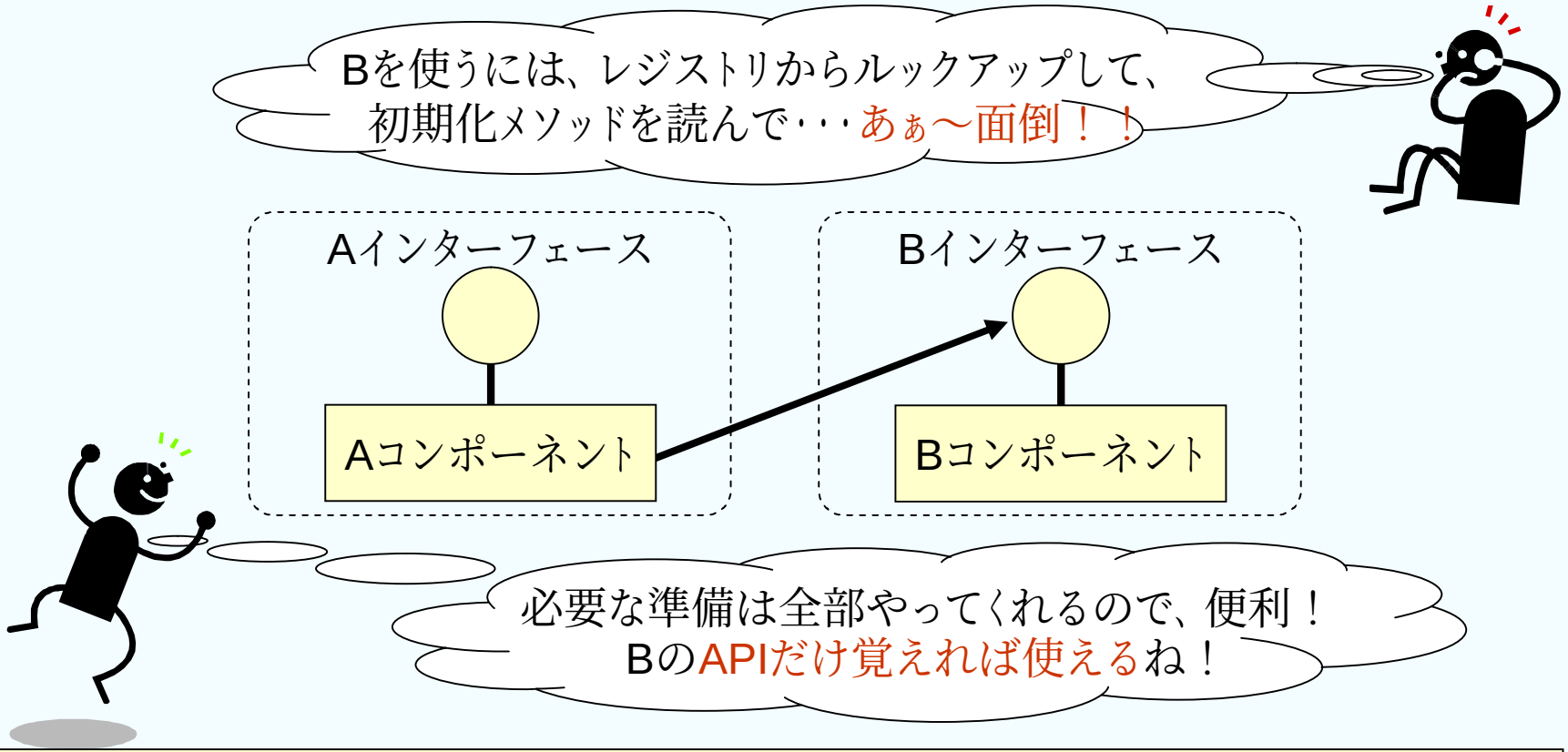
■依存先不在によるテスト効率の低下を改善



モックの利用で、テスト効率向上！

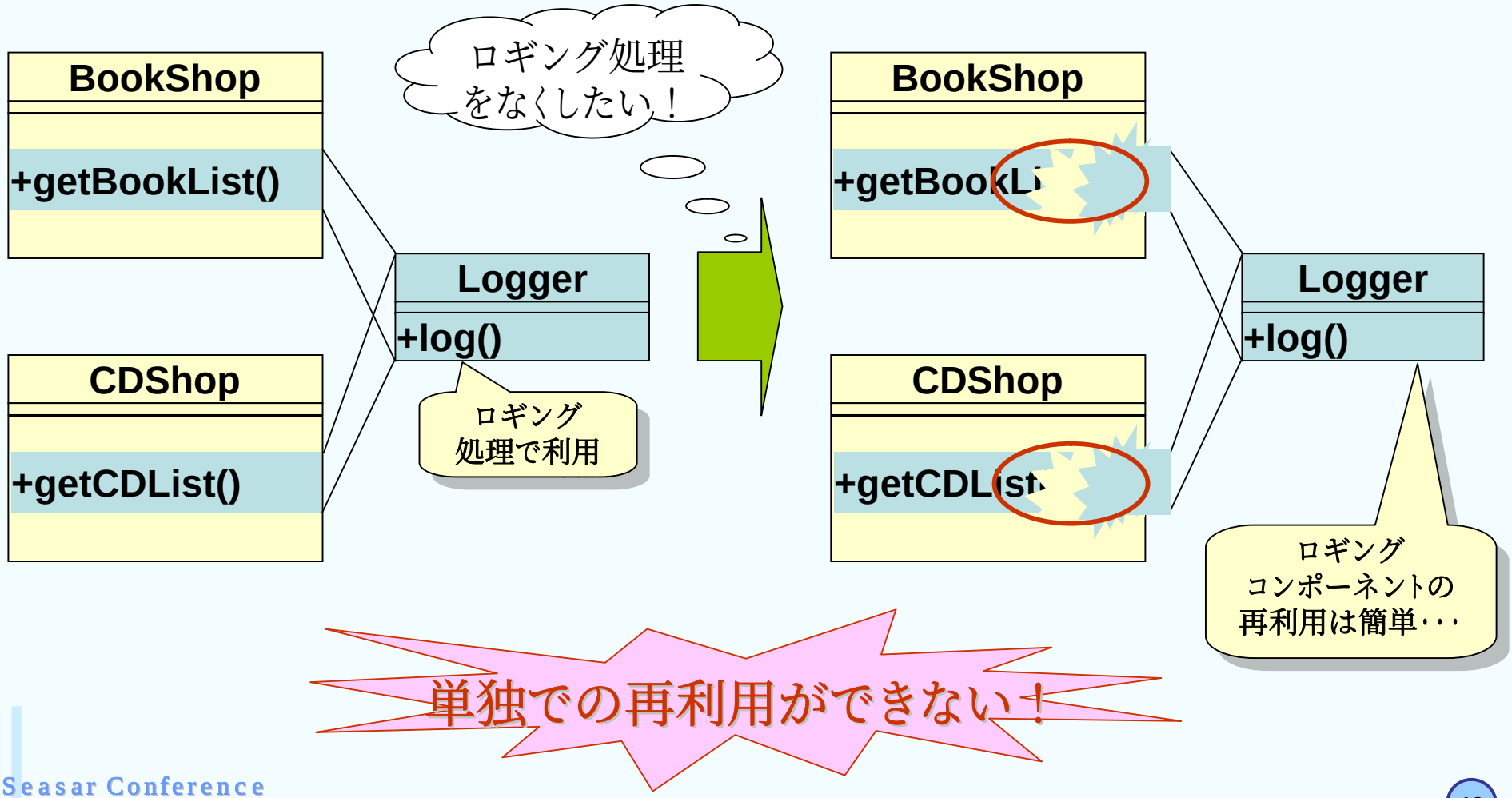
DIコンテナによる学習コスト低減

- 依存先オブジェクト取得のための面倒な手続きが不要
  - ✓ インターフェースさえわかれば、使えるようになる



面倒な準備はコンテナまかせ、学習コスト低減！

■例:ロギング処理で発生する問題

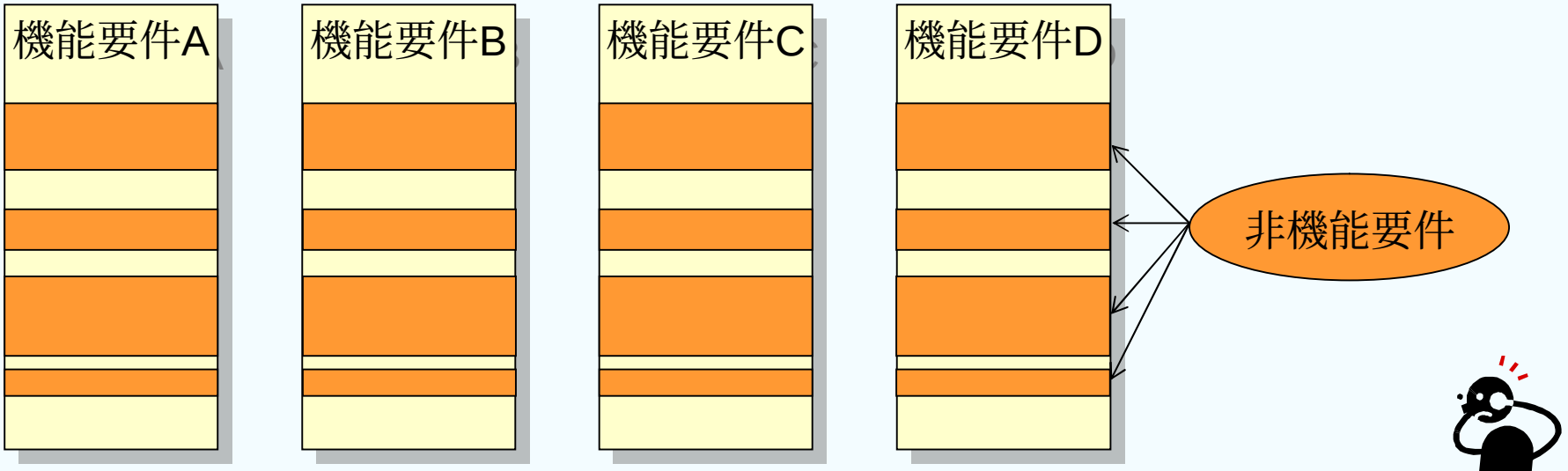


非機能要件はモジュール単独分離が難しい

- 非機能要件に関する処理は、全機能に影響するため、モジュールとして分離しにくい

➡ ロギング処理も非機能要件の一つ

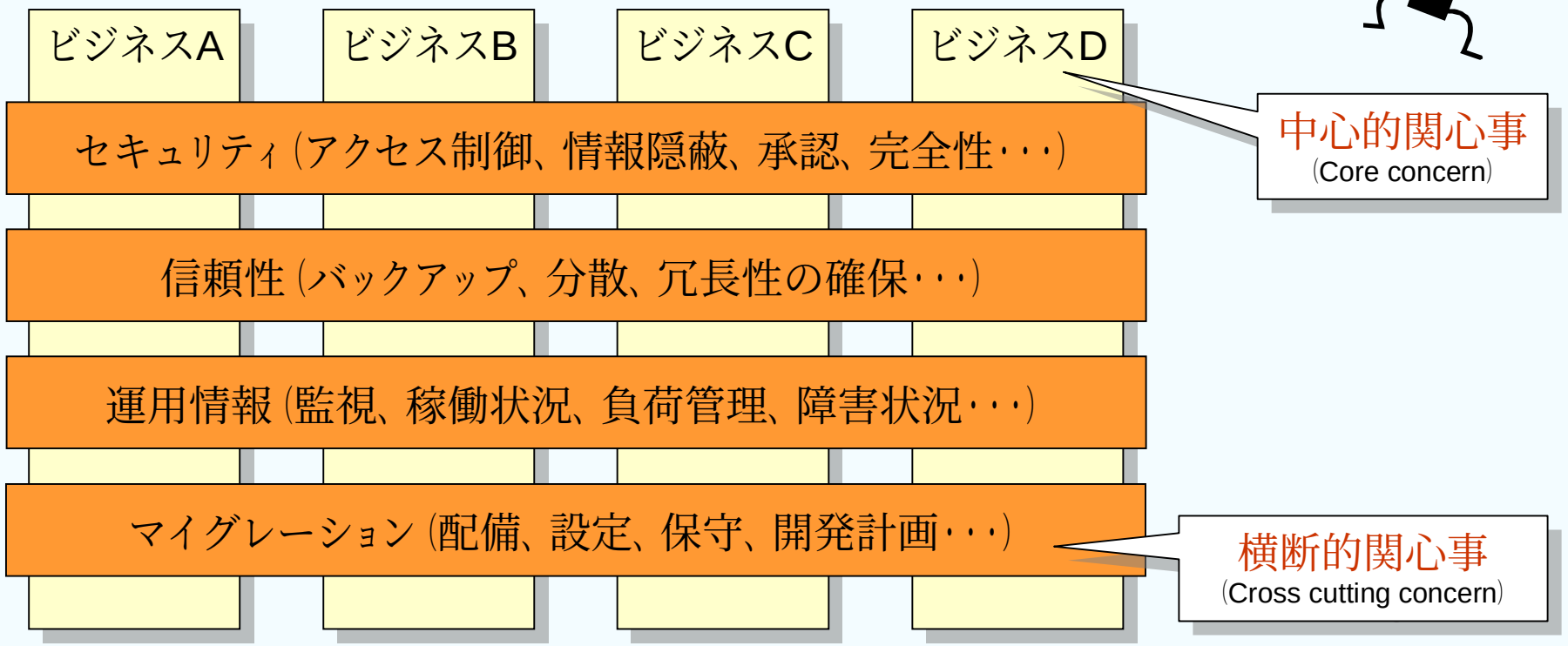
非機能要件・・・システムの本来の機能とは関係ないが、信頼性や保守性、使いやすさを向上させるための要件



各機能に共通する処理をモジュール化する方法はないの？

アスペクト指向は関心事の分離から

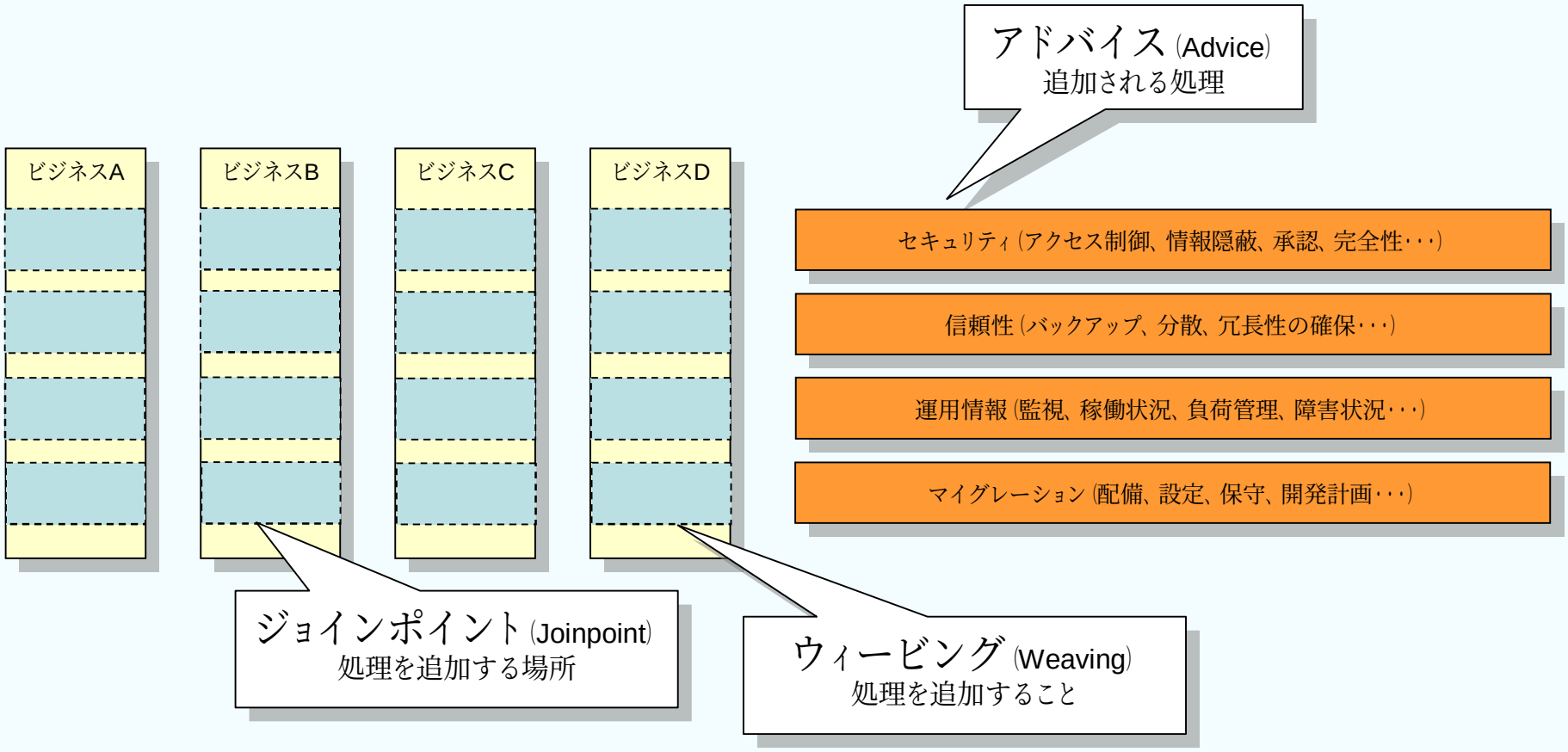
- システムを2種類の要件に分け、横断的関心事を分離するのがアスペクト指向の考え方



【出典】『Seasar2で学ぶDIとAOP』（arton著、技術評論社）p20

ジョインポイントとアドバイス

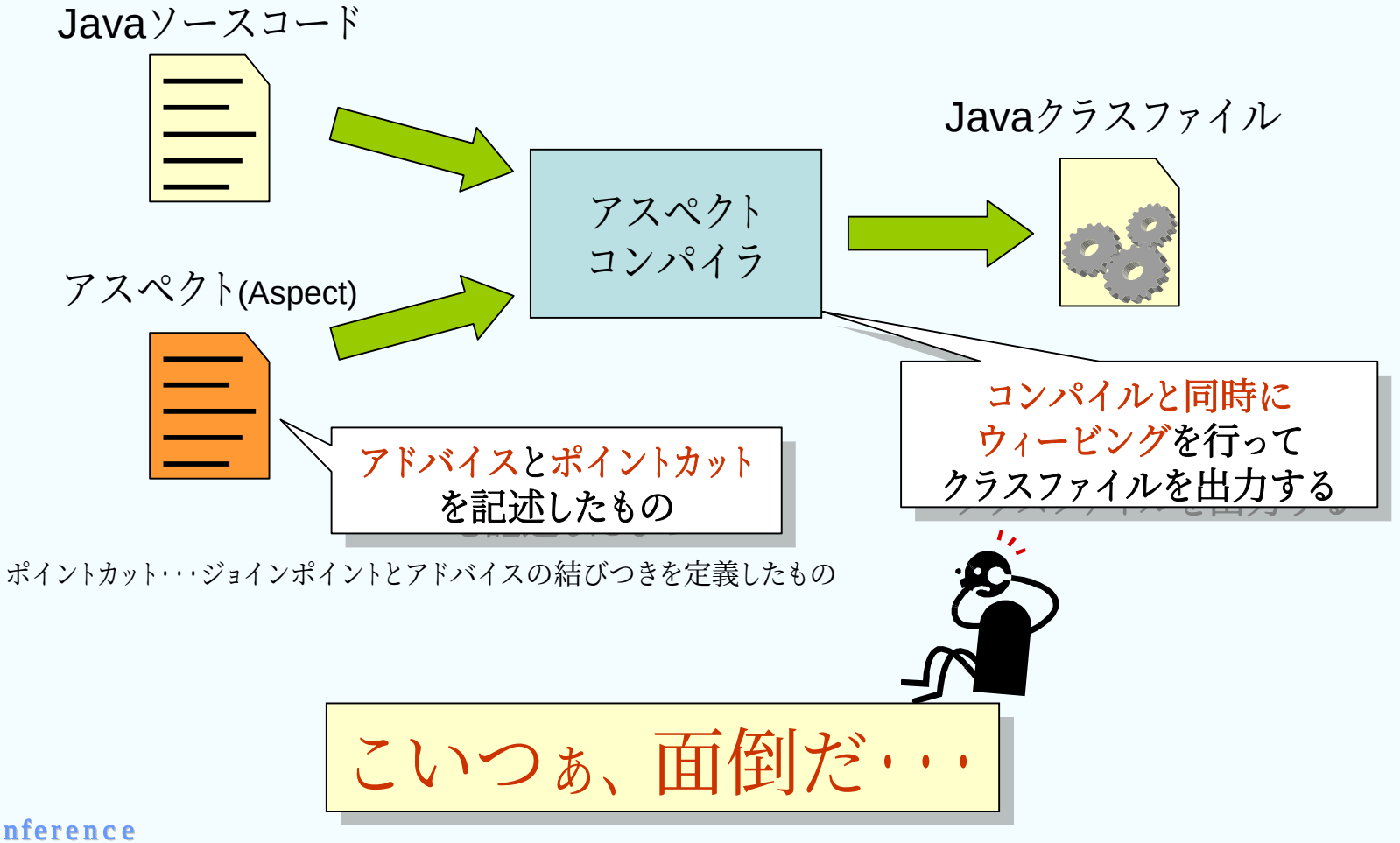
- アスペクト指向では、機能の中にジョインポイントを定義し、アドバイスをウィービングする



なぜDI×AOPが嬉しいのか

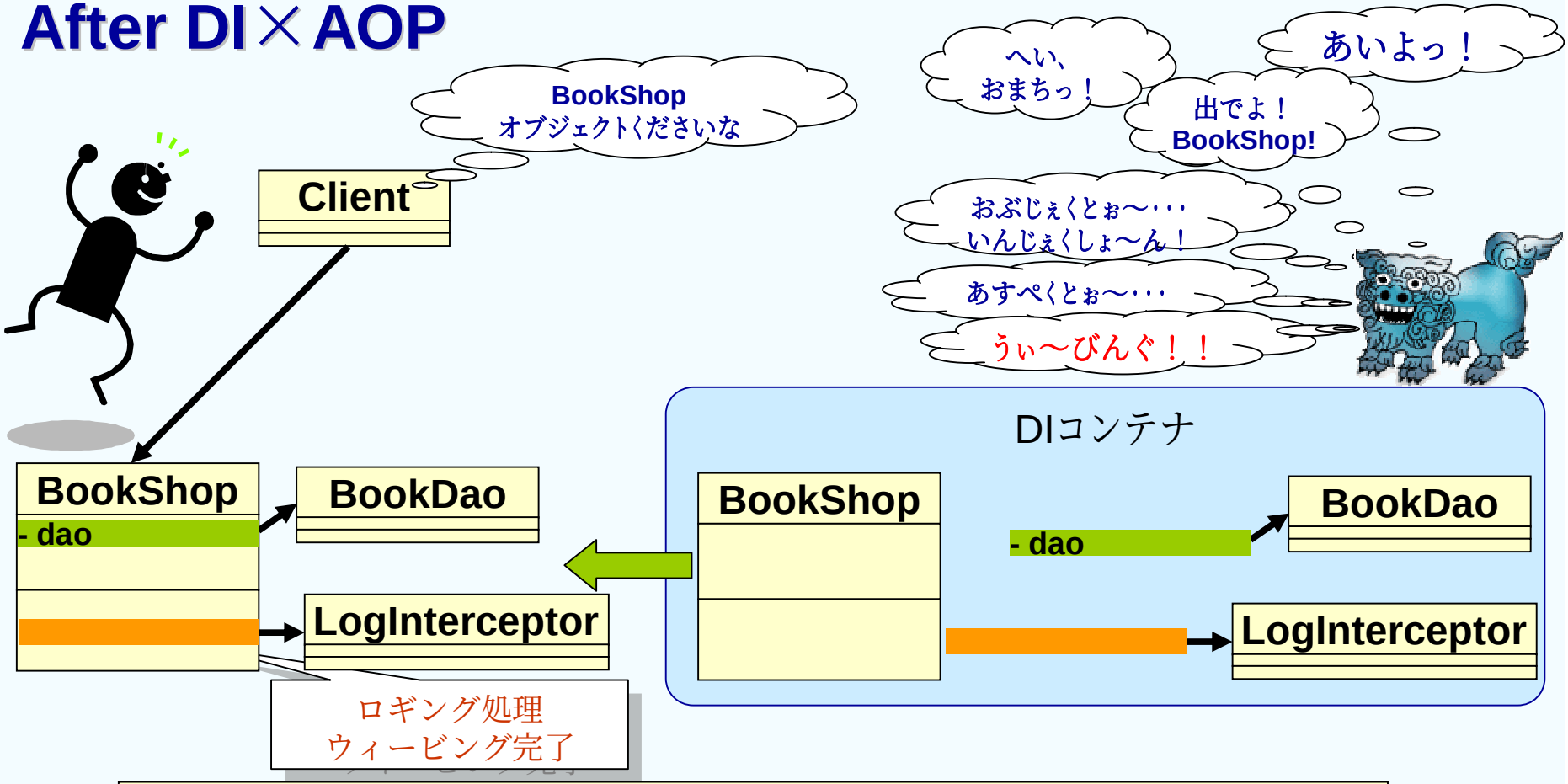
- 当初、AOPの実現には専用のコンパイラが必要だった

Before DI×AOP



なぜDI×AOPが嬉しいのか

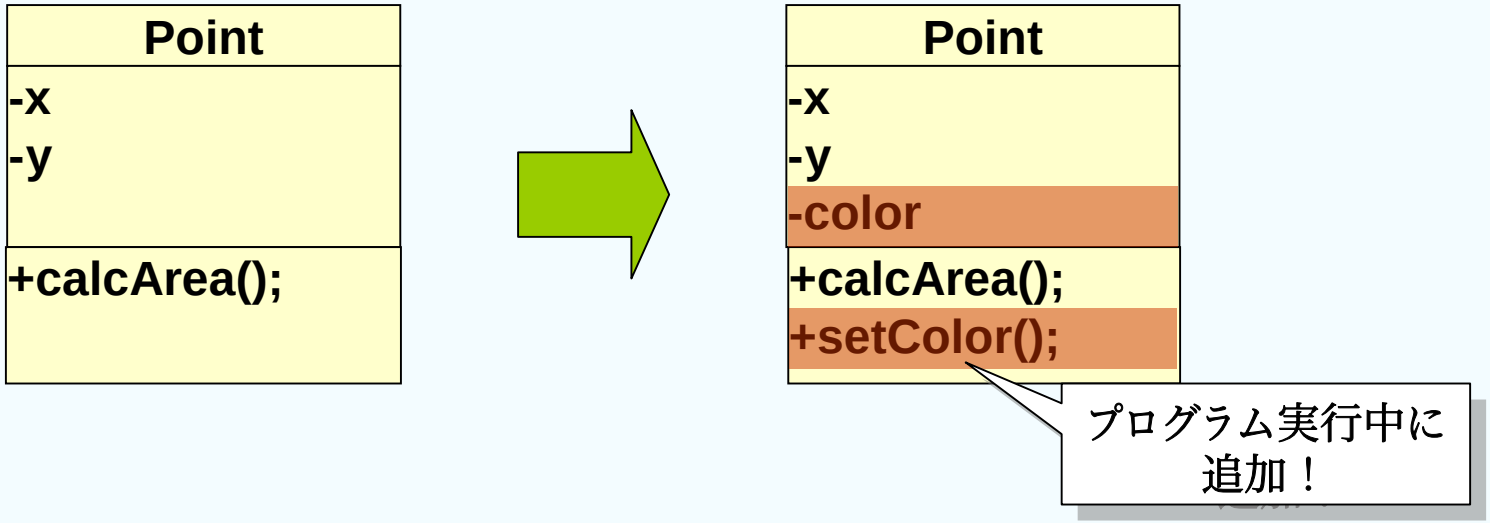
■DIコンテナでウィービングも同時に行う



DI×AOPで自動ウィービングが可能に！

さらに強力なDI×AOP - InterType

- インタータイプ宣言とは
  - ✓ 既存のクラスの静的構造を変更するアスペクトの機能
- Seasar2.4で提供される新機能InterType
  - ✓ 既存のクラスに対してメソッドやフィールドを自由に追加可能



ソースコード自動生成に替わる手段として利用可能！

More Info! http://s2container.seasar.org/ja/aop.html#OriginalInterType

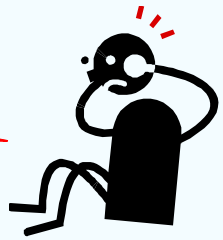
DIコンテナに対する5つのギモン

- 依存関係はどうやって判断するの？
  - ✓ Seasar2なら、インターフェースに基づいて自動判断します
    More Info! http://s2container.seasar.org/ja/DIContainer.html#AutoBindingMode
- 結局設定ファイルをたくさん書くのでは？
  - ✓ Seasar2なら、AutoRegisterでカンタンに登録！
    More Info! http://s2container.seasar.org/ja/DIContainer.html#ComponentAutoRegister
- 設定ファイルのデバッグが大変？
  - ✓ Kijimuna (キジムナ) で記述の誤りをチェックできます！
    More Info! http://kijimuna.seasar.org/
- 結局はリフレクションでしょ、遅くないの？
  - ✓ 独自のキャッシュ機構で速度低下はほとんどありません
  - ✓ Seasar2はSpringと比較して2～300倍高速です
    More Info! http://s2container.seasar.org/ja/benchmark/20060412_seasar_vs_spring.pdf
- **Setterを書くのが面倒くさい！**
  - ✓ PropertyInterType で setter いらず！
    More Info! http://s2container.seasar.org/ja/aop.html#AOPInterType

どうやって使っていったらよいの？

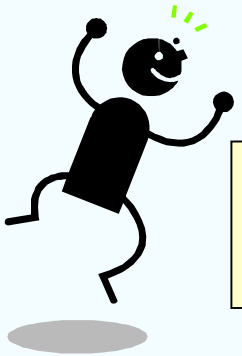
■DIコンテナのメリットはなんとなくわかった・・・

でも、
開発現場で
どう役立てればいいの？



DIコンテナの機能を
フルに使い切って設計するのは、
けっこうムズカシイ！

そこで・・・



まずは、S2Containerを100%生かして作られた
周辺プロダクトを使ってください！

まずは、S2Dao、Teedaがオススメです！

本日のまとめ

- DIコンテナの登場した背景を理解した
- DIコンテナの4つのメリットを理解した
  - ✓ 生産性の向上
  - ✓ 拡張性の向上
  - ✓ テスト効率の向上
  - ✓ 学習コストの低減
- AOPの必要性について理解した
- DI×AOPがなぜ便利かを理解した

Seasar2・AOPについて、もっと知りたい方へ

- 『Seasar入門　～はじめてのDI&AOP～』
  - ✓ 監修：ひが やすを　著：須賀 幸次 他
  - ✓ 価格：3,570円
  - ✓ 出版社：ソフトバンククリエイティブ
  - ✓ ISBN：4797331968

- Seasar2で学ぶDIとAOP アスペクト指向によるJava開発
  - ✓ 著：arton
  - ✓ 価格：3,360円
  - ✓ 出版社: 技術評論社
  - ✓ ISBN: 4774128554

- 『アスペクト指向入門』-Java・オブジェクト指向からAspectJプログラミングへ
  - ✓ 著：千葉 滋
  - ✓ 価格：￥2,480＋税
  - ✓ 出版社：技術評論社
  - ✓ ISBN：4-7741-2581-4