J2EE/DI/アスペクト勉強会

ファースト・シーズンのまとめ

本日のお品書き

- 早わかりDIxAOPコンテナ
- J2EE/DI/Aspect勉強会
- 問題(?)

早わかりDIxAOPコンテナ

- 早わかりDI
- 早わかりAOP
- 早わかりDIxAOPコンテナ
- DIxAOPコンテナを取り巻くプロダクト
- DIxAOPを利用する利点

早わかりDI

- 開発者がセットしなくとも、利用するコンポーネントが自動的にセットされること

```
public class ServiceImpl {
  private Dao dao;
  public setDao(Dao dao) { this.dao = dao; }
  public List find() { return dao.find(); }
}
```

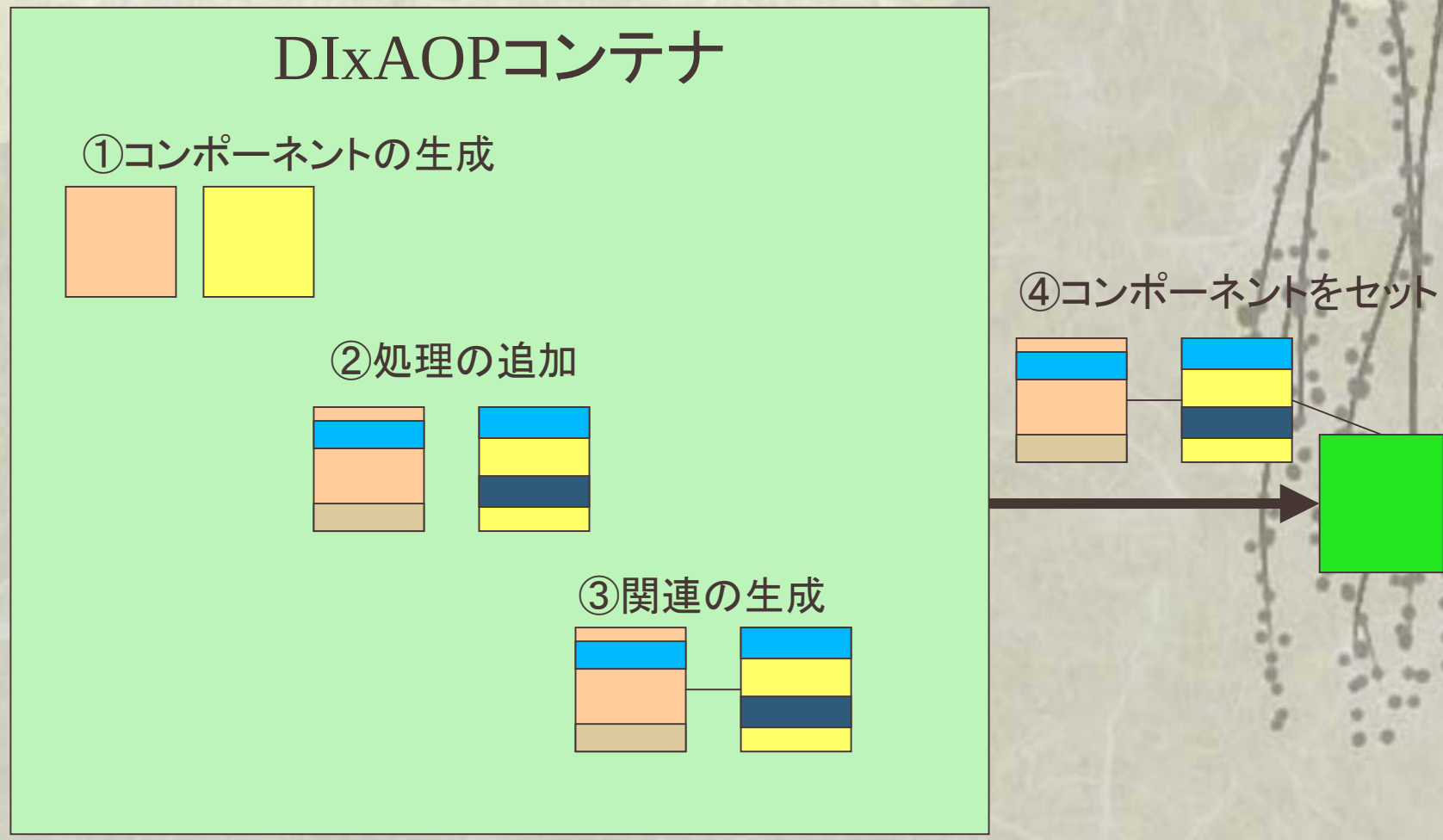
早わかりAOP

- 今あるコードに全く影響を与えないで、他のコードを追加すること

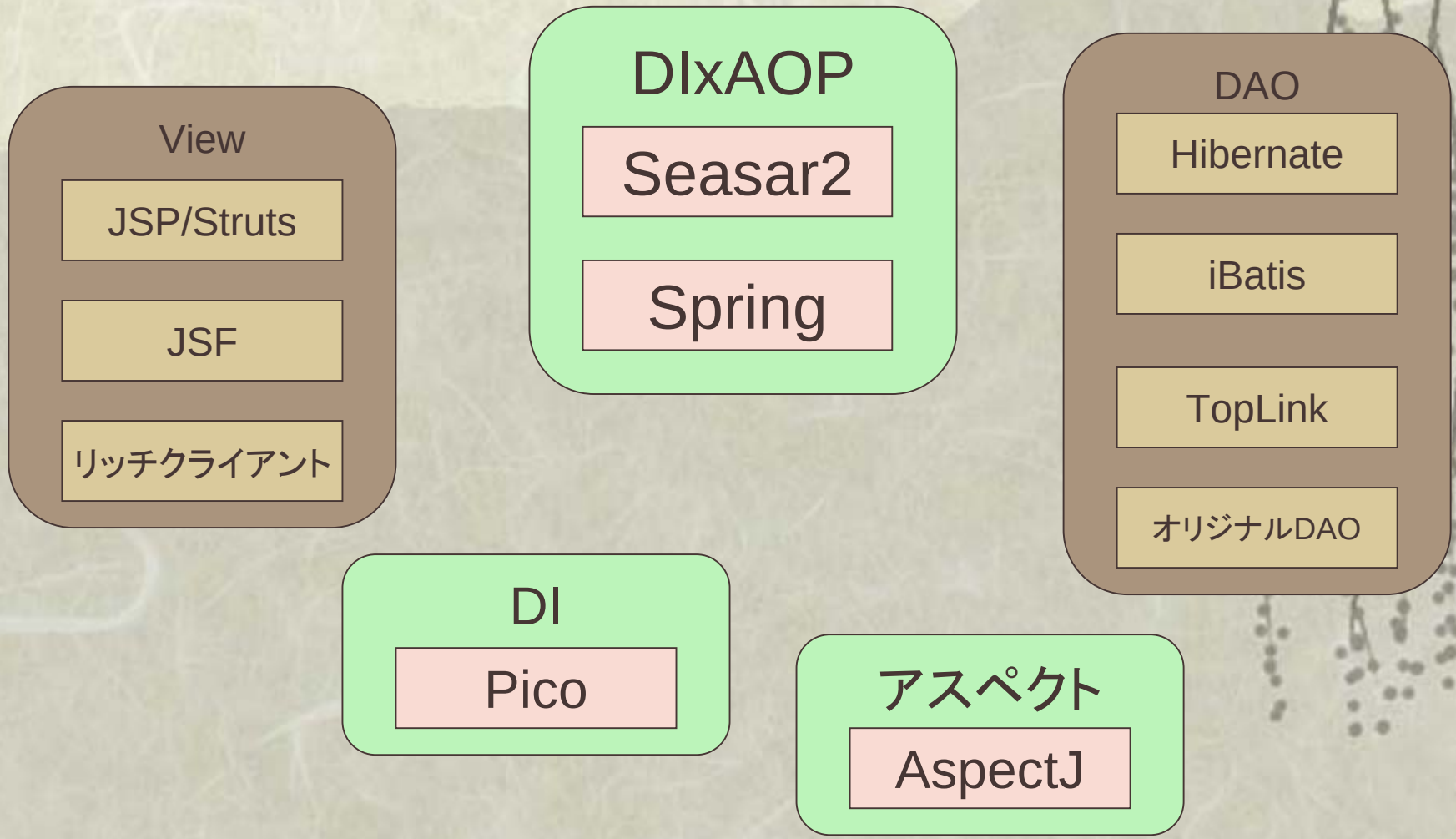
```
public class Calc {
  public static void main(String[] args) {
    int ret = 100 + 100;
  }
}
```

```
>java Calc
Ans : 200
```

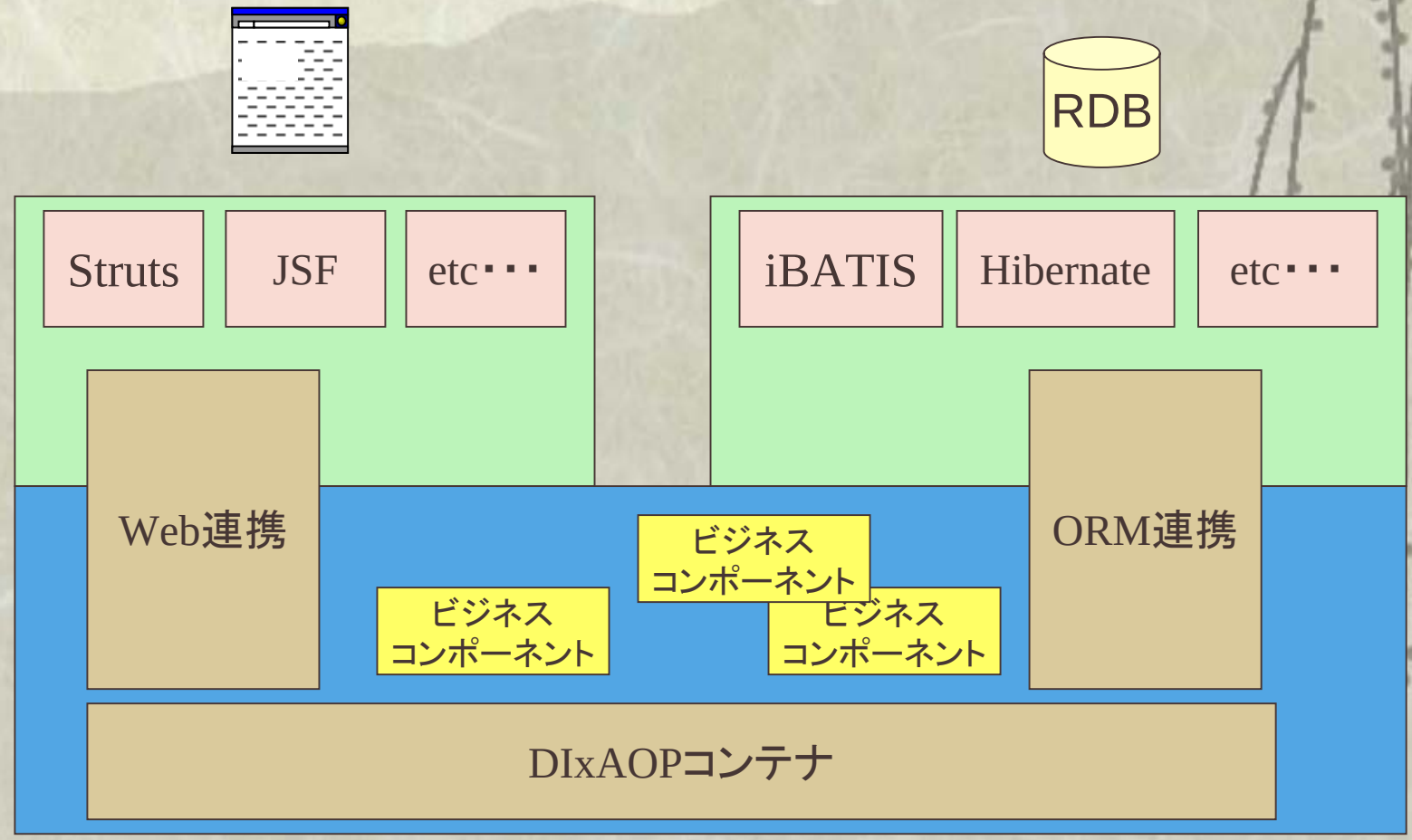
早わかりDIxAOPコンテナ



J2EE/DI/Aspectの現状



DIxAOPコンテナを取り巻くプロダクト



J2EE/DI/Aspect勉強会

- 活動の概要
- 第1回勉強会　活動報告
- 第2回勉強会　活動報告
- 第3回勉強会　活動報告
- 今後の活動

活動の概要

- 2005年度総会、2005年6月1日、早稲田大学
- 第1回勉強会、2005年7月14日、豆蔵
- 第2回勉強会、2005年8月18日、豆蔵
- 第3回勉強会、2005年9月22日、豆蔵

テーマ

- 最初の3ヶ月におけるテーマ案
  - 本当に良いの？DI/AOPで本当に保守性や拡張性が上がるの？根拠は？
  - 定義ファイルの管理は？複数のチームで開発する時とかどうするの？
  - DIコンテナ（+ AOP）の利用～成功パターン、アンチパターンは？
    - 問題領域としてのドメイン
    - 設計
    - 管理
  - スキルセットは？OOの知識とかいるの？
  - アスペクトのデザインパターン
  - ツールを調べよう どんな便利なツールがあるの？
  - 他の言語（.NET、Rubyなど）と比べてみよう

第1回勉強会
DIコンテナの成功パターン・アンチパターン

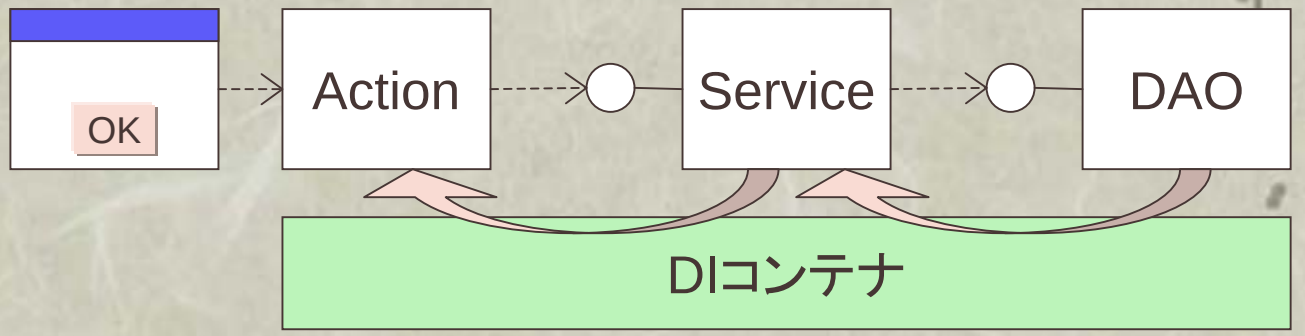
- DIコンテナを使った場合の設計
  - 成功パターン、アンチパターンを探る
- 背景などの解説から
- 参加者全員でディスカッション

背景

- Transaction Script VS. Domain Model
  - 現在のJ2EEシステム（主に帳票ベースのWebアプリケーション）はTransaction Scriptが多い
    - DIで実現している・いないに関わらずTransaction Scriptが多い
  - DIコンテナはDomain Modelにも向くのか？

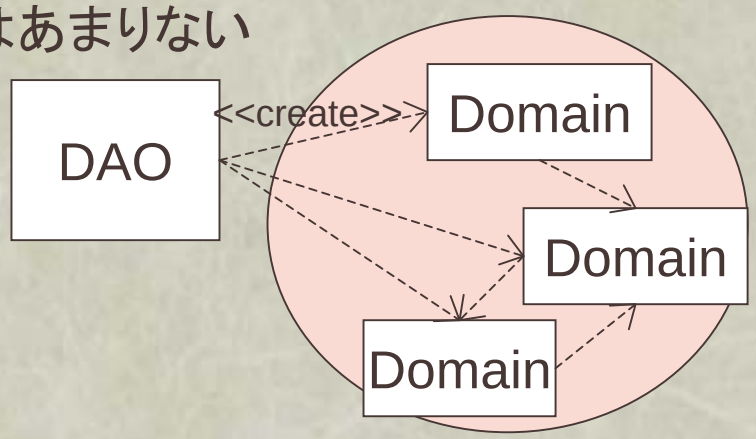
Transaction Script

- Transaction Script
  - 振る舞いのみの（ステートレスな）オブジェクトと値のみを持つオブジェクトで構成される
  - 現在、DI（コンテナ）で実現しているのはステートレスなオブジェクトをつなぐこと
- DIで実現するTransaction Script
  - ロジックの変更が容易
  - テストが容易
  - 開発のチーム分けが容易



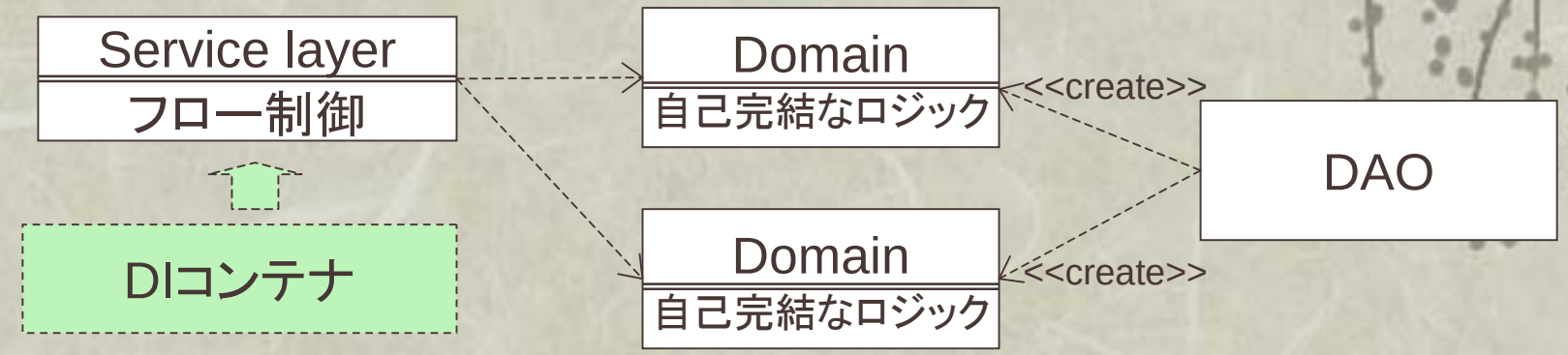
Domain Model

- Domain Model
  - 属性と振る舞いを持ったオブジェクトがメッセージパッシングを行ってシステムを実現
- 現在、J2EEで実現している主なシステム（帳票ベースのWebアプリケーション）では…
  - 属性を持つオブジェクトはDAOで生成する→DIコンテナで管理できない
  - 帳票ベースのWebアプリケーションではDomain Modelで持たせるべきメソッドはあまりない



可能性として

- AspectJのような、静的/クラスローディング時でのクラス定義変更が、DIコンテナにおけるオブジェクト生成時で同様に可能であればDomain Modelでも変更の容易性は保たれるかも（アスペクトの可能性）
- 自己完結したプリミティブなロジックのDomain Modelとフロー制御するService layerであれば、フロー制御の切り替え，変更にコンテナを利用するメリットがあるかも
  - 自己完結するロジックもTransaction Scriptに含めてしまった方が理解し易く一貫性がある？



結論

- DIコンテナでTransaction ScriptもDomain Modelも利用可能
- ただし、現状のシステム開発でDomain Modelが必要なケースは少ない
- またDomain Model に、DIコンテナを適用するメリットはTransaction Scriptよりもかなり低い
- Domain Modelの変更容易性向上にアスペクトを利用するのは如何？今後要調査

第1回勉強会
アスペクトの成功パターン・アンチパターン

- DIコンテナの設計に引き続いて、参加者全員でディスカッション

成功パターン・アンチパターン

- 成功パターン
  - トランザクション
  - ロギング
    - Traceなど横断的なもの。Debugには利用しない
  - 例外ハンドリング
- アンチパターン
  - なし
- 今のところの結論
  - まだあまり使われていないので、アンチパターンがない
  - あるいは、上記成功パターン以外でアスペクトを使おうとすることが、アンチパターン

結論

- アスペクトはトランザクション，ロギング，例外ハンドリング以外はアンチパターン（今のところ）
- アスペクトこそパターンがないと危険
  - 何でもできてしまうアスペクト
  - アスペクトをどう利用するのかを含め、今後要調査

第2回勉強会
AOSD

- JacobsonらのAspect-Oriented Software Development (AOSD)について
  - 太田さんから解説
- ・・・

第3回勉強会
定義ファイルの分割

- 定義ファイルをどうやって分割するのが良いか
  - 前提は大規模な開発
  - チーム分けも考慮する
- ５〜６名×４チーム
  - チーム内で議論
  - 結果発表

定義ファイルの分割（1）

- グループ1
  - アクタによる分割と、レイヤによる分割のハイブリッド。
  - アクタによる分割では、複数のユースケースに対応する機能が1つの定義ファイルにまとめられる可能性の問題あり。
- グループ2
  - 3階層に分ける。1階層目は機能単位、3階層目はコンポーネント・部品単位。間の2階層目は、include中心の対応付け・マッピング用の層（ほかを includeするためだけの定義ファイルを2階層目におく）。
  - CSSによる定義ファイル管理の仕方を応用した。

定義ファイルの分割(2)

- グループ3
  - DAO設定関連をそれぞれ独立した定義ファイルにし、他はユースケースに対応する部分単位で定義ファイルにまとめる。
- グループ4
  - ビジネスロジック層は機能グループ単位で複数に分割、DAO層で１つ、データベース層で１つ、横断的関心事（トランザクション、ログ管理など）で１つ。

今後について

- 今後、何か期待するものはありますか？
- 今後、取り上げて欲しい課題はありますか？

問題　その1

- 貴方は基盤グループのリーダです。定義ファイルはどうやって管理しますか？
  - ログと例外処理をアスペクトにしたいです。
  - 開発中はメソッドの開始と終了をログに出したいです。
  - 例外処理をUnitテストに含めるかどうかはどうしましょう？
- 適当な想定のもと、ズバリ管理方法を決めて下さい。

問題　その2

- 貴方はアーキテクトです。
- 例外とログとトランザクションという定番をアスペクトにしようとしたらお客さんに言われました。
- 「その程度かよ、もっとアスペクトで凄いことしてくれよ」
- なんか凄いことを提案して下さい。

問題　その3

- 貴方はコンサルタントです。
- お客さんはWebアプリ構築で悩んでいます。
  - Seasar2にするかSpringにするか？
  - StrutsにするかJSFにするか、それとも別の何かか？
  - HibernateかiBATIS、それともS2DaoかSpringJDBCか？
- 適当な想定のもと、お薦めの組合わせをズバリ言ってください