全部まとめてHOT deploy

～ドン引き？～

自己紹介

- 名前:木村　聡(きむら　さとし)
- Seasarプロジェクトコミッタ:
  - S2Struts
  - S2Mai
  - 舞姫
- 仕事
  - (株)フルネス
  - フレームワーク
  - 自動生成ツール

これまで書いたものとか

- 書籍:
  - Eclipseで学ぶはじめてのJava
  - Seasar入門　～はじめてのDI&AOP～
- 雑誌、Web記事
  - CodeZine
  - DB Magazine
  - WEB+DB
    - 「Seasar2徹底攻略」(Vol.31)
  - JavaWorld
    - 「開発者にとって“易しく、優しい”軽量コンテナSeasar2の実力を探る」(2005/05)

アジェンダ

- HOT deployとは
- SeasarのHOT deployの仕組み
- HOT deployのウソ・ホント
- もう悩まない

HOT deployとは

- Slim3のサイトから引用

HOT deployment

HOT deployment means re-deployment an application without restarting web application. Due to HOT deployment, you can see the result as soon as you change the source code. The quick feedback is essence of agile development.

  - Webアプリケーションの再起動なし
  - アプリケーションの再デプロイをする
  - ソースコードの変更をすぐに確認できる

プロダクト例

- Seasar2 (2.4系)
- JavaRebel
- その他
  - Ymir
  - Seam
  - Tapestry
  - など

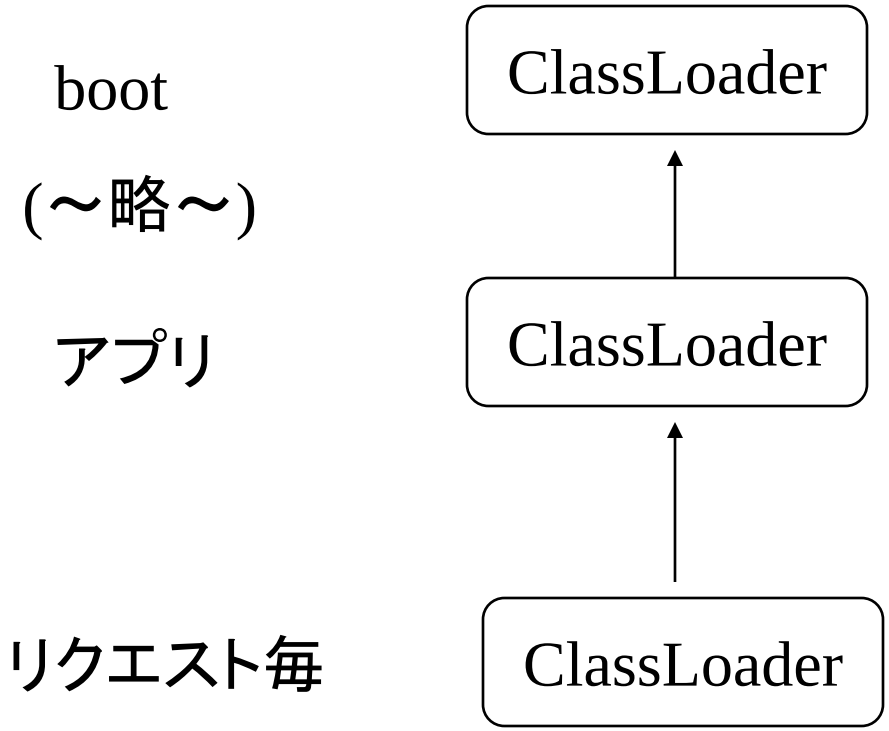
特徴

- Seasar
  - リクエストごとに毎回クラスローダーを作る
  - HOT deployフレームワーク
  - OSS
- JavaRebel
  - 変更を察知して、バイトコード修正
  - プラグイン方式
  - 有償

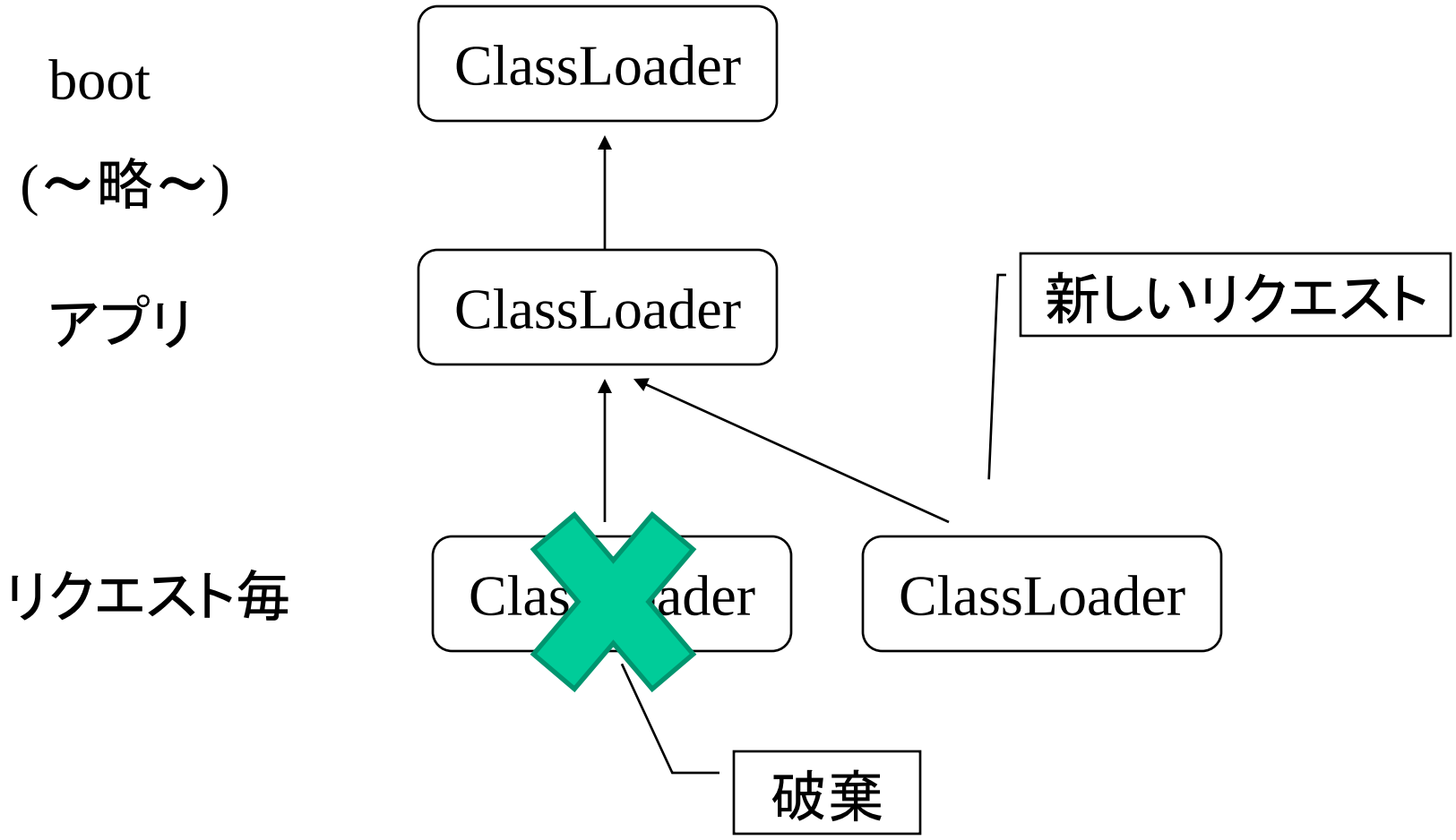
アジェンダ

- HOT deployとは
- SeasarのHOT deployの仕組み
- HOT deployのウソ・ホント
- もう悩まない

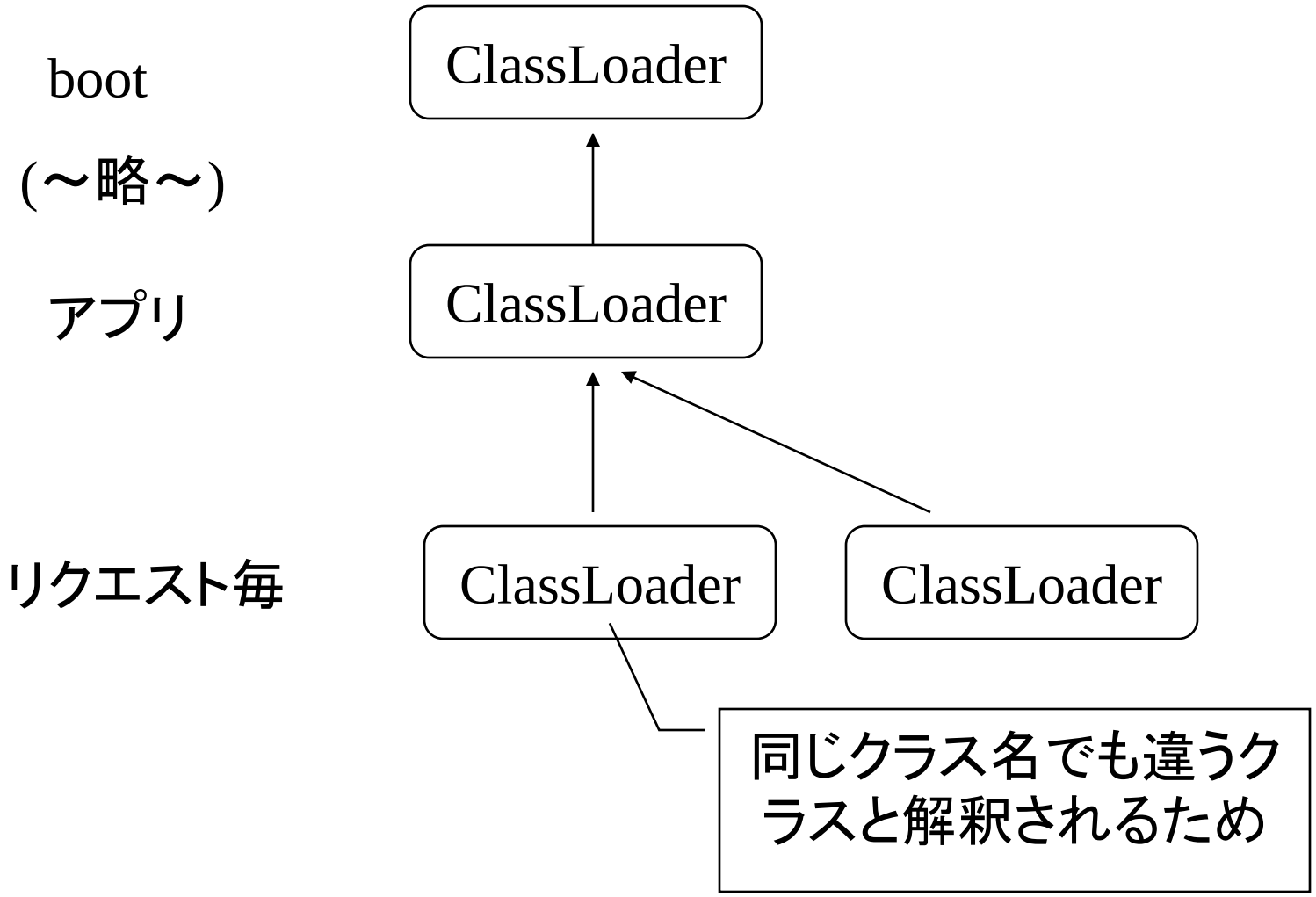
SeasarのHOT deployの仕組み



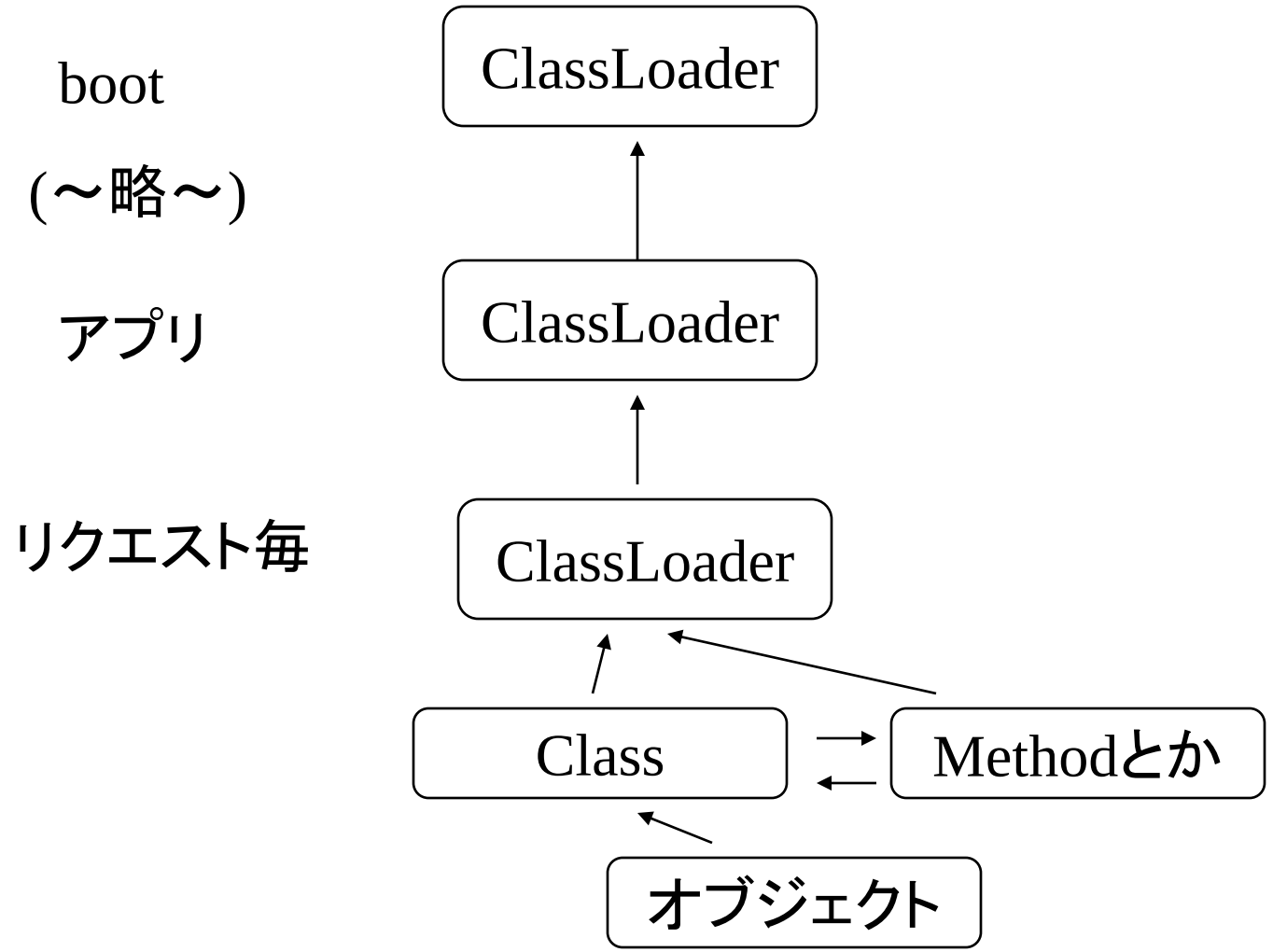
SeasarのHOT deployの仕組み



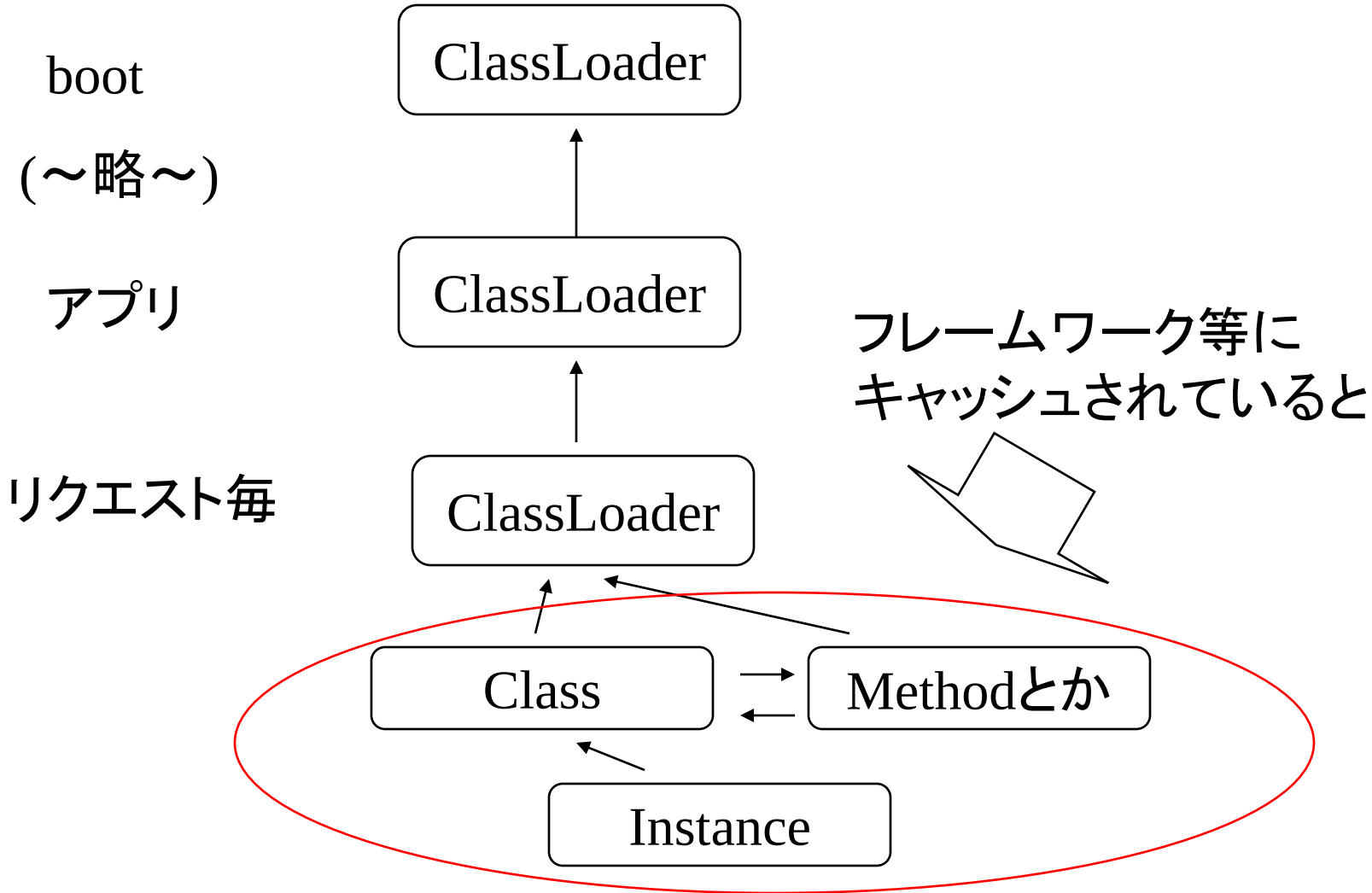
ClassCastException



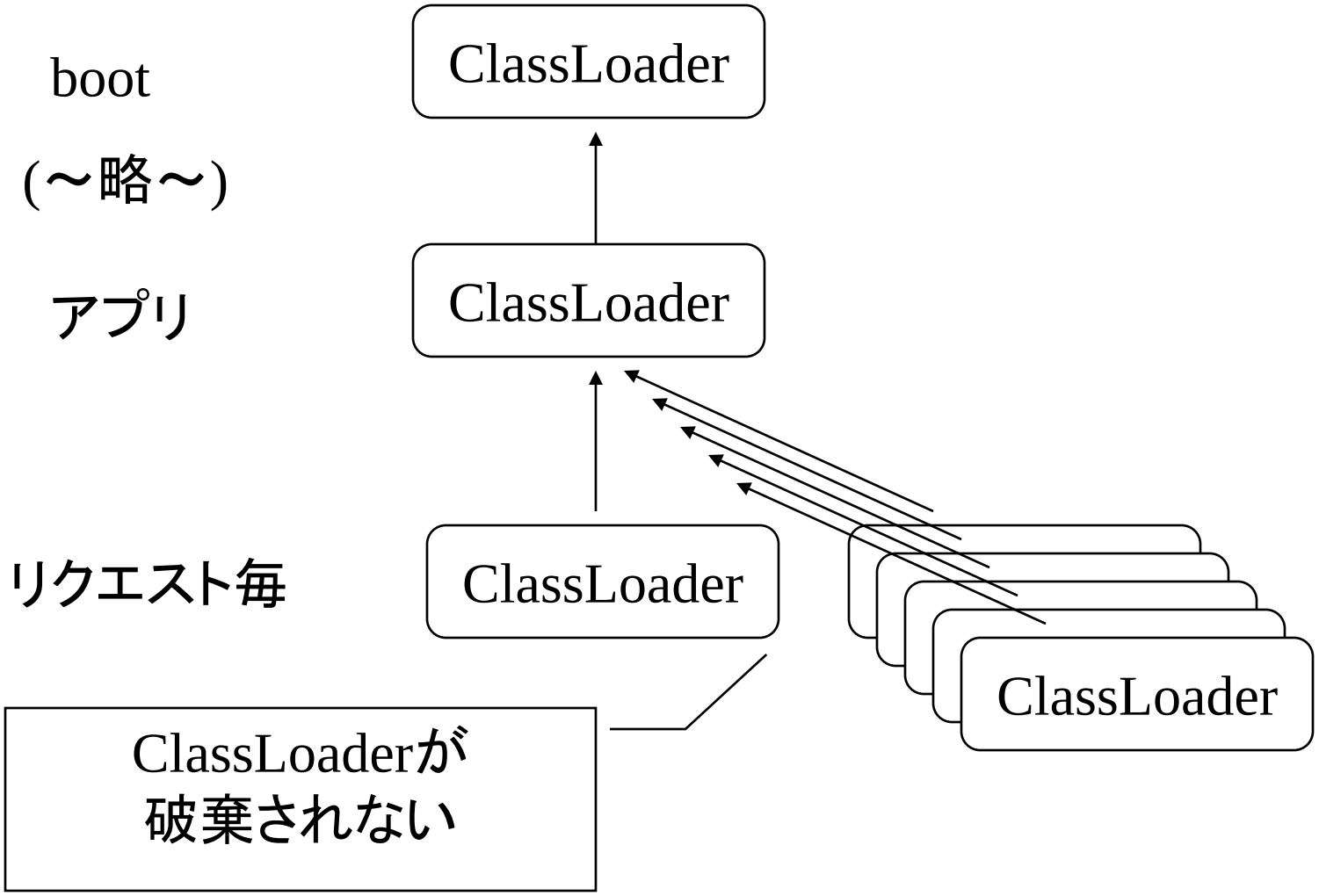
SeasarのHOT deployの仕組み



SeasarのHOT deployの仕組み



OutOfMemoryError



アジェンダ

- HOT deployとは
- SeasarのHOT deployの仕組み
- HOT deployのウソ・ホント
- もう悩まない

ウソ・ホント

- WebLogicでHOT deploy
  ～ひがやすを blog～
  - http://d.hatena.ne.jp/higayasuo/20081013/1223888469

JavaRebelでもそうですが、VMレベルで変更を認識することが可能でも、結局フレームワークがHOT deployに対応していないことには、どうしようもありません。JavaRebelのSpring対応では、クラスファイルの変更を認識すると結局、**Springの再起動**をする必要があり、**そこに時間がかかってしまう**ので、**実質的は使えない**のです。

ウソ・ホント

- WebLogicでHOT deploy
  ～ひがやすを blog～
  - http://d.hatena.ne.jp/higayasuo/20081013/1223888469

JavaRebelでもそうですが、VMレベルで変更を認識することが可能でも、結局フレームワークがHOT deployに対応していないことには、どうしようもありません。JavaRebelのSpring対応では、クラスファイルの変更を認識すると結局、**Springの再起動**をする必要があり、**そこに時間がかかってしまう**ので、**実質的は使えない**のです。

- そうでもない
- Springの起動は速い（SeasarのようにAOPのWeaving等しないので）
- 再デプロイするよりは速いし、「使えない」は言いすぎ

ウソ・ホント

- Seasar2とSpringの将来性
  ～ひがやすを blog～
  - http://d.hatena.ne.jp/higayasuo/20090201/1233464318

JavarebelのHOT deployの実装を見ればわかると思いますが、リクエストのたんびに全コンポーネントをデプロイして破棄していると思います。これでは、クラス数が増えると使い物になりません。
試してみるとわかると思いますが、**component-scanは、重い処理**なので、リクエストのたびに呼び出すのは、あまり現実的ではありません。

ウソ・ホント

- Seasar2とSpringの将来性
  ～ひがやすを blog～
  - http://d.hatena.ne.jp/higayasuo/20090201/1233464318

JavarebelのHOT deployの実装を見ればわかると思いますが、リクエストのたんびに全コンポーネントをデプロイして破棄していると思います。これでは、クラス数が増えると使い物になりません。
試してみるとわかると思いますが、**component-scanは、重い処理**なので、リクエストのたびに呼び出すのは、あまり現実的ではありません。

- メソッド呼び出し毎にクラスの変更をチェックしてるっぽい
- メソッドの呼び出しが多い処理だと厳しいかも？
  - 100000000回のコールで、4.2秒くらい増えた

ウソ・ホント

- それに対して、こんな意見もある
  - http://d.hatena.ne.jp/alunko/20090102/1230891319

Seasar2のHotDeployも同じだ。
毎リクエストで行われるコンポーネントの再登録に時間がかかる。
Web層（Action）が依存しているコンポーネントが子、孫まで合わせて全部で100くらいあると、
1回の画面遷移で行われる再登録処理に**30秒**くらいかかってしまう。

ウソ・ホント

- それに対して、こんな意見もある
  - http://d.hatena.ne.jp/alunko/20090102/1230891319

> Seasar2のHotDeployも同じだ。
> 毎リクエストで行われるコンポーネントの再登録に時間がかかる。
> Web層（Action）が依存しているコンポーネントが子、孫まで合わせて全部で100くらいあると、
> **1回の画面遷移で**行われる再登録処理に**30秒**くらいかかってしまう。

- ホント？
  - 会場の人に聞いてみたい

    →20秒頂きました

ウソ・ホント

- HOT deployでハマった実例あれこれ
  ～新・たけぞう瀕死の日記～
  - http://www3.vis.ne.jp/~asaki/p_diary/diary.cgi?Date=20090214#2009021401

ClassCastExceptionが発生することがある
S2JDBCでジョインしたエンティティが不正というエラーが出ることがある
などなど

ウソ・ホント

- HOT deployでハマった実例あれこれ
  ～新・たけぞう瀕死の日記～
  - http://www3.vis.ne.jp/~asaki/p_diary/diary.cgi?Date=20090214#2009021401

ClassCastExceptionが発生することがある
S2JDBCでジョインしたエンティティが不正というエラーが出ることがある
などなど

- あると思います
- 説明した通りの事が起こっている
- レールはあるが脱線しやすい

ウソ・ホント

- SeasarのHOT deployは難しい
  ～新・たけぞう瀕死の日記～
  - http://www3.vis.ne.jp/~asaki/p_diary/diary.cgi?Date=20090212#2009021200

フレームワークの拡張を行う場合も、きちんとHOT deploy対応をするにはそれなりの**ノウハウが必要**になります。JavaによるWeb開発で相当なスキルと経験を持っていて、なおかつSeasarの HOT deployの動作原理を把握していても、それでもHOT deployに**きちんと対応するフレームワーク拡張を行なうのは難しい**場合がある、というのが個人的な感覚です。また、ライブラリの中にはパフォーマンスのために自前でリフレクション情報などを**キャッシュしている**ものもあり、こういった**ライブラリとの相性も悪いです**。

ウソ・ホント

- SeasarのHOT deployは難しい
  ～新・たけぞう瀕死の日記～
  - http://www3.vis.ne.jp/~asaki/p_diary/diary.cgi?Date=20090212#2009021200

フレームワークの拡張を行う場合も、きちんとHOT deploy対応をするにはそれなりの**ノウハウが必要**になります。JavaによるWeb開発で相当なスキルと経験を持っていて、なおかつSeasarの HOT deployの動作原理を把握していても、それでもHOT deployに**きちんと対応するフレームワーク拡張を行なうのは難しい**場合がある、というのが個人的な感覚です。また、ライブラリの中にはパフォーマンスのために自前でリフレクション情報などを**キャッシュしている**ものもあり、こういった**ライブラリとの相性も悪いです**。

- 私も同意見
- 独自拡張が難しい。hot spotが狭い。
- 某Seasarコミッタ「S2XxxをHOT deploy対応したのに、きちんと動かない」

ウソ・ホント

- SeasarのHOT deployは難しい
  ～新・たけぞう瀕死の日記～
  - http://www3.vis.ne.jp/~asaki/p_diary/diary.cgi?Date=20090212#2009021200

SeasarのHOT deployはとても強力だけど、ちゃんとHOT deployの仕組みや挙動、サポート方法を熟知したアーキテクトがいないと**ハマる可能性も大きい**

ウソ・ホント

- SeasarのHOT deployは難しい
  ～新・たけぞう瀕死の日記～
  - http://www3.vis.ne.jp/~asaki/p_diary/diary.cgi?Date=20090212#2009021200

SeasarのHOT deployはとても強力だけど、ちゃんとHOT deployの仕組みや挙動、サポート方法を熟知したアーキテクトがいないと**ハマる可能性も大きい**

- そう思う

HOT deploy実装の難しさ

- クラスローダー型
  - キャッシュしてダメ
    - オブジェクトはもちろんClass、Method、Field
    - キャッシュをクリアする仕組みが必要
      - (SeasarはDisposable、DisposableUtil。詳しく知る必要あり)
  - セッションの中身をどうするか
    - シリアライズ、デシリアライズ
      - クラス構造が変わってなければOK
      - 使う側でも注意が必要
        - » (Seasarは直接Sessionに触れない)
- バイトコード型
  - バイトコードエンジニアリングの知識
    - Javassist

SeasarのHOT deployは難しい？

- 切り分けが難しい
  - フレームワークなどのバグ
    - 対応が不十分
  - 使い方を間違えている
    - HOT deploy対象外の利用
  - 漏れ
    - 仕組みを知っていても・・・

得意領域

- Seasar
  - パフォーマンス
- JavaRebel
  - トラブりにくい
    - プラグインの実装次第だけど、基本、再起動なので
  - Webアプリ以外にも使える

苦手領域

- Seasar
  - 設定ファイル
    - キャッシュするから
  - セッション管理など(ClassLoader周り)
    - 一般の開発者に
      ClassLoaderを意識させるのはどうなのか？
- JavaRebel
  - リソース管理
    - バイトコードエンジニアリングの対象外
  - クラス階層の変更
    - 親クラス、インターフェースを変更出来ない

組み合わせて使えないのか？

- できない
  - JavaRebelも独自のクラスローダを使っている
    - カスタムクラスローダーの利用はダメらしい
      - 検証はしてません・・・

特定のプロダクトは？

- Seasar
  - S2Xxxなど
    - 各プロダクト毎にHOT deploy対応
- JavaRebel
  - 各プロダクト(Struts,Springなど)
    - 各プロダクト毎にプラグインを作り対応
      - プラグインでリソースの読み直しなど

まとめ（その１）

- どちらも苦手領域・制約あり
- どっちもプロダクト毎の対応が必要

まとめ（その１）

- どちらも苦手領域・制約あり
- どっちもプロダクト毎の対応が必要

- プロダクト毎の対応ってどうなの？

解決するプロダクトあります

kimu-reloadableの概要

- クラスファイル、リソースファイル、設定ファイルが変更されたら変更を反映する
- セッション情報は引継ぎ
- 制約少ない
  - 規約は無し
    - Seasar:このパッケージ
  - 具象クラスから継承できる
  - 親クラス、インターフェースを変更できる

    など

手順

1. WEB-INF/classesとWEB-INF/libを移動
   - WEB-INF/dev/classes
   - WEB-INF/dev/lib
2. jarを配置
   - WEB-INF/lib/kimu-reloadable.jar

手順

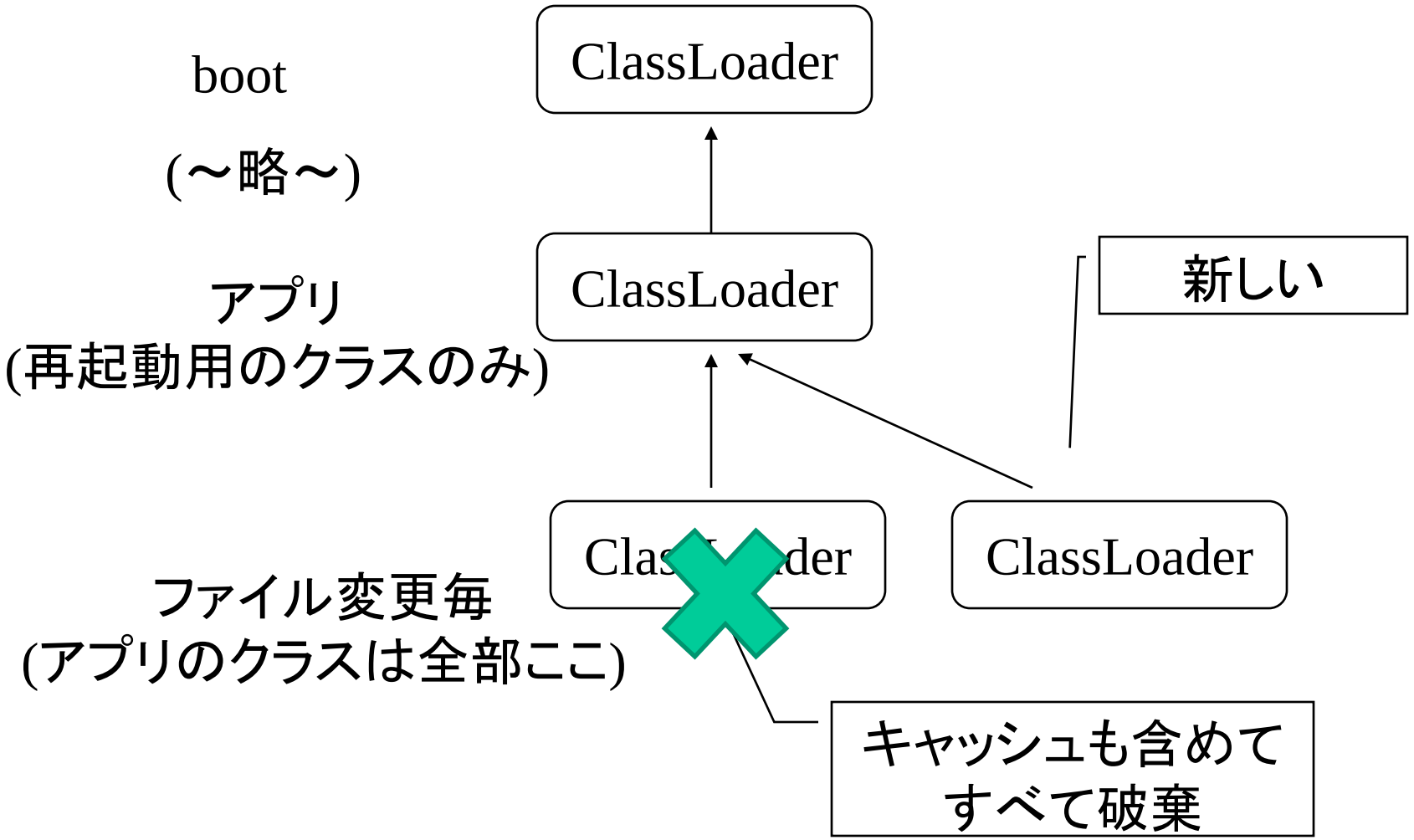
3. web.xmlファイルの名前を変える
   - web.xml→web.xml_org
4. ツール起動
   - java -jar kimu-reloadable-tool.jar web.xml_org > web.xml
5. リダイレクトで出力したweb.xmlを編集
   - ReloadableFilterを最初に持ってくる(コピペ)
   - JspServletを追加(コピペ)
   - (必要に応じてListenerフィルターの対応)

デモ

仕組み

- ファイルが変更されたら
  1. クラスローダを作り直し
  2. Servlet、Filterを再起動
  - リクエスト毎ではないので、変更が無ければ高速
- セッションオブジェクト
  - 自前でシリアライズ・デシリアライズ
    - クラス構造が変わっていてもOK
- 全部のクラスが同じクラスローダなので
  - ClassCastException出ない
  - キャッシュごと破棄
    - フレームワーク開発者の対応なし

仕組み


boot
(～略～)
アプリ
(再起動用のクラスのみ)
ファイル変更毎
(アプリのクラスは全部ここ)
ClassLoader
ClassLoader
ClassLoader
ClassLoader
新しい
キャッシュも含めて
すべて破棄

Seasar2やSlim3と組み合わせる

- セッション管理の仕組みを組み合わせることができる
  - ClassCastExceptionに悩まなくて済む
    (他の問題は解決しないけど)
- サーブレットなどを再起動しないので、より高速

デモ（その２）

デモ（その２）

1. Hogeクラスを定義
2. SessionにList<Hoge>を取り出し
   - 画面表示する
3. SessionにList<Hoge>をセット
   - ここまでで画面表示する
4. Hogeクラスにフィールドを追加する
   - フィールドの内容を表示する
   - 画面表示する

→ClassCastExceptionが出る

デモ（その２）

- web.xmlに次の設定をしただけ

```
<filter>
  <filter-name>SessionFilter</filter-name>
  <filter-class>jp.dododododo.reloadable.filter.SessionFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>SessionFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

- ClassCastExceptionが出ない

プロダクト情報

- 名前
  - kimu-reloadable
- URL
  - http://code.google.com/p/kimu-reloadable/

まとめ

- もう悩まない
- 多少遅いかも

最後に

- Seasarの昔のコンセプト
  - 「J2EEを易しく・優しく」

終わり

- ご清聴ありがとうございました