～ JBoss Seam と Embedded EJB3 で見る～ 次世代の Java EE アプリケーション開発

株式会社カサレアル
プロフェッショナルサービスセンター
阿島哲夫
岡本充洋

Agenda

- JBoss Seam とは
- JBoss Seam のコンテキスト管理
- JBoss Seam と JSF
- JBoss Seam のその他の機能
- JBoss Embeded EJB3
- まとめ

JBoss Seam とは

JavaEE コンポーネント - JSF

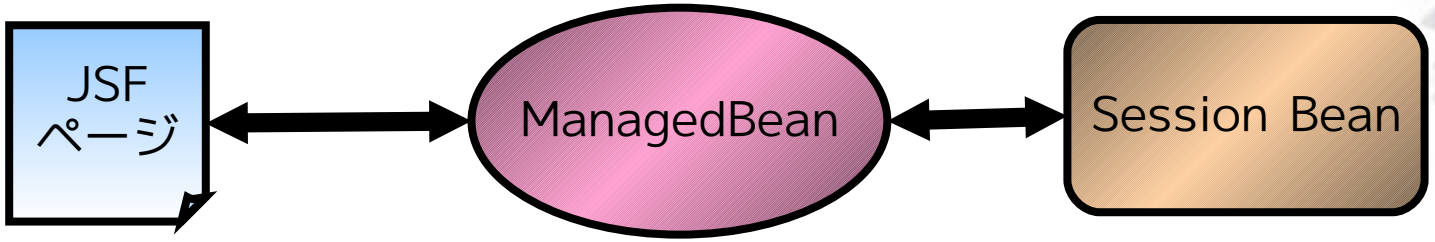
- Java EE 標準の Web アプリケーション フレームワーク
- 高機能なアプリケーションを簡単に作成
  - UIComponent アーキテクチャ
  - POJO ベースのコンポーネント開発
  - イベントドリブンなアプリケーションの制御
  - ステート (Session) 管理
- 拡張可能で柔軟なアーキテクチャ
  - Facelets
  - Ajax4jsf
  - etc...

JavaEE コンポーネント - EJB3

- EJB(Enterprise JavaBeans)3.0
  - 分散オブジェクト仕様
  - POJO+ アノテーションによる開発
    - アノテーションによる DI
    - JPA (Java Persistence API) による O/R マッピング
- エンタープライズシステムに必要な様々な機能を持つ
  - 分散トランザクション・トランザクション制御
  - リモート呼び出し
  - セキュリティ
  - ステート管理

JSF と EJB を連携させた場合の問題

- Managed Bean と Stateful Session Bean
  - HTTPSession によるセッション管理と Stateful Session Bean によるステートの２重管理
- View(JSF ページ) からは常に ManagedBean 経由で EJB にアクセスする必要がある
  - 開発者は ManagedBean を作成し、設定ファイルを書かなければならない



JBoss Seam とは

- JavaEE アプリケーションの Web 層と EJB 層を「縫い合わせる」フレームワーク
  - 「 Seam 」は「縫い目」という意味
  - JSF と EJB3 のコンポーネント、ステート管理を統合する
- OSS （ LGPL ライセンス）
  - 自由に利用可能
- JBoss(Red Hat) の Gavin King 氏が開発
  - Hibernate の作者でもある

JBoss Seam の特徴

- Seam コンテキストによるステート管理
  - EJB 層で独自のステート管理を行う
  - ステートの管理に HTTPSession を使わない
    - HTTPSession に縛られない
    - 豊富なスコープ
- アノテーションによるアプリケーションの設定
- DI+α を利用したコンテキスト操作
  - Injection ・ Outjection ・ Bijection

JBoss Seam で View と Model をつなぐ

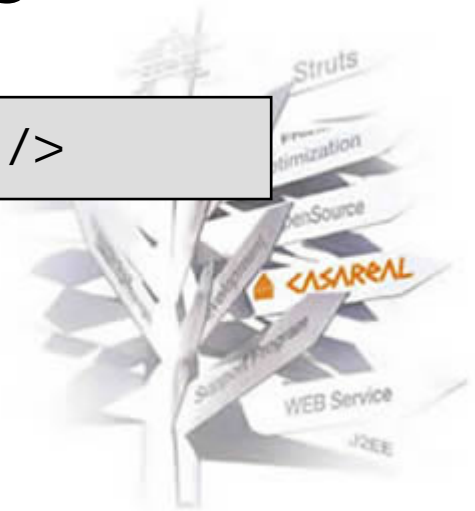
- JBossSeam が管理する EJB コンポーネントに @Name で一意な名前を付ける

```
@Entity
@Name("user")
@Scope(SESSION)
public class User implements Serializable{
```

- JSF ページからはこの名前で ManagedBean と同様にアクセスできる

```
<h:inputText id="username" value="#{user.username}" />
```

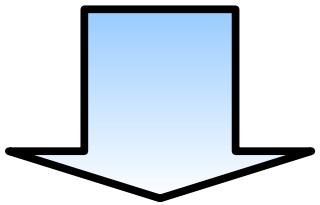
- ManagedBean 不要！



JBoss Seam のコンテキスト管理

HTTP Sessionの限界

- Sessionスコープの問題点
  - 複数ウィンドウの同時立ち上げ
  - ウィザードなどの複数リクエストにまたがるスコープ
  - Sessionコンテキストへのバインド・アンバインドをアプリケーションで管理
  - 戻るボタン対策

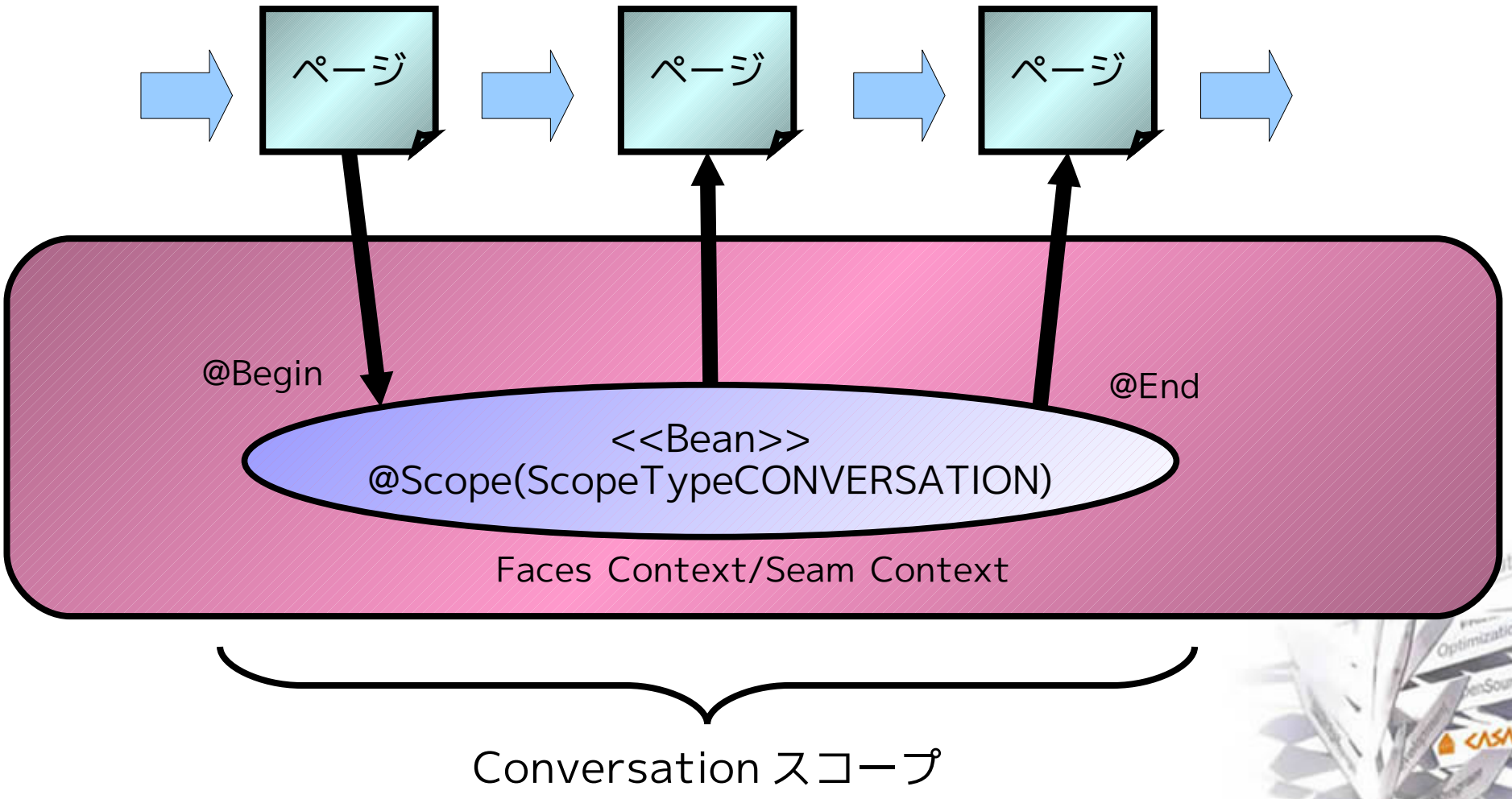


Conversationスコープで解決

Conversation スコープ

- Seam コンテキストで管理されるスコープの一つ
  - Stateful Session Bean を利用
  - 複数の画面遷移（ページフロー）に対応
  - 二度押し、戻るボタン、複数ウィンドウにも対応
- スコープの開始・終了はアノテーションで設定
- コンテキストへのバインド・アンバインドをアノテーションで設定できる
- 入力ウィザード等に適用可

Conversation スコープ

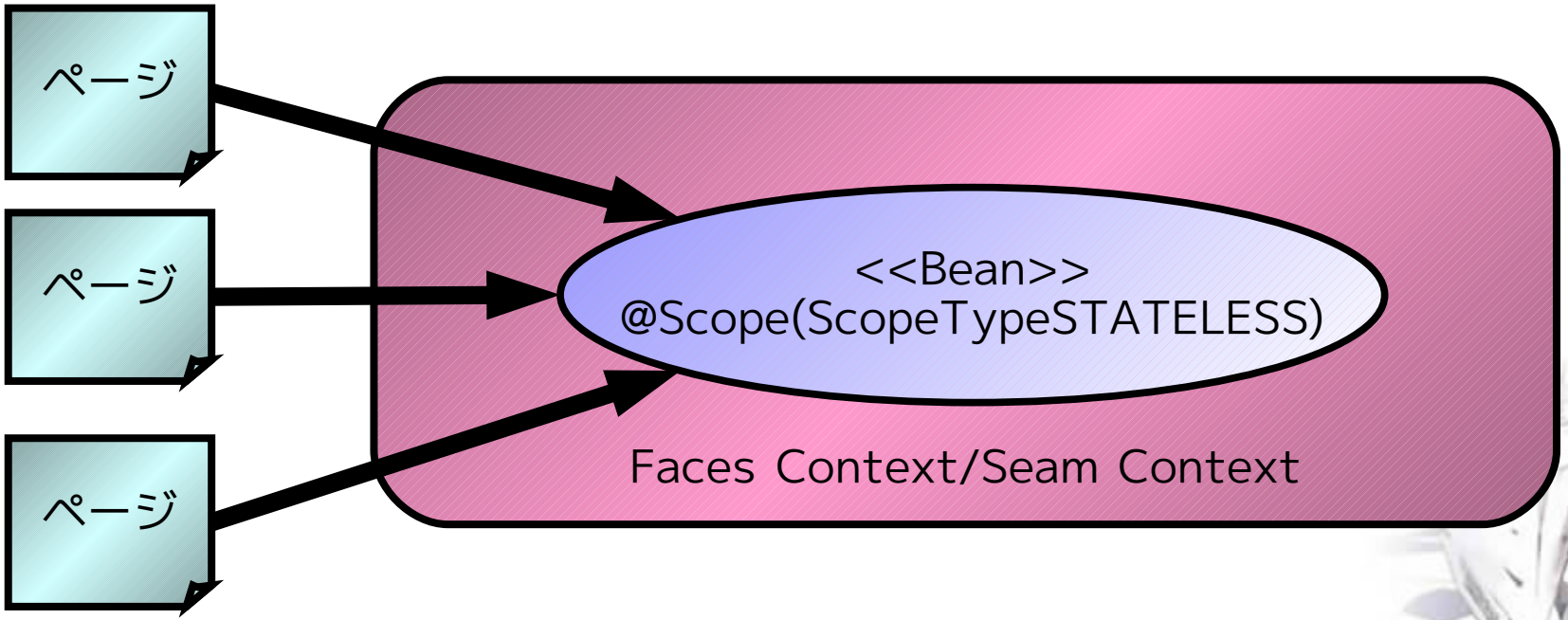


その他の JBoss Seam のスコープ

- Stateless - 状態を持たない
- Event - Servlet の request スコープ + α
- Page - JSP の page スコープと同等
- Conversation - 対話状態を保持
- Session - Servlet の session スコープと同等
- BusinessProcess - ビジネスプロセスが有効な間
- Application - Servlet の application スコープと同等

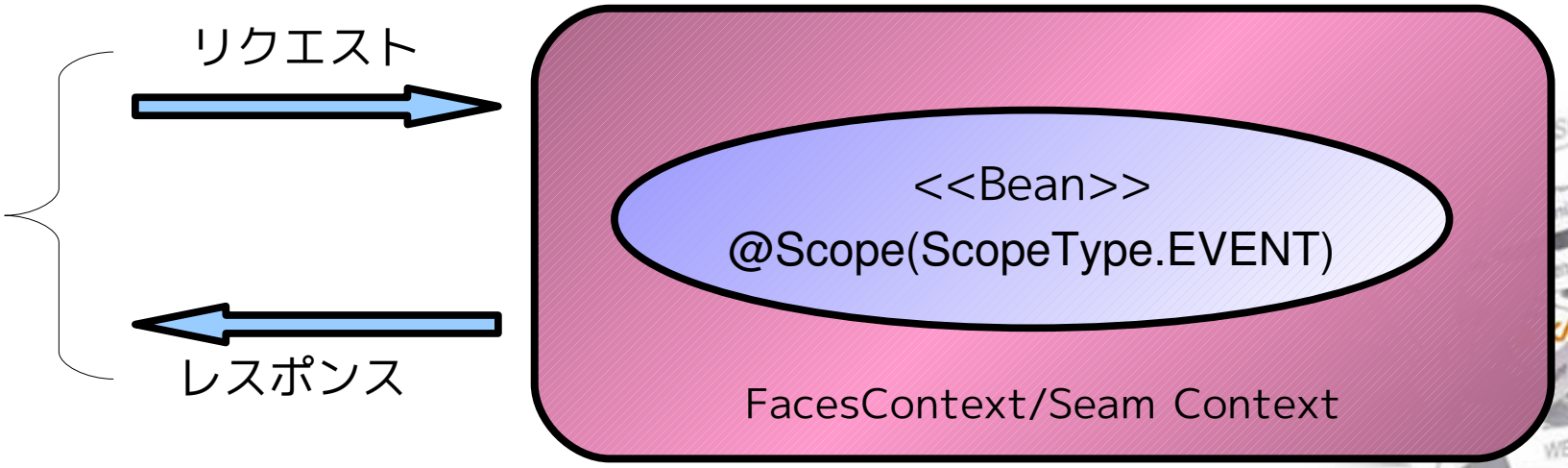
Stateless スコープ

- 状態を全く保持しないスコープ
- EJB の Stateless Session Bean と同等



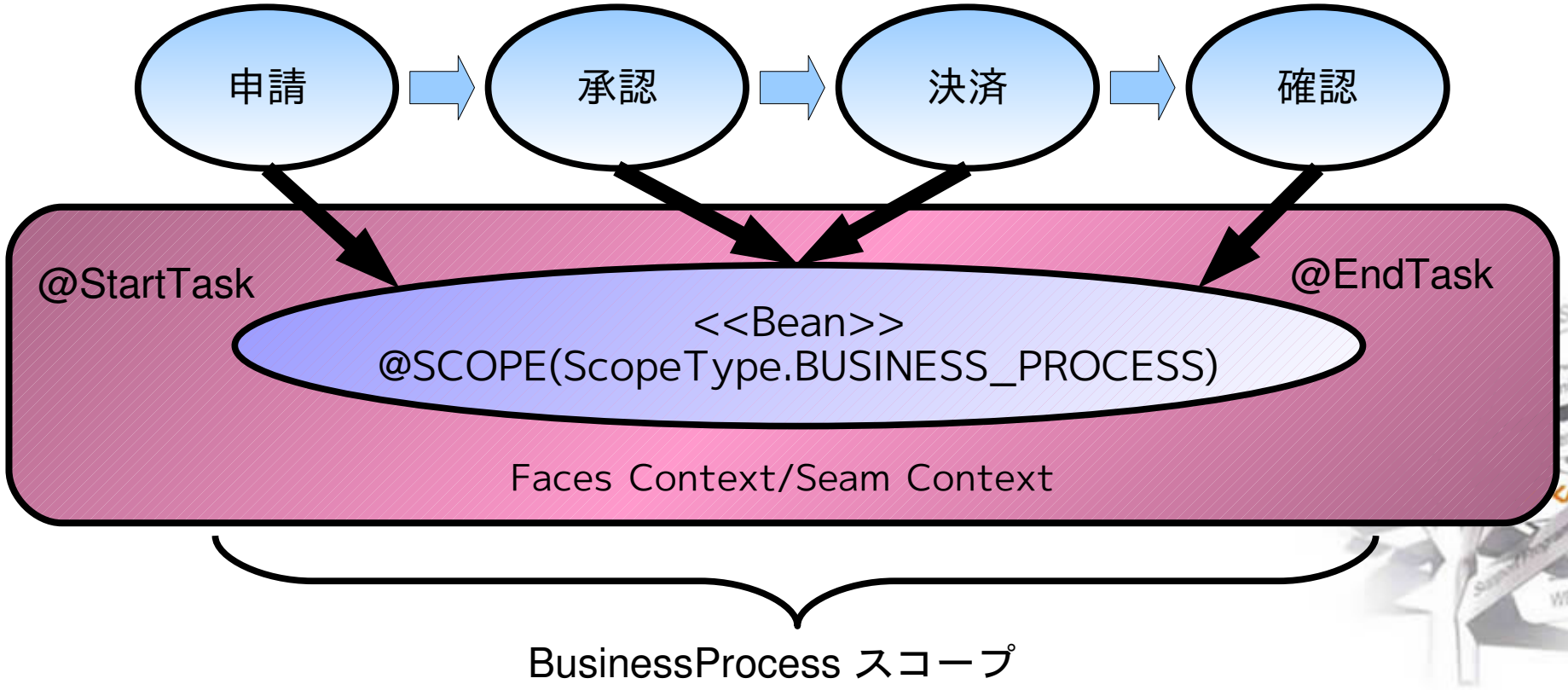
Event スコープ

- JSF のライフサイクルイベントの間（一回の画面遷移の間）のみ有効なスコープ
- 画面遷移にリダイレクトが使われても OK
  - Servlet API の request スコープとここが違う
- SeamRedirectFilter が必要



BusinessProcess スコープ

- ビジネスプロセス実行中は有効なスコープ
- JBoss jBPM 等と連携
- 承認ワークフロー等に適用



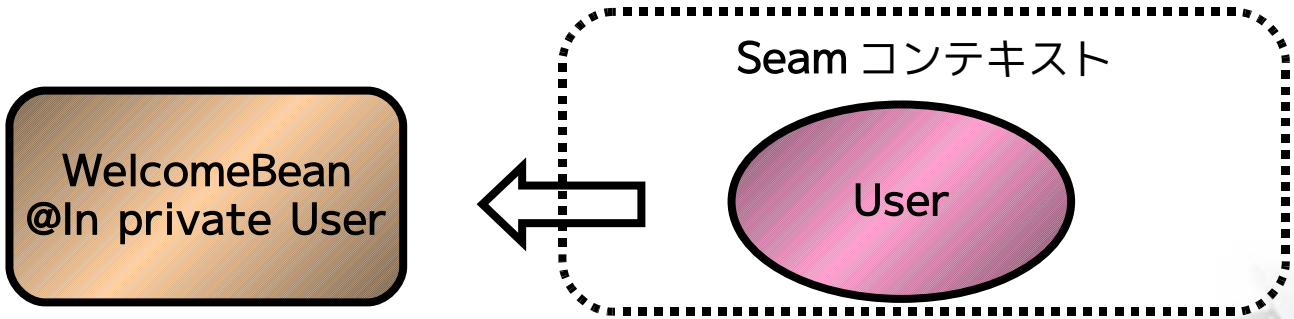
Injection ・ Outjection ・ Bijection

- アノテーションによる DI
- DI(Dependency Injection) とは
  - 依存（Dependency）オブジェクトをコンポーネント外部から設定（Injection）する仕組
- JBoss Seam では各スコープにごとにオブジェクトを管理する

@In アノテーション（Injection）

- フレームワークからオブジェクトを取得し、フィールドにセットする

```
@Stateful
@Name("bookingList")
public class BookingListAction implements BookingList, Serializable {
    @In
    private User user;
     . . . .
```

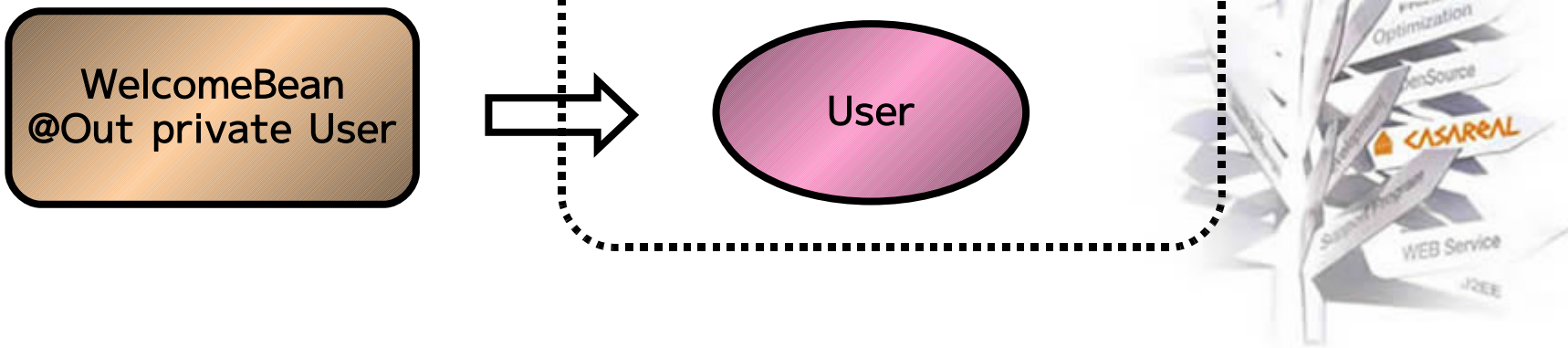


- Seam コンテキストが管理するオブジェクトが取得され、フィールドにセットされる

@Out アノテーション（Outjection）

- オブジェクトを Seam コンテキストに登録する

```
@Stateful
@Scope(EVENT)
@Name("changePassword")
public class ChangePasswordAction implements ChangePassword{
   @In @Out
   private User user;
    . . . .
```



Injection + Outjection = Bijection

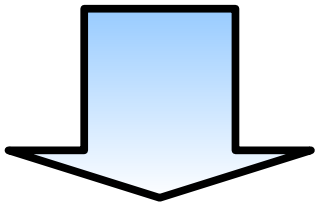
- @In と @Out を併記する
- メソッド実行前に @In でオブジェクトを取得
- 多くの DI コンテナとは違い、
  毎回のメソッド実行時に Inject が行われる
- メソッド実行後に @Out でオブジェクトを
  任意のスコープへセット

```
public class ChangePasswordAction implements ChangePassword{
   @In @Out
   private User user;
    . . . .
```

JBoss Seam と JSF

JSF の問題点

- JSF の UI コンポーネントアーキテクチャ自体は良い
  - POJO ベース、イベントドリブン、 EoD
- だが容認し難い問題点がある
  - 第 1 位　URL が遅れる (Postback)
  - 第 2 位　GET Method のパラメータは取得不可
  - 第 3 位 戻るページ対策、二度押し対策の不備



JBoss Seam の機能で解決

戻るボタン、二度押し対策

- JSF では、戻るボタン対策や、二度押し対策のための仕組みが無い
  - TransactionToken 等を自作する必要がある
  - JSF が流行らない原因第３位

- JBoss Seam では・・・
  - Conversasion スコープで自動的に対策される
    - conversation ID と言う一意な ID でステートを管理しているため
    - TransactionToken の役割も果たす

JSF のリクエスト Method 問題

- JSF では、 GET Method のリクエストパラメータを取得出来ない
- GET が使えないと・・・
  - 外部サイトからのパラメータ付きリンク不可
    - バナー広告など
  - フォームの入力値によって動的に結果が変わるページのブックマークが不可
    - 検索結果ページなど
- JSF が流行らない原因第２位

@RequestParameter

- JBoss Seam では・・・
  - @RequestParameter アノテーションで リクエストパラメータの取得が可能
  - Page アクションや @unwrap などの技術と組み合わせれば、 GET リクエストで任意の処理が可能
- Page アクション
  - ページへのリクエスト時に動作するアクション
    - JSF のレンダリングより前に動作
- @unwrap アノテーション
  - インジェクトしたいインスタンスを Getter を利用して生成する

JSF の Postback 問題

- JSF では Postback 呼ばれる技術を利用
  - 描画元の URL に対してデータを送信

↓

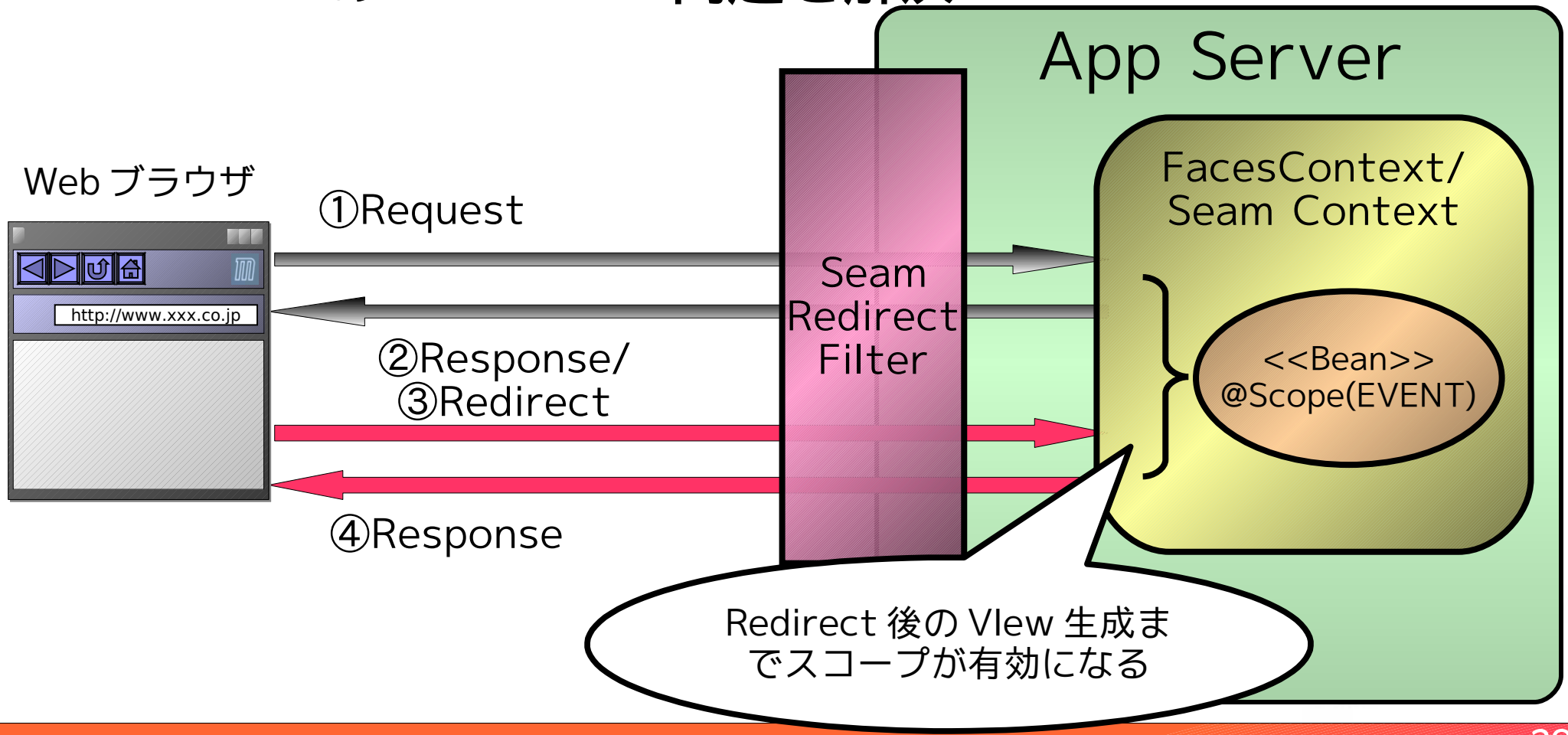
- (体感的に)*URL が 1 テンポ遅れる*
  - ブックマークの問題
  - そもそも気味が悪い
  - 大問題
  - JSF が流行らない原因第 1 位

PostBack 問題の解決方法

- リダイレクトすれば正しい URL の表示になる
  - いわゆる Post-Then-Redirect パターンの導入
- 普通に Post-Then-Redirect をすると Request スコープのデータは表示できない (Redirect 時のレスポンスで破棄される)
- JBoss Seam では・・・
  - リダイレクト後まで Request(Eevnt) スコープの値を保持する仕組みを提供
    - SeamRedirectFilter

SeamRidirectFilter

- Event スコープが、リダイレクトの後まで有効になる
  - JSF の PostBack 問題を解決



JBoss Seam のその他の機能

Seam Remoting

- JBoss Seam コンポーネントをリモートのクライアントから直接呼び出せる
  - EJB を JavaScript で呼び出せる
  - 要するに Ajax
- JavaScript から EJB を呼び出せる
- JMS メッセージを JavaScript でサブスクライブも可能
- ポーリングのサポート

JBoss Rules や jBPM との連携

- JBoss Rules
  - ルールエンジン
  - デシジョンテーブルから定義読み込みが可能
- jBPM
  - BPM エンジン
  - jPDL というプロセス記述言語で業務プロセスを記述
- JBoss Seam では、ルールエンジンや BPM エンジンとも連携可能

更に JBoss Seam の持つ機能

- 国際化サポート
- ロギング
- インターセプタ
- イベント
- メッセージング (JMS - MDB)
- ワークスペース管理

とにかく非常に多くの機能を持つ

Tomcat で EJB3
～ JBoss Embedded EJB3 ～

Embedded EJB3 とは

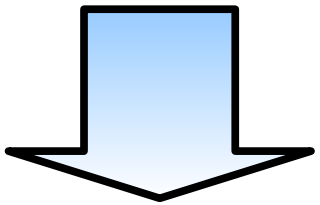
- 組込み EJB3 コンテナ
  - DI コンテナベースで EJB の機能を実現
- オープンソースライセンス（ LGPL ）
- EJB3,Local JNDI,Transaction,Security,JMS
  - JBossAS の機能が DI コンテナに凝縮
- Tomcat 上で EJB を動作させることも可能
- JBoss5 のコアとなる技術

EJB アプリケーション開発の問題点

- EJB コンテナが重い
- デプロイが面倒
- 製品依存の設定・実装が多い
  - アプリケーションのポータビリティ低下
  - OSS 開発ツールの対応が充実しにくい

EJB コンテナが重い

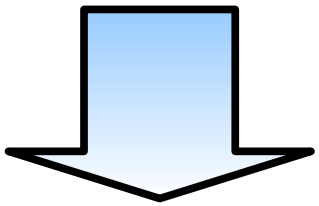
- EJB コンテナを含んだ、 JavaEE フル対応のアプリケーションサーバ
  - メモリや CPU 等のリソース消費が大きい



開発効率の低下

デプロイが面倒

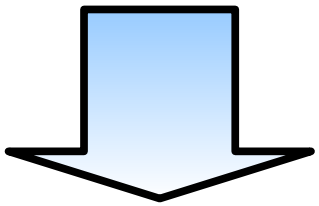
- EJB モジュール、 Web モジュール、EAR モジュールを作成
- 各モジュールの DD を記述
- 各モジュールを組み合わせてアーカイブ



開発効率の低下
開発の複雑化

製品依存の実装・設定が多い

- 多くの場合、標準仕様の DD の他に製品依存の DD が必要
- デプロイ手順は製品依存
- 場合によっては製品依存の API やアノテーションを利用しなくてはならない



アプリケーションのポータビリティ低下
OSS 開発ツールの対応が充実しにくい

Embedded EJB3 を使うと

- EJB コンテナレスで EJB アプリケーション
- Web アプリケーションにライブラリとして EJB コンテナを組み込む
  - Servlet コンテナのみで OK
  - WAR ファイルのみのシンプルデプロイ
  - アプリケーションが製品依存しない

Embedded EJB3 + JBoss Seam

- Tomcat で EJB & JBoss Seam
- Jetty で EJB & JBoss Seam

- Web サービスや Global JNDI、分散トランザクション、クラスタリングまでサポートする予定
  - JBossAS 自体が DI コンテナに

まとめ

- JBoss Seam の特徴
  - JavaEE ベースで EoD を実現
  - リッチなステート管理
  - JSF の欠点を克服
  - EJB を JMS を Ajax で呼び出し可能
  - JBoss Rules や jBpm との連携
  - 他にも様々な機能
- Embedded EJB3
  - 組込み EJB
  - 次世代 JBoss のコアとなる技術

JBoss Seam の今後

- JBoss Seam1.1 が開発中
  - Generator(Seam-gen) が実装される
    - Ruby on Rails を意識？
  - 例外ハンドリング機能
  - etc...

- JSR-299 WebBeans として Java 標準となる予定
  - 今から押さえておけば、 2008 年も安心

ご静聴ありがとうございました。