

JBoss Seam: framework nowej generacji

Wstęp

Piotr Kochański
p.kochanski@erudis.pl
www.erudis.pl
konsultant firmy **Erudis**

szkolenia

- java
- modelowanie UML
- zbieranie wymagań
- zarządzanie zmianą
- zarządzanie projektami
- PRINCE2

narzędzia IT

- UML
- środowiska IDE
- wersjonowanie
- śledzenie zmian i błędów
- testowanie
- wdrażanie

Agenda

- Wstęp
- Elementy JBoss Seam: JSF + EJB3 + ...
- ... brakujące ogniwo
- Możliwości JBoss Seam
- Przykład zastosowania
 - uproszczenie aplikacji
 - konteksty, konwersacje
- Generowanie aplikacji (EJB3 on Rails)
- Projektowanie procesów (biznesowych)

Co jest wyjątkowego w JBoss Seam

- Frameworków jest wiele, ale
 - nie są standardem (w sensie Java EE)
 - często ignorują część standardu Java EE
- JBoss Seam opiera się w możliwie dużym stopniu o standardowe technologie Java EE
- Rozwiązuje napotymane problemy:
 - tworzenia architektury aplikacji
 - związane z charakterystyką aplikacji WWW
 - drobniejsze, czysto praktyczne (np. linki do stron)
- Prawdopodobnie rozwiązanie zbliżone do JBoss Seam stanie się standardem: JSR 299 (WebBeans)

Java Server Faces

- Komponentowa technologia tworzenia dynamicznych stron WWW
- Co dostajemy?
 - szkielet aplikacji oparty o wzorzec Model-Widok-Kontroler (MVC). (JSF = Widok + Kontroler)
 - możliwość tworzenia kontrolek (podobnie jak dla GUI)
 - sterowanie aplikacją oparte jest o obsługę zdarzeń
 - konwersja typów i walidacja formularzy
 - Spójny sposób konfiguracji aplikacji

JSF - jak to działa?

imię:

nazwisko:

strona.jsp

```
public class ManagedBean{
    String imię;
    String nazwisko;

    public String akcja() {
        //...
        return "ok";
    }
}
```

ManagedBean.java

```
<navigation-rule>
  <navigation-case>
    <from-outcome>ok</from-outcome>
    <to-view-id>/welcome.jsp</to-view-id>
  </navigation-case>
</navigation-rule>

<managed-bean>...</managed-bean>
```

faces-config.xml

JSF - jak to działa, C.D.

- Gdy formularz jest wysłany, inicjalizowany jest obiekt komponentu JavaBean stowarzyszony z formularzem (pola formularz = pola klasy)
 - w pliku konfiguracyjnym deklarujemy komponenty JavaBean
- Wykonywana jest metoda, która jest zdefiniowana jako akcja obsługi formularza
- Działanie aplikacji jest przekierowane zgodnie z wynikiem zwróconym przez akcję
 - możliwe przekierowania definiujemy w pliku konfiguracyjnym

EJB 3 + JPA

- Komponenty EJB (Enterprise Java Beans)
 - zarządzane, serwerowe komponenty do budowy rozproszonych aplikacji, realizujące ich logikę biznesową
 - EJB dbają o transakcyjność, wielowątkowość, RMI, failover, bezpieczeństwo, klastrowanie...
- JPA (Java Persistence API)
 - most relacyjno-objektowy (podobny do TopLinka, Hibernate, Kodo, itp.)
 - JPA integruje się z EJB (transakcyjność)

JSF + EJB3 + JPA

- Typowy model współdziałania:
 - komponent JavaBean JSF (*managed bean*) wywołuje metody komponentu EJB i/lub wykonuje operacje bazodanowe przy pomocy JPA
- Pytania?
 - Po co nam JSF JavaBean, jeśli mamy komponent EJB?
 - Jak zrealizować optymalnie obsługę sesji klienta?
Sesja HTTP, sesyjny komponent EJB?
 - Jak zrealizować optymalnie poprawne zachowanie w przypadku nieatomowych modyfikacji bazy danych przez wielu klientów („długie” transakcje aplikacyjne)

JBoss Seam, co dostajemy

- Integracja JSF z EJB3. Poprawienie funkcjonalności JSF
- Rozwiązanie problemów „długich” transakcji aplikacyjnych; zarządzanie zasięgami obiektów.
- Ułatwienie tworzenia aplikacji wykorzystujących AJAX (*Asynchronous JavaScript and XML*)
- Możliwość wykorzystania POJO zamiast EJB
- Konfiguracja przez metadane (*annotations*)
- Generowanie szkieletu aplikacji (*scaffolding*)
- Wsparcie dla BPM (Business Process Management)
- Łatwiejsza konfiguracja nawigacji (*pageflow*)

JBoss Seam



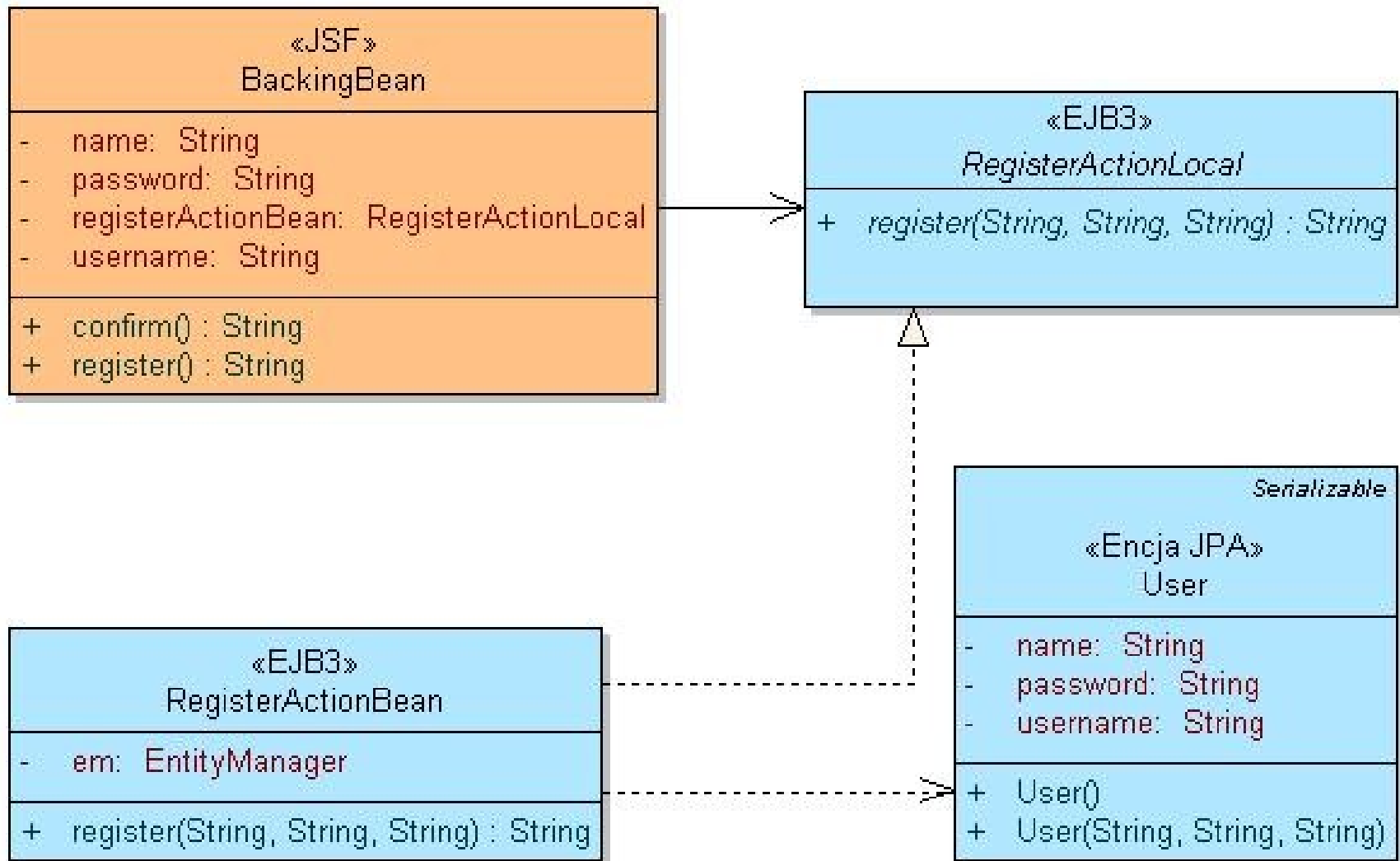
Enterprise Java Beans

Przykład

- Aplikacja JSF bez JBoss Seam
- Ta sama aplikacja, ale wykorzystująca JBoss Seam
 - problem transakcji aplikacyjnych
- Jeszcze raz ta sama aplikacja, ale wykorzystująca konwersacje Seam

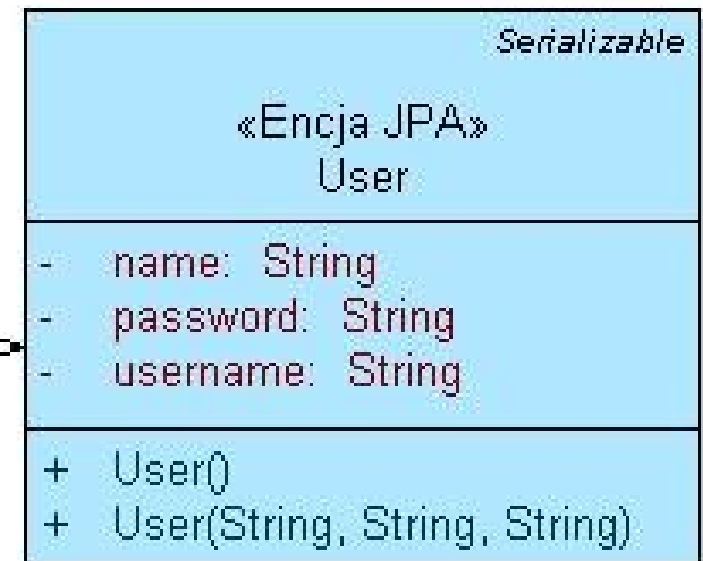
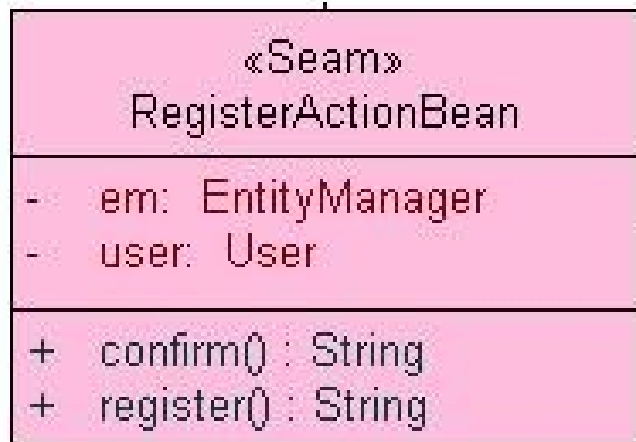
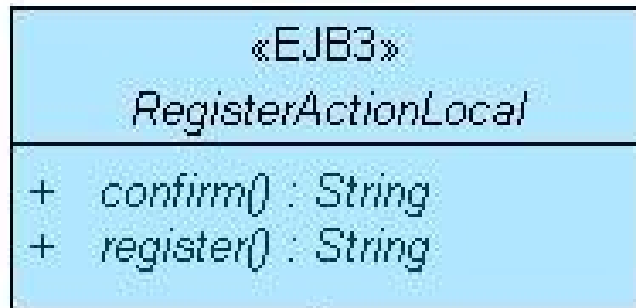
Aplikacja JSF + EJB bez Seam

class Rejestracja bez Seam



Aplikacja JSF + EJB3 wykorzystująca JBoss Seam

class Rejestracja z Seam



Aplikacja JSF + EJB3 wykorzystująca konwersacje Seam

```
@Name ("user")  
@Scope (ScopeType.CONVERSATION)  
public class User {  
    //...  
}
```

```
@Stateless  
@Name ("register")  
public class RegisterActionBean {  
    @In  
    User user;  
  
    @Begin  
    public String confirm() {...}  
  
    @End  
    public String register() {...}
```

Konteksty i konwersacje

- JBoss Seam umożliwia umieszczanie obiektów w różnego rodzaju *kontekstach* (zasięgach)
- standardowych: bezstanowym, strony, sesji, żądania, aplikacji
- niestandardowych: **konwersacyjnym** i kontekście procesu biznesowego
- Nas interesuje zasięg konwersacyjny

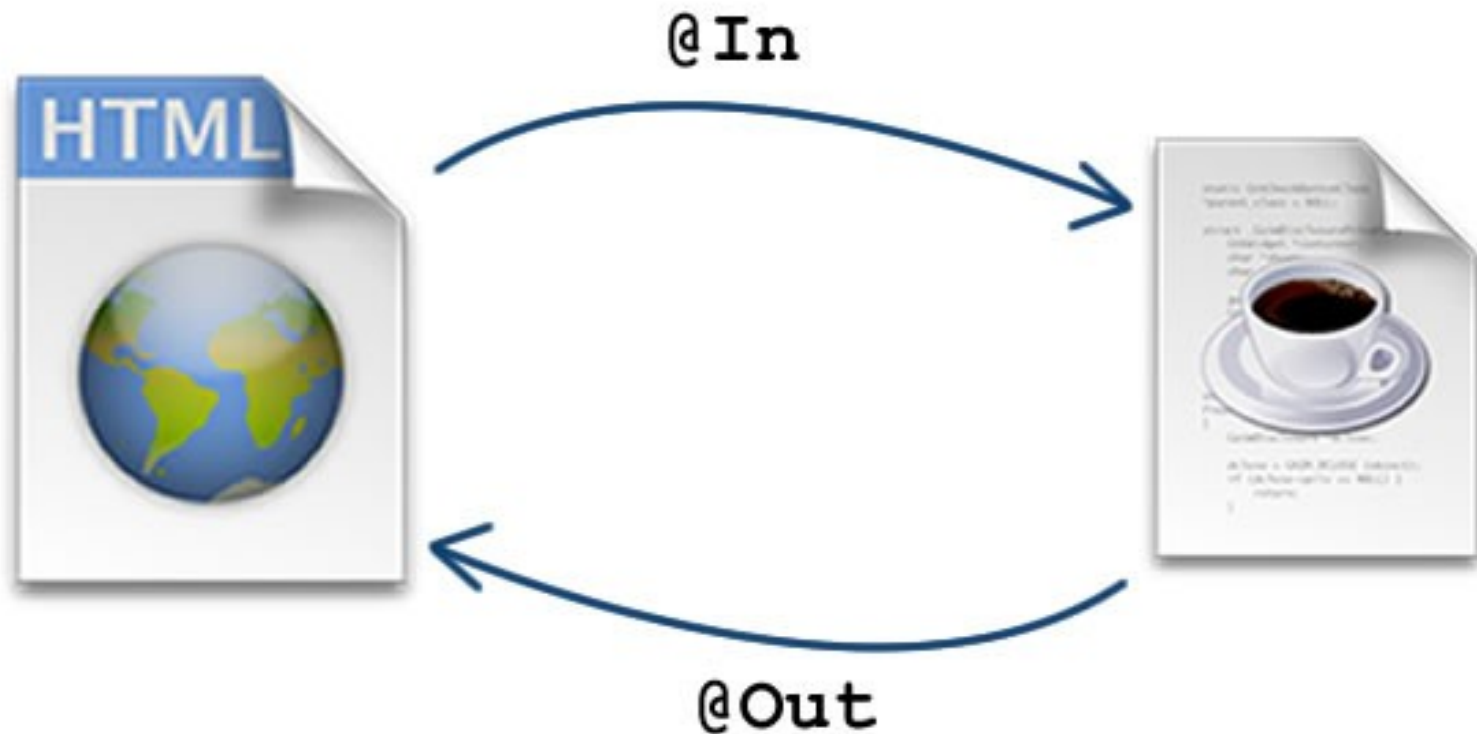


Konwersacje (atomowe)

- Konwersacja: sekwencja (wielu) żądań klienta realizująca jakieś zadanie
- Konwersacje „na surowo”, dwie typowe metody:
 - (a) po każdym żądaniu odtwarzamy stan aplikacji z bazy danych
 - (b) zapisujemy stan aplikacji w sesji
- (a) prowadzi często do braku skalowalności i problemów wydajnościowych
- rozwiązanie (b) jest trudne do oprogramowania
- JBoss Seam sam dba o to, żeby konwersacje (jednoczesne!) nie zakłócały się (atomowość)

Obiekty zarządzane przez Seam

- *Dependency bijection: injection + outjection*



Jak działa magia? Dwa elementy

- Sesyjne **stanowe** komponenty EJB
 - pogłoski o ich śmierci są mocno przesadzone
 - to nieprawda, że stanowe EJB są „niewydajne”, źle się klastrują
- JPA Extended Persistence Context
 - obiekty reprezentujące dane z bazy danych (encje JPA) są z nią złączone przez cały czas życia sesji...
 - ...**ale** w taki sposób, że nie musi być otwarta cały czas transakcja, a mimo to nie tracimy izolacji transakcji



Niezawodność aplikacji

- Każdy typ aplikacji ma swoje „miękkie punkty”
 - aplikacje GUI: np. poprawne zachowanie wielowątkowe, reakcja na zmianę wielkości okna
 - aplikacje WWW: zakłócenie przewidywanego toku działania, np.:
 - otwieranie nowych okien,
 - przycisk „Wstecz” przeglądarki,
 - ręczne wpisywanie adresu
 - ...
- JBoss Seam automatycznie pozwala uwzględnić wiele typowych problemów aplikacji WWW

JBoss Seam + AJAX

- JBoss Seam jest idealnym rozwiązaniem dla aplikacji AJAX, dlaczego?
 - AJAX = bardzo dużo drobnych żądań, co przekłada się na wiele zapytań do bazy danych
 - W praktyce stan aplikacji musi być przechowywany wydajniej, co robi za nas Seam: **cache**
 - W aplikacjach AJAX tym bardziej występują problemy transakcji aplikacyjnych i ich izolacji
- *Dygresja*: do bardziej złożonych aplikacji AJAX jest złym technicznie rozwiązaniem: warto porównać AJAX-owe edytory tekstu/arkusze kalkulacyjne z ThinkFree, który jest apletem Java

Więcej...

- Klastrowanie nie tylko na poziomie EJB, ale także zwykłych POJO
- Schowek (cache) JBoss Seam (obiektów i stron JSF)
- *Scaffolding*, czyli generowanie szkieletu aplikacji (jak Ruby on Rails) przy pomocy **seam-gen** (dla Eclipse i NetBeans)
- Walidacja może być zawarta w kodzie Java, a nie na stronie JSF
- Autoryzacja oparta o role i uprawnienia
- Tworzenie logu aplikacji (@Logger)
- Wsparcie dla skórek (np. wersja do druku, wersja dla niedowidzących)

JBoss Seam a reszta Świata?



JBoss Seam a reszta Świata, C.D.

- Zaleta JBoss Seam: wykorzystanie standardów Java EE i minimalna ingerencja w aplikację
- Ale są inne rozwiązania (szkielety aplikacji)
 - najpopularniejsze: Spring Framework
 - WebWork2/Struts, Tapestry, OpenLaszlo, ...
- Od wersji 1.2 Seam integruje się ze Spring-iem
 - możliwość użycia *Dependency Injection* w obie strony: komponent Seam > Spring JavaBean i na odwrót
 - Spring JavaBean może być umieszczany w kontekście JBoss Seam

Wyższy poziom abstrakcji

- Projektowanie nawigacji między stronami poprzez jPDL (pageflow definition)
 - przy złożonych przebiegach działania aplikacji jPDL jest łatwiejszy w „ogarnięciu” niż standardowa nawigacja JSF
- Tworzenie aplikacji poprzez definicję jBPM (Business Process Management)
 - definiujemy proces w pliku konfiguracyjnym
 - w kodzie umieszczamy przy pomocy metadanych uchwyt do elementów procesu

Uwagi

- Seam może współpracować z każdym serwerem aplikacji Java EE 5.0: JBoss 4.0.5 (prawie JEE 5), GlassFish
- Seam działa także na serwerze WWW Tomcat: zamiast komponentów EJB używamy zwykłych klas Java
- Seam może współpracować z dowolną biblioteką komponentów JSF: w szczególności z Facelets, ajax4JSF, RichFaces, ICEfaces
- Seam nadaje się nie tylko do aplikacji WWW, w ramach Seam możemy swobodnie używać wszystkich możliwości EJB

Źródła wiedzy

- Tutorial JBoss Seam
<http://labs.jboss.com/jbossseam/docs>
- Przykładowe *case study*:
http://blogs.nuxeo.com/sections/blogs/eric_barroca/2006_11_23_nuxeo-s-presentation-jboss-world-berlin-2006
- Blog Briana Leonarda (Seam + NetBeans + GlassFish)
http://weblogs.java.net/blog/bleonard/archive/2006/05/trying_out_jbos_2.html
- Artykuł na ONJava
<http://www.onjava.com/lpt/a/6512>
- Wstęp do JBoss Seam (fragment książki)
<http://www.infoq.com/articles/jboss-seam>

Podsumowanie

- JBoss Seam - nowa jakość
 - oparcie się o standardy
 - konwersacje, cache, klastrowanie
 - rozwiązanie wielu problemów jakie dostarcza tworzenie aplikacji WWW
- Prezentacja + przykładowe projekty:
<http://www.erudis.pl/sc/seam>
- Kontakt: p.kochanski@erudis.pl