

Poznań Java User Group 2005

# Java Server Faces

Wprowadzenie  
(Fakty i Mity)

# Plan prezentacji

- Ograniczenia szkieletów WWW
- Podstawowe elementy JSF
- JSF krok po kroku:
  - Warstwa wizualna
  - Obsługa zdarzeń
  - Nawigacja
- Podsumowanie

# Ograniczenia szkieletów WWW

# Wady szkieletów WWW

- Mnogość rozwiązań (kilkadziesiąt mniej lub bardziej popularnych rozwiązań)
- Brak zorientowania na komponenty
- Brak zorientowania na zdarzenia (bliski związek z HTTP)
- Słabe wsparcie ze strony narzędzi WYSWIG (wynikające z powyższych)

# Wady szkieletów WWW

- Poprzednie stwierdzenia nie są prawdziwe dla wszystkich rozwiązań (Barracuda, Web on Swing, W4toolkit)
- ALE żadne z popularnych rozwiązań poprzedniej generacji (Struts, Spring MVC, WebWork2) nie rozwiązało tych kwestii

# Obietnice JSF

- Prostota wytwarzania aplikacji porównywalna z ASP.NET
- Duże wsparcie ze strony narzędzi WYSWIG
- Wiele konkurencyjnych implementacji
- Rynek komponentów nie związanych z danym dostawcą
- Doświadczony twórca 😊 - Craig McClanahan

# Podstawowe elementy JSF

# A co dostaliśmy?



# Składniki JSF

JSF w dużej części składa się z wtyczek:

- ViewHandler – mechanizm odpowiedzialny za budowanie drzewa komponentów
- NavigationHandler – mechanizm „dobierający” wyświetlaną stronę

# Składniki JSF – cd.

- `VariableResolver` – Mechanizm wiążący komponenty z obiektami Java
- `PropertyResolver` – Mechanizm powiązany z `VariableResolver`
- `StateManager` – Mechanizm budujący i usuwający drzewo komponentów

# JSF krok po kroku

# Warstwa wizualna

- Istnieją ViewHandler dla XUL, HTML i dla klas Java (bardzo niedojrzałe)
- Specyfikacja wymaga obsługi JSP jako warstwy wizualnej
- Obsługa JSP ma na celu przyciągnięcie szerokiej rzeszy użytkowników Java Server Pages
- JSP zrealizowane jako dwie biblioteki znaczników – core i html

# Warstwa wizualna

- Kontynuowanie wad JSP
  - Skomplikowana, podatna na błędy, nielubiana ;)
- Nieporozumienia wynikające z filozofii JSF:
  - JSP służy jedynie do budowy drzewa komponentów – nie obsługuje danych nie-JSF w sposób przewidywalny przez początkujących
  - Nie współpracuje z JSTL (bo wg. mnie nie powinien 😊)
- Brak podstawowych komponentów  
(np. prostej listy )

# Warstwa wizualna - przykład

```
<% @ taglib uri="http://java.sun.com/jsf/core" prefix="f"%>
<% @ taglib uri="http://java.sun.com/jsf/html" prefix="h"%>
<f:view>
<h:form>
<h:inputText value="#{editor.book.title}" />
<h:commandLink actionListener="#{editor.bookEdited}"
                action="categoryEdit">
    <h:outputText value="Dodaj kategorie" />
</h:commandLink>
</h:form>
</f:view>
```

# Warstwa wizualna – pułapki

- W większości wypadków tagi JSF akceptują jedynie tagi JSF ☺:

```
<h:commandLink actionListener="#{editor.bookEdited}"  
                action="categoryEdit">
```

**JSF jest cudowne ;)**

```
</h:commandLink>
```

**JSF jest cudowne ;)** `<a href=„#” onclick=„...”></a>`

# Warstwa wizualna - pułapki

```
<f:view>
```

```
<h:form>
```

```
  <h:outputLabel for="name">
```

```
    <h:outputText value="Name:" />
```

```
  </h:outputLabel>
```

```
<h:inputText id="name" /> <h:commandButton value="Submit" />
```

```
</h:form>
```

```
</f:view>
```

- Za pierwszym razem Label nie jest wyświetlony ponieważ kontrolka name nie znalazła się jeszcze w drzewie komponentów

# Wiązanie komponentów z obiektami

- W JSF obiekty są tworzone i umieszczane w danym zakresie na żądanie
  - JSF zawiera prosty kontener IoC, ale dzięki VariableResolver możemy wykorzystać np. Spring
  - Komponenty mogą być związane z różnymi obiektami, znajdującymi się w różnych zakresach

# JSF EL

- Wiązanie z danym polem obiektu:  
`h:inputText value="#{ editor.book.title }" />`
- Obsługa nawigacji  
`<h:commandLink action="#{ editor.decide }">`  
metoda ma postać  
`public String decide() { }`

# JSF EL

- Obsługa zdarzeń

```
<h:commandLink
```

```
actionListener="#{ editor.bookEdited }">
```

metoda ma postać

```
public void bookEdited(ActionEvent event){ }
```

- W kodzie łączymy się z JSF przez statyczną metodę `FacesContext.getCurrentInstance()`

# JSF IoC

```
<managed-bean>  
<managed-bean-name>menu</managed-bean-name>  
<managed-bean-class>  
pl.poznan.mmalecki.lib.controller.Menu  
</managed-bean-class>  
<managed-bean-scope>session</managed-bean-scope>  
<managed-property>  
<property-name>categoryFactory</property-name>  
<value>#{categoryFactory}</value>  
</managed-property>  
</managed-bean>
```

# Nawigacja

- Nawigacja jest określona przez tekstową wartość atrybutu action
  - action=„success”, lub
  - action=„#{bean.method}”

## Konfiguracja:

```
<navigation-case>  
    <from-outcome>bookDetails</from-outcome>  
    <to-view-id>/book.jsp</to-view-id>  
</navigation-case>
```

# Nawigacja – cd.

- W przypadku nieistniejącego `<from-outcome>` ponownie wyświetlona jest strona bieżąca
- JSF nie umożliwia ukrycia stron JSP w WEB-INF

# Zdarzenia

- Zdarzenia nie umożliwiają nawigowania pomiędzy stronami
- Jeden listener dodajemy przez atrybut `actionListener=„#{ }”`
- Więcej listenerów – zagłębiony `<f:actionListener type=„”>` którego atrybutem jest klasa implementująca `ActionListener` – **WADA!!!**
- `ValueChangeListener` – wywołany zmianą wartości pola
- `PhaseListener` – wywoływany podczas wskazanego etapu przetwarzania żądania

# Walidatory

- JSF zawiera mechanizm walidatorów, niestety jedynie server-side
- Przyciski i odnośniki zawierają atrybut `immediate`, pozwalający uniknąć walidacji (gdy wykonujemy akcję nie związaną z danymi, np. zmianę preferencji użytkownika)
- Brak możliwości walidacji wielu pól razem (wymaga stworzenia własnego walidatora)

# Walidatory

- `<h:inputText id="amount" size="8" required="true" value="#{entryHandler.currentEntry.amount}">  
    <f:validateDoubleRange minimum="1"/>  
</h:input_text>`
- Stworzenie własnego walidatora wymaga rejestracji w konfiguracji i stworzenia nowego znacznika JSP (tld ☹)

# Konwertery

- Mechanizm konwerterów – klas implementujących interfejs Converter
- Dwie metody – `getAsObject` i `getAsString`
- Konweterów nie można konfigurować przez IoC ☹

# Podsumowanie

*So I beg of you all, don't use JSF, boycott it, avoid it all you can.  
Do not let it thrive or prosper.  
We don't want it improved or tweaked,  
we want it to die the horrible painful death it so richly deserves.*

[http://www.jroller.com/page/fate/20040920#boycott\\_jsf](http://www.jroller.com/page/fate/20040920#boycott_jsf)

# Zalety

- Bajecznie prosta obsługa części z aspektów WWW – zdarzenia, dowolność wiązania z obiektami, możliwość programowej manipulacji komponentami
- Istniejące komponenty (Oracle, MyFaces)
- Wizualne edytory – Java Studio Creator, JDeveloper (podobno ;))

# Wady

- Błędy w specyfikacji
- Niedojrzała implementacja Apache MyFaces (<http://myfaces.org>)
- Mała liczba produkcyjnych aplikacji wykorzystujących jsf

# O czym nie wspomniałem?

- Etapy przetwarzania żądań
- Mechanizm komunikatów
- Dane tabelaryczne
- Umiejdzynarodowienie
- Własne renderery
- Spring-JSF

Dziękuję za uwagę

*A kto opowiedziałby o Tapestry??*