Zaawansowane Aplikacje Internetowe

Struts 2 framework

Piotr Mazur

Katedra Mikroelektroniki i Technik Informatycznych

Łódź, 13 października 2010

Informacje o przedmiocie

Dokładne informacje o przedmiocie znajdują się na stronie:

`http://neo.dmcs.p.lodz.pl/zswww/`

Kontakt

Wykład prowadzi: mgr inż. **Piotr Mazur**
Doktorant w Katedrze Mikroelektroniki i Technik Informatycznych
Email: **pmazur@dmcs.pl**
Telefon: 42 631-26-51

Warunki zaliczenia

Warunkiem zaliczenia wykładu jest pomyślne zrealizowanie projektu oraz zaliczenie kolokwium wykładowego. Projekt będzie realizowany samodzielnie.
Należy zastosowań następujące biblioteki/technologie:

- Spring Framework 3.0.
- Hibernate ORM.
- Struts 2.
- Spring Security

Tematyka wykładu

Budowanie aplikacji internetowych w oparciu o otwarte, ogólnodostępne narzędzia korzystające z platformy J2EE.

- Spring Framework jako przykład kontenera IoC
- Spring Security jako biblioteka zabezpieczająca aplikację internetową
- Struts 2 jako silnik MVC
- Hibernate jako przykład biblioteki mapowania obiektowo-relacyjnego

Java Beans

JavaBeans oraz EJB (Enterprise Java Beans) jest próbą usystematyzowania typów obiektów wykorzystywanych w języku Java.
W założeniach twórców obiekty tworzone podczas uruchamiania aplikacji J2EE można pogrupować w kategorie.
Przykładowe kategorie obiektów.

- Singleton
- Obiekt sesyjny (Stateful/Stateless Session Bean)
- Obiekt typu Entity (Entity Bean)
- Obiekt sterowany komunikatami (Message Driven Bean)

Java Beans

Każdy obiekt typu Bean musi spełniać pewne wymagania:

- Bezargumentowy konstruktor
- Wszystkie pola powinny być zakresu prywatnego
- Dostęp do pól odbywa się za pomocą akcesorów get/set

Java Beans

Uzasadnienie wymagań:

- Z początku metody typu get/set miały umożliwić kontrolę dostępu do pól klasy
- Z czasem umożliwiły jednak usprawnianie działania beanów za pomoca generacji kodu
- Bezargumentowy konstruktor umozliwia automatyczne tworzenie nowych instancji obiektów

Proxy

Większość bibliotek opisywanych na wykładzie korzysta z tzw: Proxy aby usprawnić działanie beanów.
Korzystanie z obiektów typu proxy pozwala na wstawienie dodatkowego kodu do klasy.
Przykłady proxy:

- JDK 1.4 Proxy (Interfejs)
- CGLIB Proxy
- JavaAssist Proxy

Spring Framework

Spring Framework jest (między innymi) kontenerem typu IoC (Inversion of Constrol) umożliwiającym tzw: Wstrzykiwanie Zależności (Dependency Injection - DI).

- Konfiguracja za pomocą plików XML i/lub annotacji
- Deklaratywna transakcyjność oraz współpraca z wieloma popularnymi bibliotekami ORM **(@Transactional)**
- Warstwa MVC do budowania aplikacji internetowych opartych na platformie J2EE
- Dodatkowe klasy usprawniające pracę aplikacji (Obsługa wysyłania wiadomości E-Mail, Raportowanie itd..)

Spring Security

Biblioteka powstała z włączenia ACEGI Security do projektu Spring.

- Udostępnia interfejs zabezpieczania aplikacji oparty na rolach oraz uprawnieniach (ACL - Access Control List).
- Korzysta z AOP w celu umożliwienia zabezpieczania wykonania metod (annotacja **@Secured**).
- Umożliwia kontrolę dostępu z poziomu kontrolera MVC
- Kontrola dostępu z poziomu widoku: tag: **ifallgranted**

Struts 2

Implementacja modelu MVC przygotowywana przez Apache Software Foundation.

- Prosta w konfiruacji
- Mechanizm oparty na wykonywaniu akcji
- Implementacja kontrolera w oparciu o przestrzenie nazw oraz wyniki akcji
- Intensywne wykorzystanie polityki **Convention over Configuration**

Struts 2 Framework

Framework przygotowany przez **Apache Software Foundation** powstały z połączenia frameworków **WebWork 2** oraz **Struts**

- Podejście do aplikacji internetowych na zasadzie **Akcja - Rezultat**
- Grupowanie akcji za pomocą **pakietów**
- Rozbudowany system wyzwalaczy **(Interceptors)**
- Podejście **“Convention over Configuration”**

Przykładowy plik konfiguracyjny

struts.xml

```
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
</struts>
```

Przy budowaniu aplikacji zaleca się uruchomienie trybu deweloperskiego:

struts.xml

```
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
  <constant name="struts.devMode" value="true" />
</struts>
```

Plik konfiguracyjny **struts.xml** może zawierać wiele sekcji, przykłady:

- Sekcja pakietów **package**
  - Sekcja definicji akcji **action**
- Sekcja interceptorów **interceptors**
- Sekcja wyników **result-types**
- Sekcja obsługi wyjątków **global-exception-mappings**

Sekcje opisane są szczegółowo na stronie **http://struts.apache.org/2.x/docs/configuration-elements.html**

Wraz z rozwojem Struts 2 kolejne części pliku konfiguracyjnego zostały sukcesywnie zastępowane annotacjami. Obecnie większość konfiguracji może odbywać się bez ingerencji w plik konfiguracyjny. Pozostają jednak elementy które powinny się w pliku znaleźć np:
stałe

Przykłady stałych

```
<constant name="struts.devMode" value="true" />
<constant name="struts.action.extension" value="do" />
<constant name="struts.url.includeParams" value="none" />
```

Definicje stałych nie są prezycyjnie określone, nowe stałe mogą być definiowane w dodatkowych **pluginach**

Migracja konfiguracji z pliku do annotacji, oraz wykorzystywanie **konwencji** powiązanej z poszczególnymi elementami Struts 2 to przykład **“Convention over Configuration”**

Polityka ta została wprowadzana wraz z kolejnymi wersjami frameworku Struts 2 w postaci pluginów **codebehind** oraz zastępującego go **convention plugin**
Aby dodać plugin do Struts 2 wystarczy zwykle przegrać odpowiedni plik .jar - funkcjonalność zostanie automatycznie uruchomiona przy starcie frameworka.

Przydatne pluginy Struts 2

- Convention Plugin - uruchamia mechanizmy **Convention over configuration**
- Config Browser Plugin - panel administracyjny prezentujący listę akcji, wyników oraz interceptorów
- Tiles Plugin/Sitemesh Plugin - pluginy integrujące Struts 2 z bibliotekami ułatwiającymi budowanie rozkładu stron
- Spring Plugin - integracja ze **Spring Framework**
- JSON Plugin - przdstawienie wyniku w postaci **JSON**
- JasperReports Plugin - obsługa raportowania za pomocą biblioteki **JasperReports**

Przykładowa aplikacja korzystająca z **Convention over configuration**

Plik konfiguracyjny

```
  <!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
   <constant name="struts.devMode" value="true" />
   <constant name="struts.convention.action.packages" value="org.app"/>
</struts>
```

Stała **struts.convention.action.packages** pozwala na ustawienie głównego pakietu do poszukiwania akcji. Framework przeskanuje wszystkie klasy znajdujące się w tym pakiecie w poszukiwaniu akcji które można wykonać.

Przykładowa klasa akcji

org.app.actions.Test.java

```
package org.app.actions;

public class Test extends ActionSupport{

   @Override
   public String execute() throws Exception {
      System.out.println("Przykładowa akcja Struts 2");
      return SUCCESS;
   }
}
```

Jeżeli został uaktywniony plugin **convention** akcja jest od razu dostępna pod adresem:

http://serwer/aplikacja/actions/test.action

Nazwa pakietu względem stałej **struts.convention.action.packages** oraz nazwa klasy zostały uwzględnione w adresie. Nazwa akcji podlega konwersji do **camelcase**

W poprzednim przykładzie nazwa pakietu względem stałej **struts.convention.action.packages** to tzw: “Przestrzeń Nazw” **(namespace)**

Przestrzenie nazw bezpośrednio mapują się na adresy URL
Przestrzeń nazw każdej akcji można nadpisać: dodając do klasy

akcji annotację **@Namespace**

```
@Namespace("/mynamespace")
public class Test extends ActionSupport{

   @Override
   public String execute() throws Exception {
      System.out.println("Przykładowa akcja Struts 2");
      return SUCCESS;
   }
}
```

Co się stanie po wykonaniu akcji?

Po wykonaniu akcji następuje wykonanie jednego z możliwych **rezultatów**
Jeżeli uaktywniony jest **convention plugin** a nie został zdefiniowany rezultat akcji zostanie automatycznie zaprezentowana strona:

```
\WEB-INF\content\przestrzen_nazw\nazwa_akcji.jsp
```

Przykład akcji ze zdefiniowany rezultatem

```
@Result(name="success", location="aaa.jsp")
public class Test extends ActionSupport{

   @Override
   public String execute() throws Exception {
      System.out.println("Przykładowa akcja Struts 2");
      return SUCCESS;
   }
}
```

Korzystając z konfiguracji w poprzednich przykładach zostanie zaprezentowany plik:

```
/WEB-INF/content/actions/aaa.jsp
```

Wykonanie danego rezultatu zależy od wartości zwróconej przez metodę **execute()**

Wartośc zwrócona przez metodę **execute()** jest typu **String**

Przykład akcji z wieloma rezultatami

```
@Results({
   @Result(name="success",location="aaa.jsp"),
   @Result(name="input", location="bbb.jsp"),
   @Result(name="myresult",location="myresult.jsp")
})
public class Test extends ActionSupport{

   private Integer test=0;

   @Override
   public String execute() throws Exception {
      if (test==0) return SUCCESS;
      if (test>1 && test<10) return INPUT;
      else return "myresult";
   }

   public Integer getTest() {
     return test;
   }

   public void setTest(Integer test) {
      this.test = test;
   }
}
```

W powyższym przykładzie została wprowadzona zmienna typu Integer **test**

Zmienna ta zostanie automaytycznie ustawiona przez framework na jedną z wartości:

- Przekazanej w adresie url: **test=6**
- Przekazanej poprzez formularz

```
<input type="text" name="test" />
```

Struts 2 posiada kilka wbudowanych typów rezultatów

- DispatcherResult - domyślny
- StreamResult - jeżeli resultatem ma być strumień binarny (np: download pliku, obrazek)
- PlainTextResult - serwuje statyczne pliki tekstowe/html
- RedirectActionResult - przekierowanie na inną stronę

Przekierowanie do innej akcji

```
@Result(name="myresult", type="redirectAction", params={"actionName","zosia"})
```

Strona wyświetlona poprzez uruchomienie **rezultatu** jest zwykłą stroną JSP
Struts 2 dostarcza rozbudowaną bibliotekę tagów, które integrują się z frameworkiem - nie ma jednak konieczności korzystania z nich

strona.jsp - wyświetlenie wartości zmiennej test z akcji

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags" %>

Wartość zmiennej test: <s:property value="test" />
```

Prosty formularz

form.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags" %>


<s:form action="formularz">
   <s:textfield name="zmienna"></s:textfield>
   <s:submit value="OK"></s:submit>
</s:form>
```

Po wysłaniu formularza zostaniemy przekierowani do akcji **formularz** jeżeli ta akcja będzie posiadała pole **zmienna** zostanie ono ustawione wartością z formularza.

The End