

[Radostaw Holewa](#)

Wprowadzenie do Struts 2

Część I

**Konfiguracja środowiska developerskiego NetBeans
i uruchomienie przykładowej aplikacji na serwerze
aplikacyjnym Glassfish**

1. Słów kilka o tutorialu

Tutorial ten powstaje w celu przybliżenia rozwiązań takich jak framework Struts 2, środowisko programistyczne NetBeans, serwer aplikacyjny Glassfish oraz frameworki Spring 2 i Hibernate.

Tutorial będzie się składał z dziesięciu części w których poruszone zostaną kolejne aspekty frameworku Struts 2, w pierwszej części opisuję sposoby konfiguracji środowiska NetBeans 6, zarządzania serwerem Glassfish z poziomu NetBeans oraz utworzenia trywialnej aplikacji z użyciem frameworku Struts 2, którą następnie uruchomimy na serwerze Glassfish.

2. Pobranie i instalacja potrzebnego oprogramowania

Jak już wspomniałem, w części pierwszej tutoriala zajmiemy się konfiguracją środowiska developerskiego NetBeans 6.0 w wersji Beta 2 oraz uruchomieniem na serwerze aplikacyjnym Glassfish V2 przykładowej aplikacji napisanej z użyciem frameworku Struts 2.

W celu rozpoczęcia pracy z tutorialiem należy pobrać (proszę kliknąć na nazwę) :

- [NetBeans IDE 6.0 Beta 2](#)
- [Glassfish V2](#)
- [Struts 2](#)

W chwili pisania tutoriala używałem wersji 2.0.11 frameworku Struts 2 oraz wersji V2 b58g serwera aplikacyjnego Glassfish.

Ponieważ do instalacji serwera Glassfish konieczne jest wykonanie skryptu Anta o nazwie setup.xml, należy również pobrać Anta w wersji co najmniej 1.6.5 :

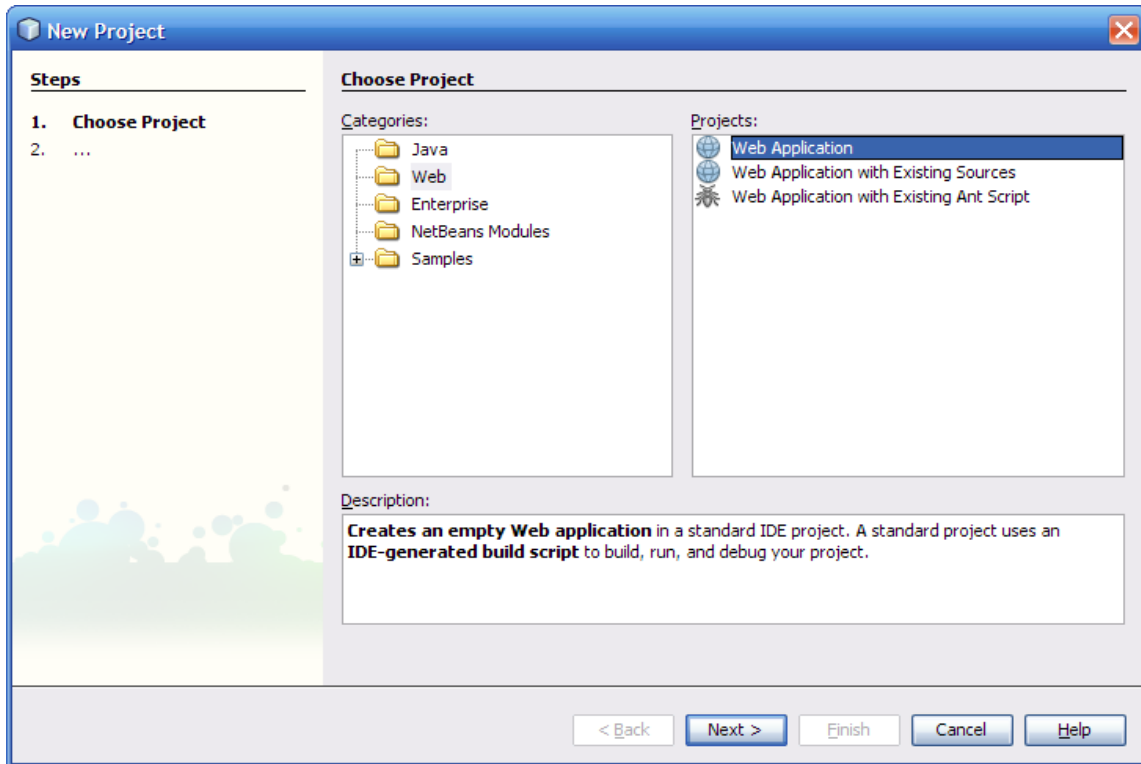
- [Ant](#)

Kolejne kroki instalacji serwera Glassfish zostały opisane [tutaj](#).

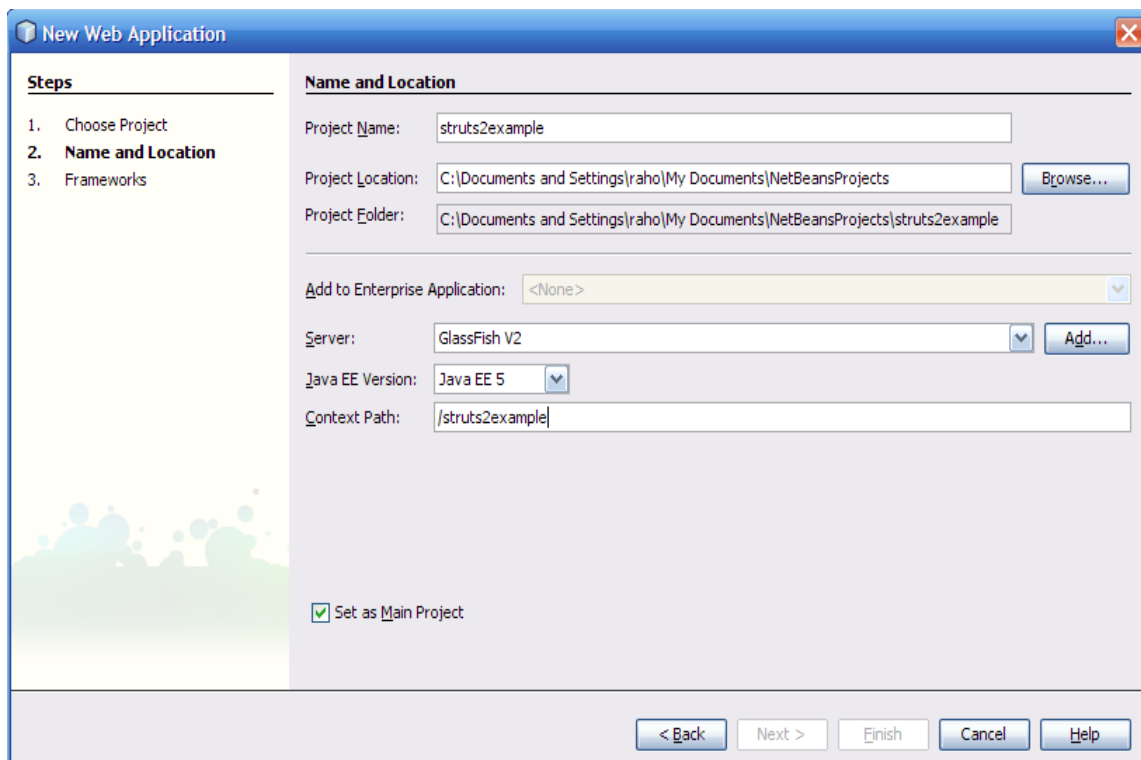
Jeśli już pobrałeś i zainstalowałeś wymienione wyżej oprogramowanie możesz przystąpić do kontynuowania tutoriala.

3. Utworzenie i konfiguracja projektu w środowisku NetBeans

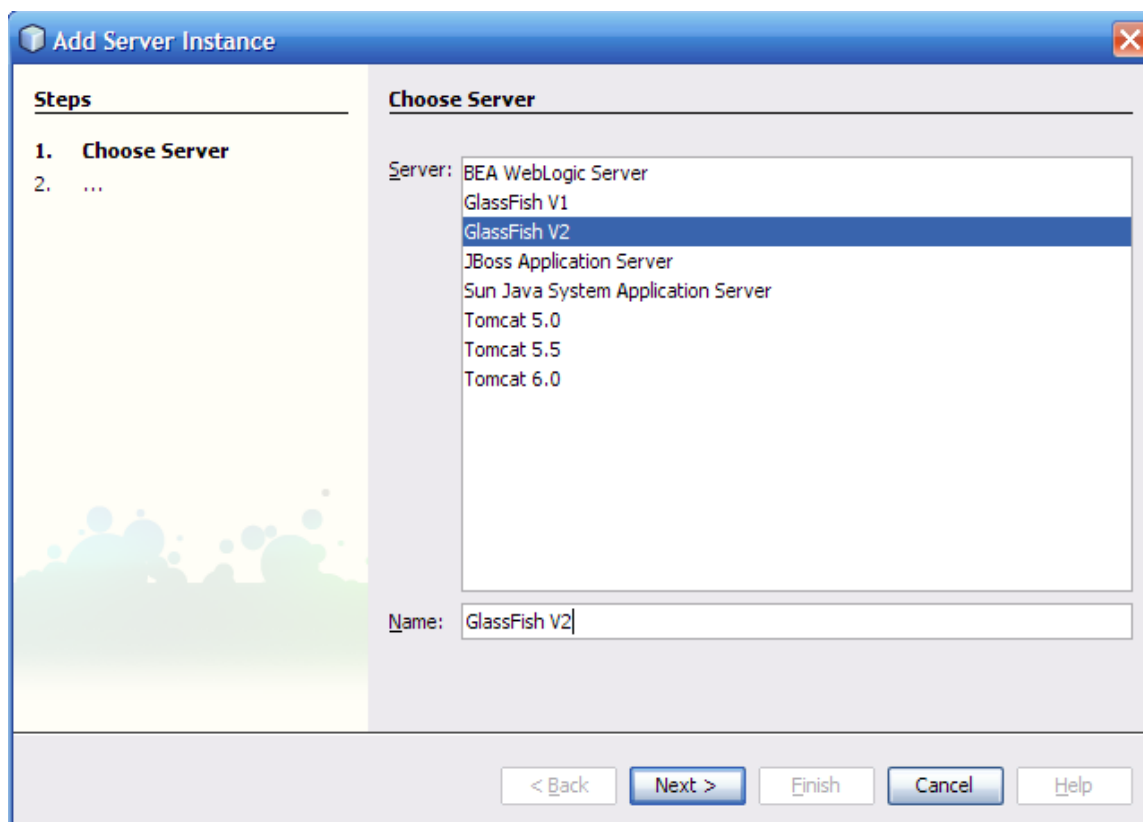
W pierwszym kroku utworzymy projekt aplikacji webowej, w tym celu z menu NetBeans wybieramy **File -> New Project** bądź też wciskamy jednocześnie kombinację klawiszy **Ctrl + Shift + N**, w oknie które nam się pojawiło wybieramy **Web -> Web Application**:



Po wybraniu wspomnianych opcji i kliknięciu na przycisk **Next** pojawi się następujące okienko:



Wpisujemy w nim nazwę naszego projektu, wybieramy lokalizację w której zostanie zapisany projekt oraz definiujemy typ serwera na jakim będzie uruchomiona nasza aplikacja, jak już wspomniałem w tutorialu, użyjemy serwera aplikacyjnego Glassfish V2. Jeśli nie masz możliwości wybrania serwera Glassfish z listy dostępnych serwerów kliknij przycisk **Add...** po czym zobaczysz następujące okienko :

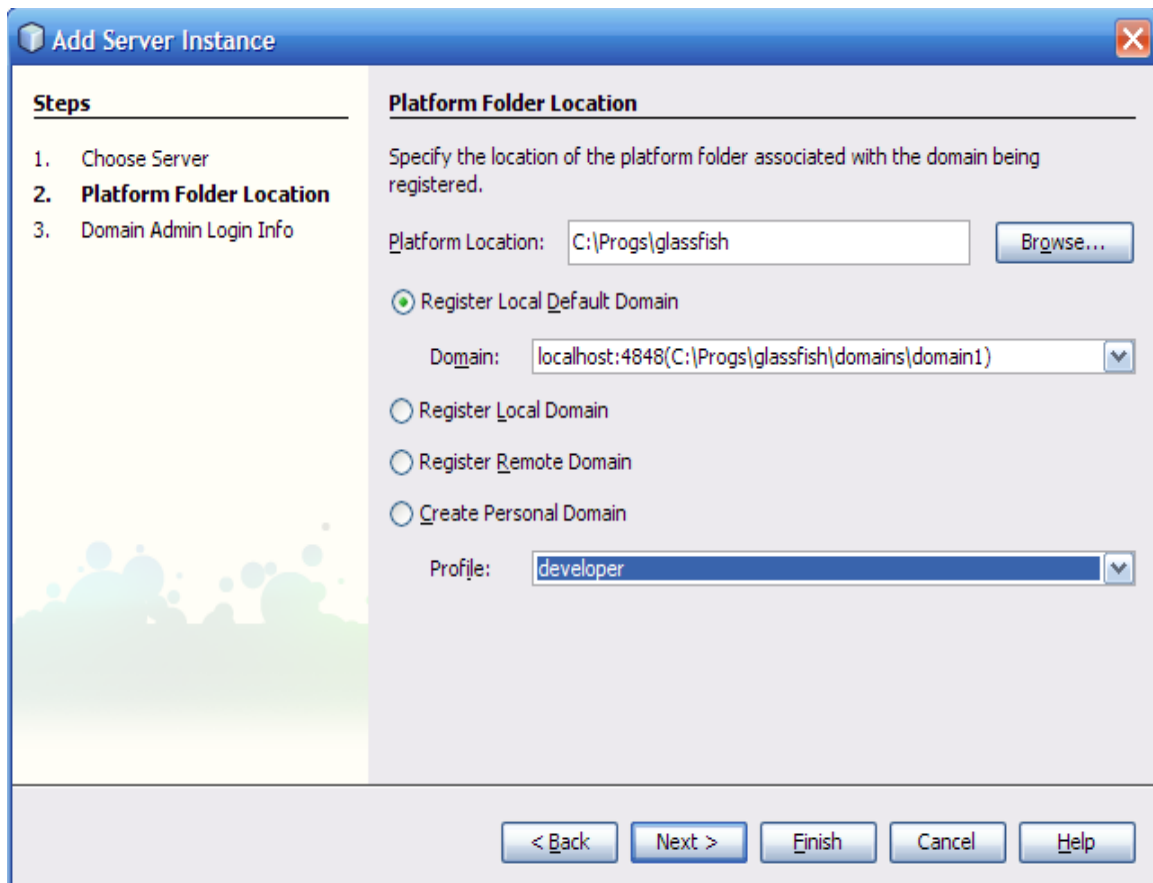


Wybierz serwer zgodnie z obrazkiem przedstawionym powyżej a następnie kliknij na przycisk **Next**.

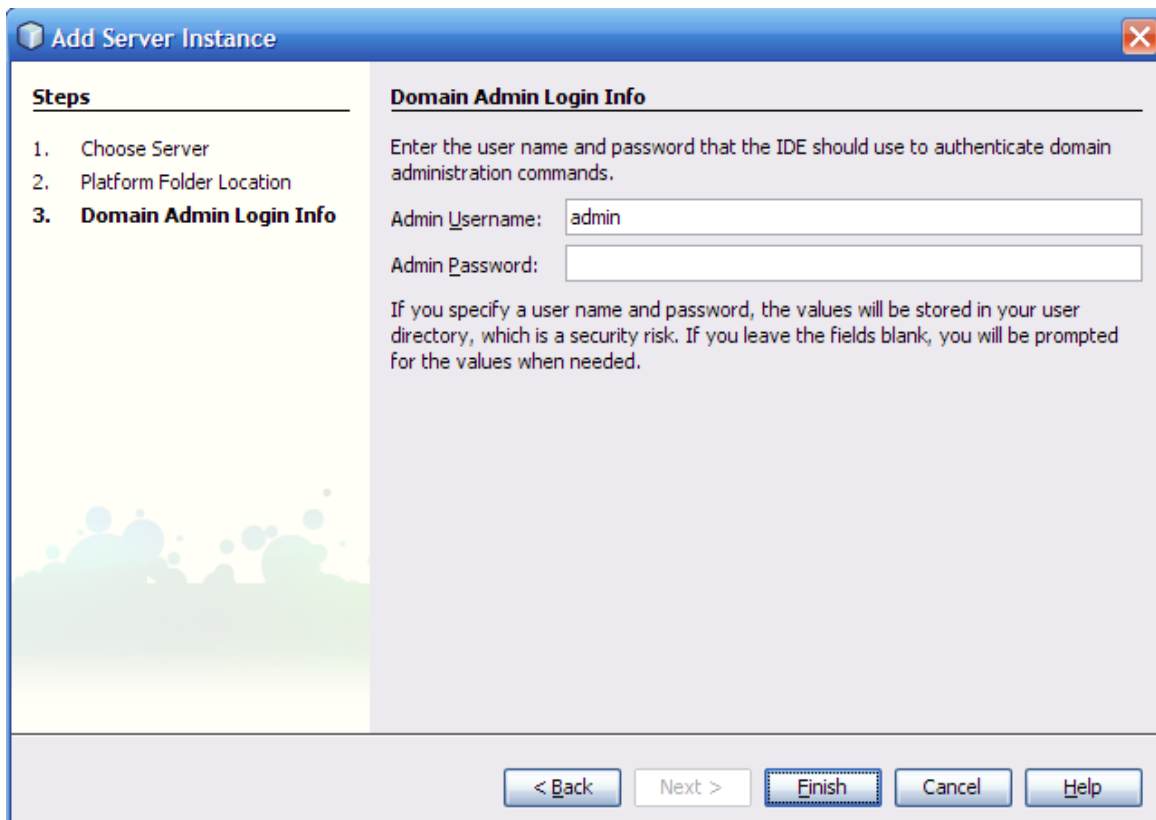
Oczywiście prezentowaną w tym tutorialu aplikację można uruchomić na innym serwerze aplikacyjnym, bądź też samym kontenerze servletów jak choćby Apache Tomcat. Szczególnie zalecam uruchomienie aplikacji na serwerze Apache Geronimo, niestety NetBeans nie posiada wbudowanej obsługi tego serwera, z pomocą przychodzi nam Jacek Laskowski który utworzył taką wtyczkę.

Więcej informacji na temat obsługi Geronimo w NetBeans znajdziecie [tutaj](#).

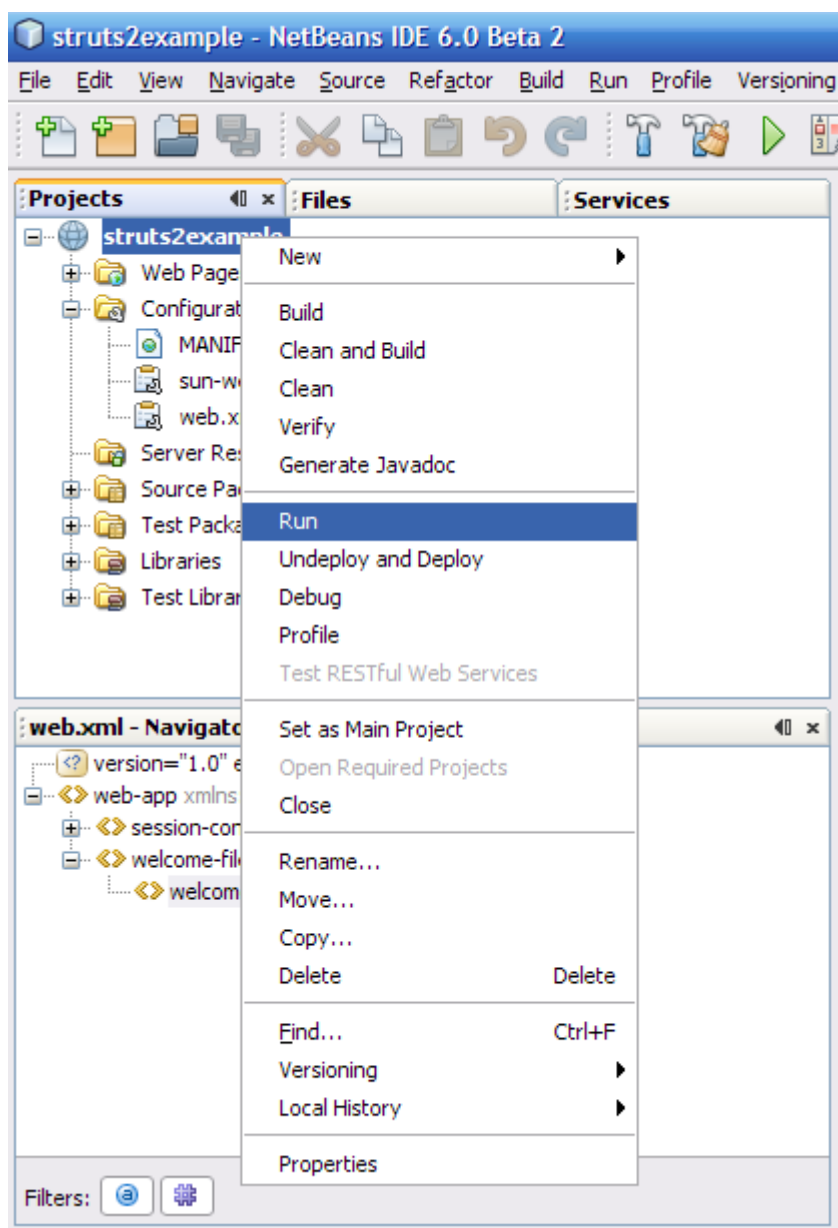
Kolejnym krokiem jest ekran na którym będziesz mógł wybrać lokalizację w której zainstalowałeś Glassfisha oraz skonfigurować ustawienia domeny, proponuję abyś wybrał opcje w sposób analogiczny do przedstawionego poniżej ekranu :



Kliknij na przycisk **Next** a pojawi się ekran gdzie zdefiniujesz login i hasło za pomocą którego środowisko uzyska dostęp do serwera, standardowy login to **admin** , hasło **adminadmin** :



Gdy klikniesz **Finish** powrócisz do formatki konfiguracji projektu, na Twojej liście wyboru dostępnych serwerów pokaże się Glassfish, którego będziesz mógł teraz wybrać. Kliknij **Finish** Twój projekt jest już skonfigurowany. Możesz teraz go uruchomić, w tym celu kliknij prawym przyciskiem myszy na nazwie projektu i wybierz opcję **Run** :

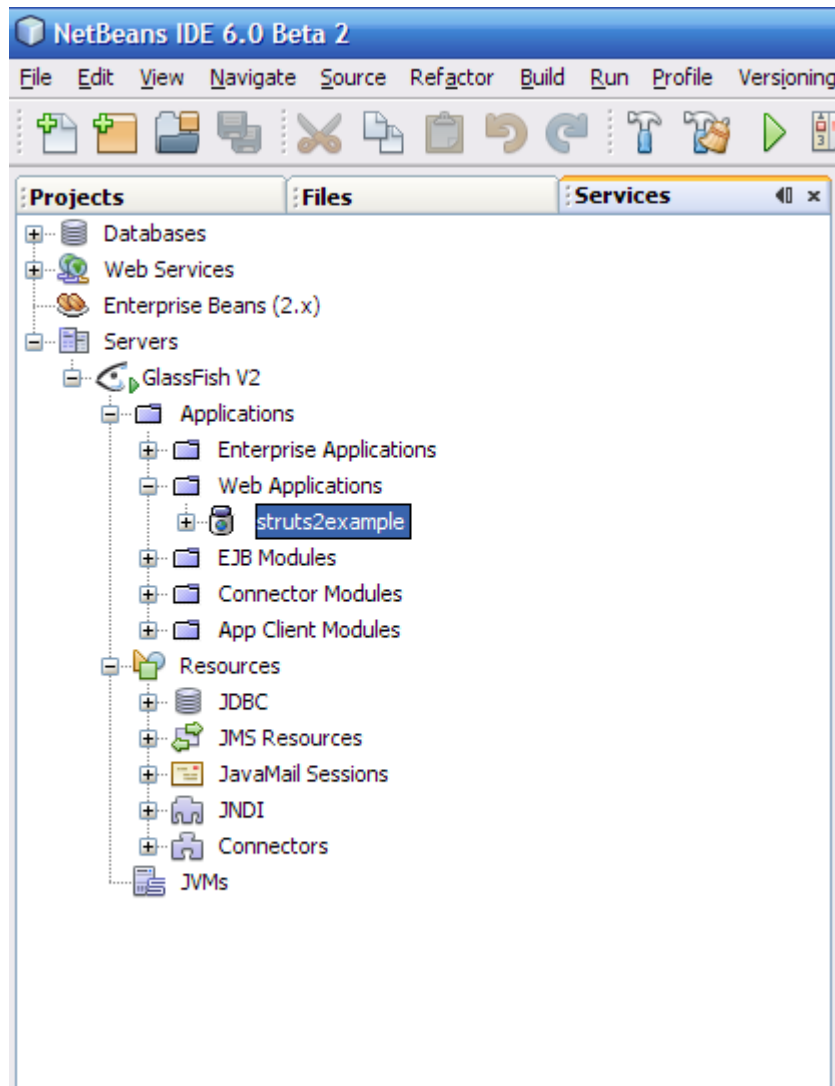


W konsoli poniżej zobaczysz jak buduje się Twoja aplikacja, a następnie pojawią się logi serwera aplikacyjnego Glassfish.

Jeśli przeglądarka nie otworzy się automatycznie to uruchom ją ręcznie i wpisz w pasku adresu <http://localhost:8080/struts2example> a Twoim oczą ukaże się strona z napisem *Hello World!* to znak, że Twoja aplikacja działa!

Swojego Glassfisha możesz obsługiwać na dwa sposoby :

- Wybrać zakładkę **Services** a następnie w drzewku odnaleźć serwer Glassfish :



- Zalogować się poprzez przeglądarkę do konsoli administracyjnej

W tym celu uruchom w przeglądarce następujący adres : <http://localhost:4848/login.jsf> a następnie zaloguj się jako użytkownik **admin** i podaj hasło **adminadmin** , ukaże Ci się konsola administracyjna.

Warto poprobać zmieniać ustawienia Glassfisha, w celu bliższego zapoznania się z tym serwerem zalecam przeczytanie dokumentacji którą możesz znaleźć [tutaj](#).

4. Dodanie Struts 2 do naszego projektu i stworzenie trywialnej aplikacji

W tym punkcie pierwszej części tutoriala dodamy do naszego projektu framework Struts 2, a następnie utworzymy przykładową, bardzo prostą aplikację, która po wpisaniu naszego imienia wyświetli komunikat *Hello wpisanę_imię*.

Najpierw, aby w ogóle móc korzystać z frameworku Struts 2 musimy dodać następujące biblioteki do naszego projektu :

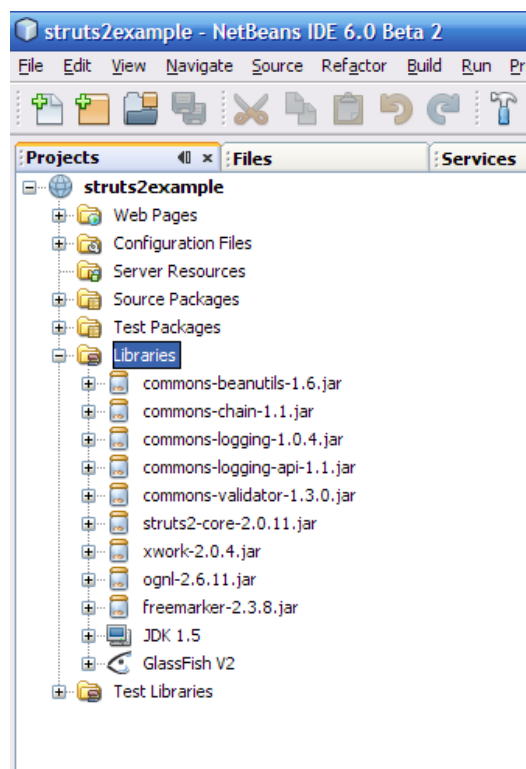
- commons-beanutils-1.6.jar
- commons-chain-1.1.jar
- commons-logging-1.0.4.jar
- commons-logging-api-1.1.jar
- commons-validator-1.3.0.jar
- freemarker-2.3.8.jar
- ognl-2.6.11.jar
- struts2-core-2.0.11.jar
- xwork-2.0.4.jar

Wymienione biblioteki dostępne są w paczce zawierającej framework Struts 2 którą pobrałeś na początku tutoriala.

Aby je dodać do projektu należy w zakładce **Projects** kliknąć prawym klawiszem myszki na **Libraries**, a następnie wybrać z menu które się pojawi opcję **Add JAR/Folder...**, w okienku które się pojawi odszukaj bibliotekę i kliknij **Open**.

Gratulacje, biblioteka została dodana do projektu, teraz musisz dodać pozostałe biblioteki.

Gdy już dodałeś wszystkie wymienione wyżej biblioteki, struktura Twojego projektu w NetBeans powinna wyglądać następująco :



Teraz skonfigurujemy nasz deployment descriptor, czyli plik **web.xml**, dodajmy do niego wpisy tak aby przyjął następującą postać :

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3.     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4.     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5.         http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
6.
7.     <filter>
8.         <filter-name>struts</filter-name>
9.         <filter-class>
10.             org.apache.struts2.dispatcher.FilterDispatcher
11.         </filter-class>
12.         <init-param>
13.             <param-name>actionPackages</param-name>
14.             <param-value>org.holewa.struts2.example</param-value>
15.         </init-param>
16.     </filter>
17.
18.     <filter-mapping>
19.         <filter-name>struts</filter-name>
20.         <url-pattern>/*</url-pattern>
21.     </filter-mapping>
22.
23.     <session-config>
24.         <session-timeout>
25.             30
26.         </session-timeout>
27.     </session-config>
28.
29.     <welcome-file-list>
30.         <welcome-file>index.jsp</welcome-file>
31.     </welcome-file-list>
32.
33.</web-app>
```

Zmodyfikujmy również stronę **index.jsp** do następującej postaci :

```
1. <%@ page contentType="text/html" pageEncoding="UTF-8"%>
2. <%@ taglib prefix="s" uri="/struts-tags" %>
3.
4. <html>
5.     <head>
6.         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7.         <title>Struts 2 Example</title>
8.     </head>
9.     <body>
10.         <s:form action="ShowName">
11.             <s:textfield name="name"/>
12.             <s:submit />
13.         </s:form>
14.     </body>
15.</html>
```

Kod zawarty w ciele taga **<s:form/>** to definicja formy która zostanie przesłana podczas requestu do serwera, jak widzimy zostanie wywołana akcja **ShowName**, podczas requestu przesłana zostanie również wartość z pola input, która zostanie przypisana do zmiennej **name** w wywoływanej akcji .

Dodamy także nową stronę *name.jsp* :

```
1. <%@ page contentType="text/html" pageEncoding="UTF-8"%>
2. <%@ taglib prefix="s" uri="/struts-tags" %>
3.
4. <html>
5.     <head>
6.         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7.         <title>Struts 2 Example</title>
8.     </head>
9.     <body>
10.        Hello <s:property value="name" />
11.    </body>
12.</html>
```

Jak widzimy oprócz tagów HTML użyliśmy również taga Struts 2 `<s:property value="name" />` który wyświetli wartość jaka jest przypisana do zmiennej *name* w akcji Struts 2 z której nastąpiło przekierowanie na stronę *name.jsp*.

Teraz czas na konfigurację Struts 2, tworzymy plik *struts.xml* :

```
1. <!DOCTYPE struts PUBLIC
2.     "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
3.     "http://struts.apache.org/dtds/struts-2.0.dtd">
4. <struts>
5.     <package name="example" extends="struts-default">
6.         <action name="Home" class="org.holewa.struts2.example.MainAction">
7.             <result>/index.jsp</result>
8.         </action>
9.         <action name="ShowName" method="showName"
10.            class="org.holewa.struts2.example.MainAction">
11.             <result>/name.jsp</result>
12.         </action>
13.     </package>
14.</struts>
```

W powyższym pliku konfiguracyjnym zdefiniowaliśmy dwie akcje korzystające z metod zawartych w klasie *MainAction* którą za chwilę utworzymy. Pierwsza akcja o nazwie *Home* przekieruje nas do strony *index.jsp*, natomiast akcja o nazwie *ShowName* przekieruje nas do strony *name.jsp* która wyświetli napis *Hello wpisane_imię* .

Oto klasa *MainAction* :

```
1. package org.holewa.struts2.example;
2.
3. import com.opensymphony.xwork2.ActionSupport;
4.
5. public class MainAction extends ActionSupport {
6.
7.     private String name;
8.
9.     @Override
10.    public String execute() throws Exception {
11.        return SUCCESS;
12.    }
13.
14.    public String showName() throws Exception {
15.        return SUCCESS;
16.    }
```

```

17.
18.     public String getName() {
19.         return name;
20.     }
21.
22.     public void setName(String name) {
23.         this.name = name;
24.     }
25.
26.}

```

Jak widzimy klasa jest bardzo prosta, żeby nie powiedzieć trywialna. Jednak jest to wszystko czego potrzebujemy aby zrealizować planowaną funkcjonalność naszej aplikacji.

Metody *execute()* i *showName()* to metody użyte w zdefiniowanych przez nas akcjach Struts 2. Jeśli spojrzymy do pliku *struts.xml* to zobaczymy, że dla akcji *Home* nie mamy zdefiniowanej metody, a dla akcji *ShowName* podaliśmy metodę *showName()*. To dlatego, że akcja *Home* używa domyślnej metody *execute()* a akcja *ShowName* odpowiada metodzie *showName()*, gdybyśmy zdefiniowali akcję *ShowName* w następujący sposób :

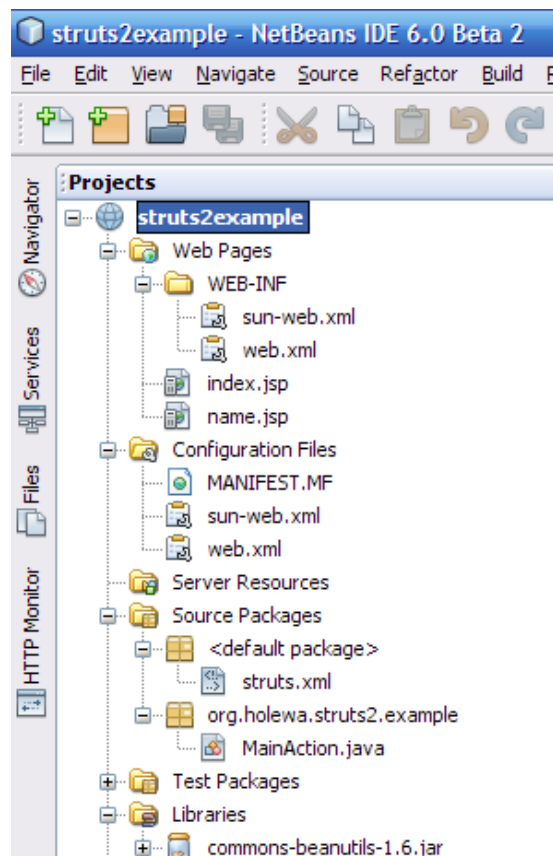
```

1. <action name="ShowName" class="org.holewa.struts2.example.MainAction">
2.     <result>/name.jsp</result>
3. </action>

```

to wywołanie akcji *ShowName* oznaczałoby wywołanie metody *execute()*, ponieważ w chwili obecnej obie metody mają identyczną postać to efekt działania aplikacji byłby ten sam.

Utworzona struktura powinna mieć następującą postać :



Teraz pozostaje tylko uruchomić naszą aplikację, klikamy prawym klawiszem myszy na nazwie projektu, wybieramy z menu opcję **Clean and Build** a następnie po zbudowaniu aplikacji wybieramy w tym samym menu na opcję **Run**.

Po chwili oczekiwania naszym oczą powinna ukazać się następująca strona :

A screenshot of a web form. It consists of a single-line text input field with a light gray border and a blue button with the word "Submit" in white text.

Teraz pozostaje już tylko wpisać imię i kliknąć na przycisk **Submit** .
Jeśli zobaczyłeś napis **Hello wpisane_imię** to znak, że dotarłeś do końca tutorialu.

5. Słowo końcowe

W razie problemów informuj mnie o tym na [moim blogu](#). Obiecuję, że postaram się rozpatrywać wszelkie uwagi jakie otrzymam, jeśli tylko coś wydaje Ci się niejasne bądź też zbyt słabo opisane to daj mi koniecznie o tym znać.