

Celem niniejszego ćwiczenia jest przedstawienie podstaw tworzenia aplikacji webowych za pomocą technologii JSP. Po wykonaniu ćwiczeń student posiędzie wiedzę jak utworzyć prostą dynamiczną stronę JSP, która pobiera parametry przekazane przez użytkownika i wykorzystuje mechanizm sesji.

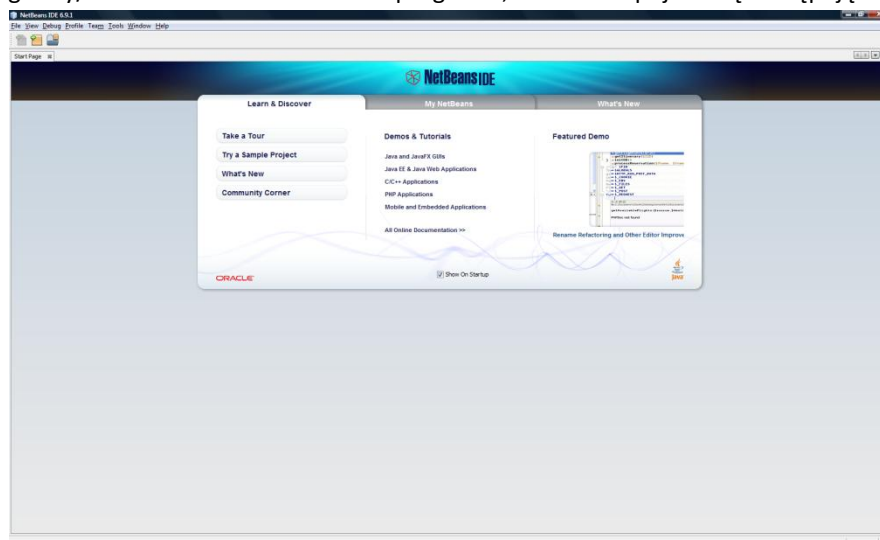
Ćwiczenie należy wykonywać w następujący sposób: przekopiowywać i wykonywać wszystkie przykłady, żeby zobaczyć ich wynik działania, oraz ewentualnie zmodyfikować i pobawić się nimi. W niektórych punktach zawarte są również zadania, które należy wykonać.

1) Przygotowanie środowiska pracy.

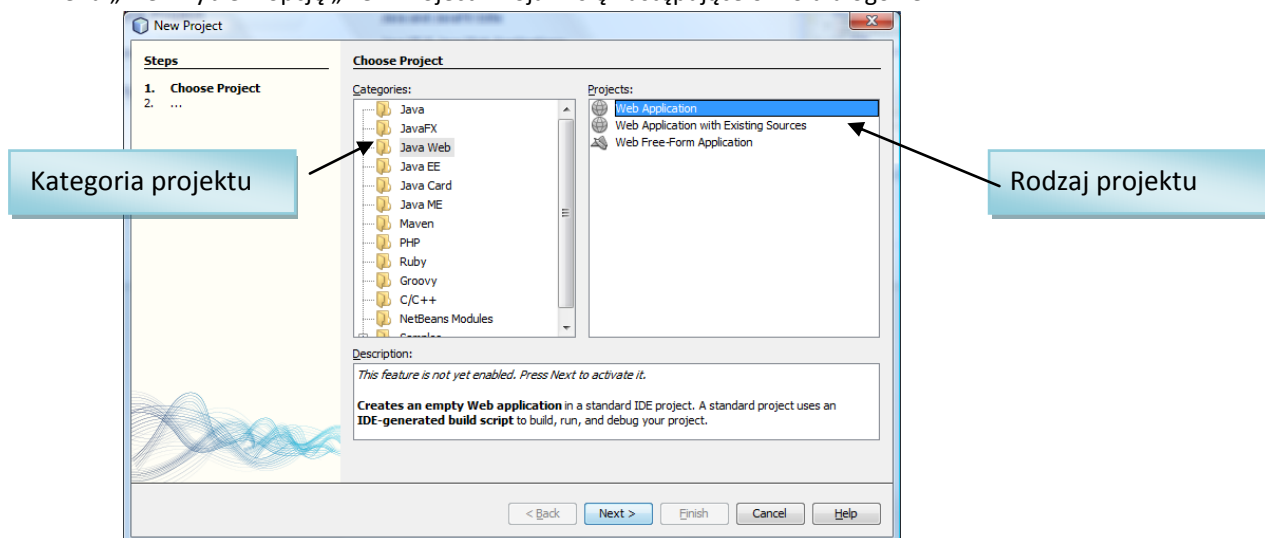
Celem niniejszego zadania jest przygotowanie środowiska pracy. Po ukończeniu zadania, na komputerze uruchomione będzie środowisko NetBeans 6.9.1, oraz utworzony domyślnie generowany projekt prostej aplikacji internetowej.

Sposób przygotowania środowiska jest taki sam jak dla ćwiczenia z Serwletów.

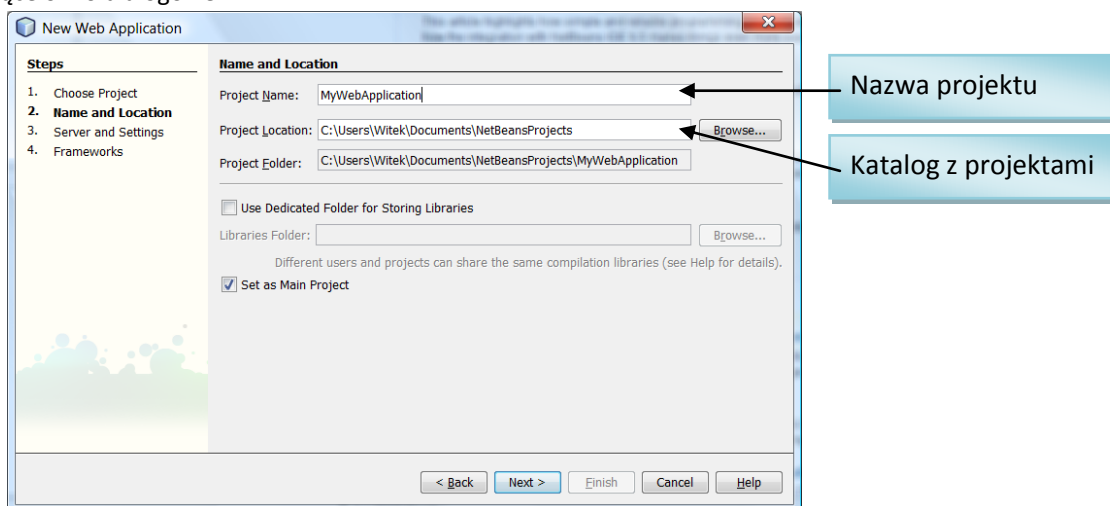
- a) Uruchom środowisko NetBeans 6.9.1. Domyślnie, ikona startująca to środowisko znajduje się w: Start/Programy/NetBeans. Po uruchomieniu programu, na ekranie pojawi się następujące okno:



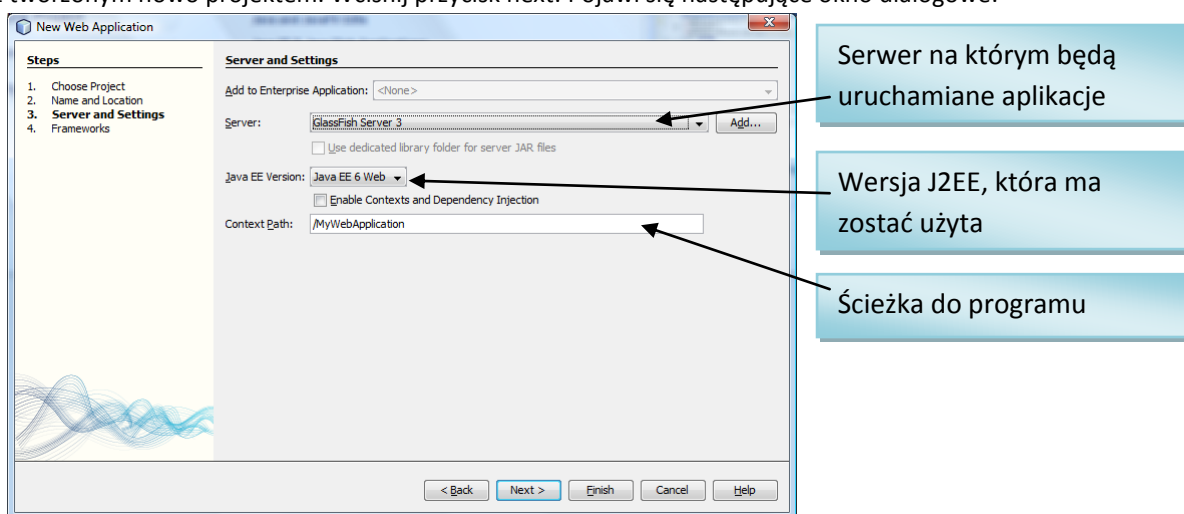
- b) Z menu „File” wybierz opcję „New Project”. Pojawi się następujące okno dialogowe:



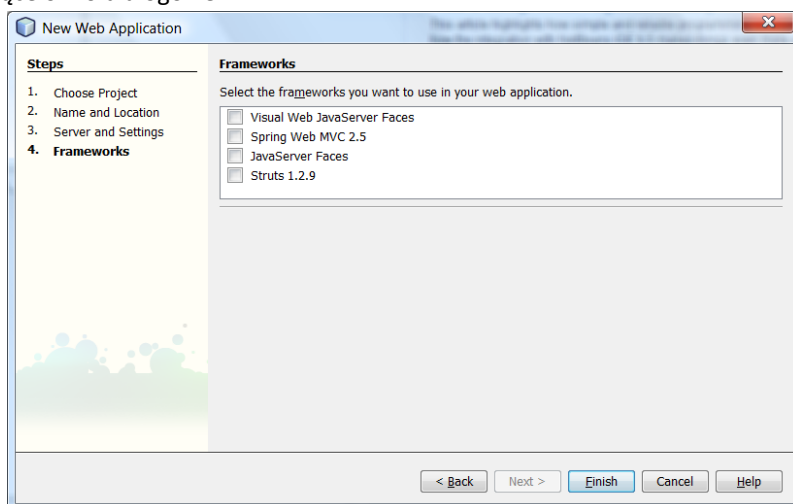
- c) Wybierz kategorię projektu „Web” i rodzaj projektu „Web application”, a następnie wciśnij „Next”. Pojawi się następujące okno dialogowe:



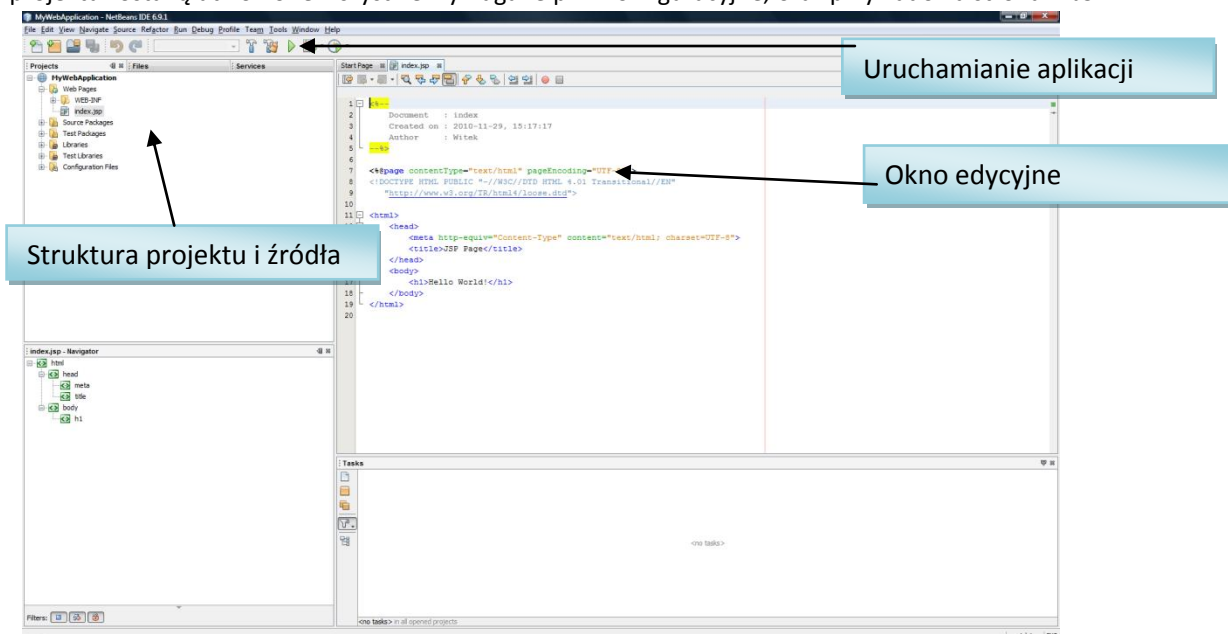
- d) W otrzymanym okienku dialogowym wypełnij pozycje dotyczące nazwy projektu i katalogu, w którym ma znaleźć się katalog z tworzoną nowo projektem. Wciśnij przycisk next. Pojawi się następujące okno dialogowe:



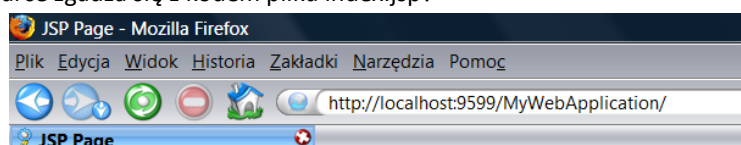
- e) W powyższym okienku dialogowym należy wybrać serwer „GlassFish” oraz wersję J2EE 6. Najciekawszym parametrem jest tutaj parametr „Context Path”. Jest to ścieżka, która umieszczona za adresem domenowym serwera będzie wskazywać na tworzoną aplikację. Uwaga! To wcale nie znaczy, że katalog odpowiadający tej ścieżce w ogóle będzie istnieć. W aplikacjach J2EE ścieżki służą jedynie do określania aplikacji, a nie fizycznego położenia pliku. Przykładowe działanie tego parametru jest następujące. Jeżeli „Context Path” jest równe „/MyWebApplication”, to aplikacja będzie uruchamiana, jeśli jako adres do przeglądarki zostanie wpisane: <http://adres.serwera.com/MyWebApplication>. Zalecane jest pozostawienie wartości domyślnej. Po wypełnieniu parametrów, wciśnij przycisk „Next”. Pojawi się następujące okno dialogowe:



- f) Pozostaw to okno bez zmian (nie wybieraj żadnej opcji). Po wciśnięciu „Finish”, projekt zostanie utworzony i otwarty. W ramach projektu zostaną utworzone wszystkie wymagane pliki konfiguracyjne, oraz przykładowa strona w JSP:



- g) Przeanalizuj zawartość domyślnie wygenerowanego index.jsp, a następnie kliknij na przycisk uruchamiający aplikację. Po chwili powinno pojawić się okno przeglądarki z uruchomioną aplikacją. Czy to co można zaobserwować w przeglądarce zgadza się z kodem pliku index.jsp?

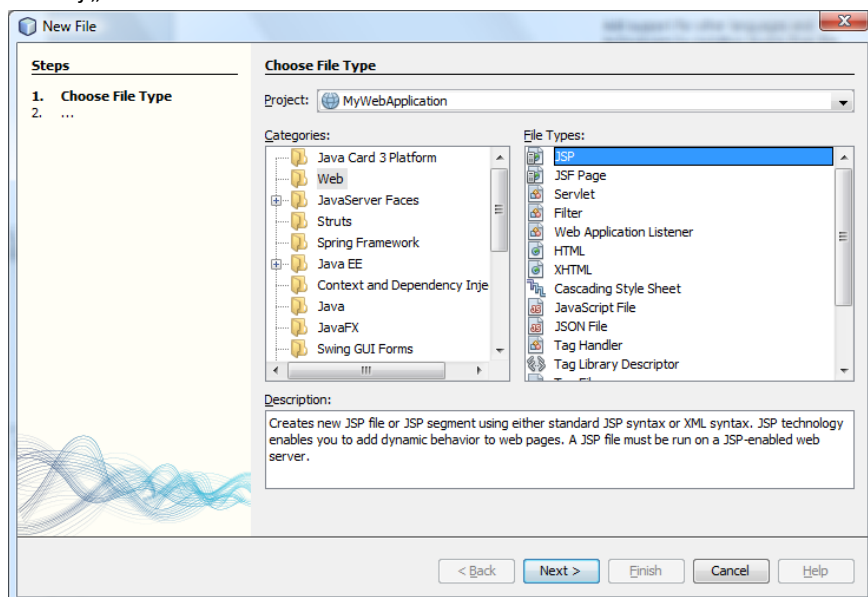


Hello World!

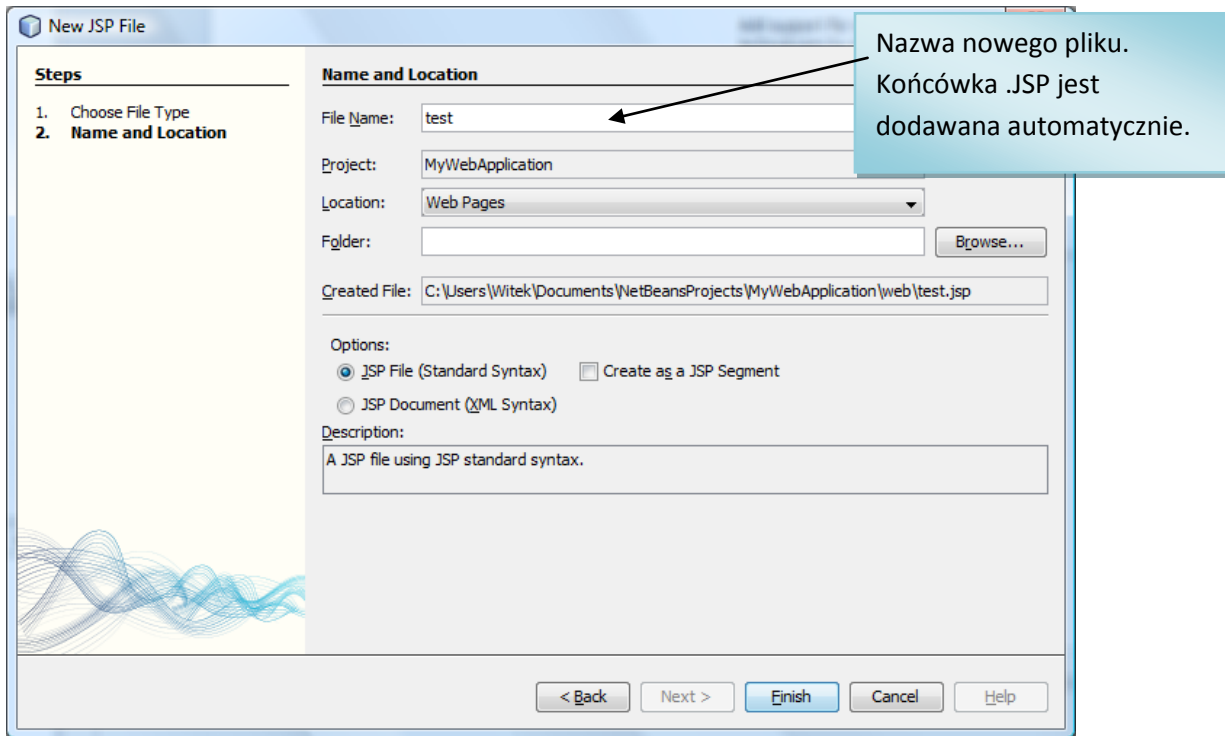
2) Utworzenie prostej strony JSP.

Celem niniejszego zadania jest przedstawienie metody tworzenia nowych stron JSP w ramach aplikacji internetowej i ich uruchamiania. Po utworzeniu projektu powstaje domyślnie jedna strona JSP. Rzadko jednak mamy do czynienia z sytuacją, kiedy cała aplikacja składa się z jednej strony. W związku z tym, poniżej przedstawiono kroki utworzenia nowej, własnej strony.

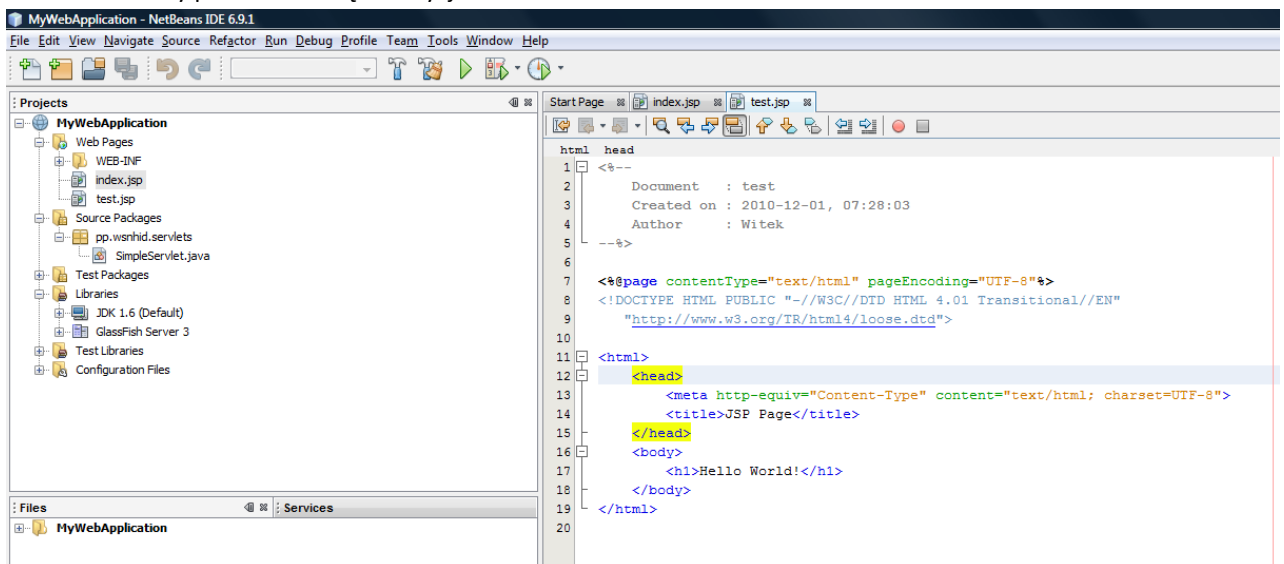
- a) Z menu „File” wybierz opcję „New File”. W okienku, które się pojawi wybierz kategorię Web, i rodzaj JSP, a następnie naciśnij „Next”.



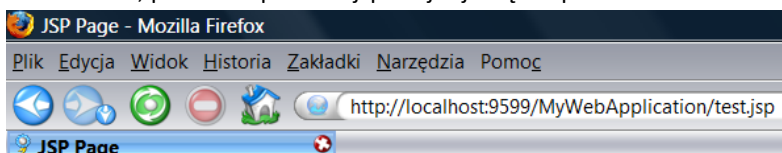
- b) Podaj nazwę nowego pliku we wskazanym na poniższym rysunku miejscu i kliknij „Finish”.



- c) Nowo utworzony plik otwiera się do edycji:



- d) Oglądanie nowo utworzonej strony jest możliwe po przesłaniu strony do serwera (opcja Deploy używana przy serwetach) i wpisaniu odpowiedniego adresu w oknie przeglądarki. Adres ten składa się z następujących członów: `http://adres.com.pl/nazwa_aplikacji/podfoldery_folderu_web/plik.jsp`. Dla nowo utworzonego przykładowego pliku `test.jsp`, może to być następujący adres: <http://localhost:9599/MyWebApplication/test.jsp>. Pominęto tutaj podfoldery folderu `WEB`, ponieważ plik `test.jsp` znajduje się bezpośrednio w folderze `WEB`.



Hello World!

- e) Jak łatwo zauważyć, nowo utworzona strona JSP jest zapisana za pomocą HTML. Jedyną rzeczą nie pochodzącą z HTML dodatkem jest linijka `<%@page...>` określająca kod MIME i kodowanie generowanego przez plik JSP dynamicznego dokumentu. W ramach niniejszych ćwiczeń nie będziemy zmieniać tej linii. Aby zrozumieć działanie stron JSP trzeba wiedzieć, że strony te, przed pierwszym wyświetleniem są transformowane do serwetów. Kolejne ćwiczenia będą omawiać reguły tej transformacji. Przypomnij sobie metodę `processRequest` z

ćwiczeń dotyczących serwletów. Pierwsze dwie reguły można określić na podstawie wygenerowanego przykładowego dokumentu JSP:

- Każdy wiersz kodu HTML jest transformowany do polecenia `out.println(" wiersz HTML ")` wyświetlającego tą linię i wstawiany (w odpowiedniej kolejności) do metody `processRequest`. Przykładowo, wiersz `<title>JSP Page</title>` zostanie przetransformowany do: `out.println("<title>JSP Page</title>");`.
- Polecenie `<%@page contentType="xxx" pageEncoding="yyy"%>` jest transformowane do `response.setContentType("xxx;charset=yyy");`. Przykładowo, polecenie `<%@page contentType="text/html" pageEncoding="UTF-8"%>` zostanie przetransformowane do `response.setContentType("text/html;charset=UTF-8");` i wstawione w odpowiednie miejsce w metodzie `processRequest`.

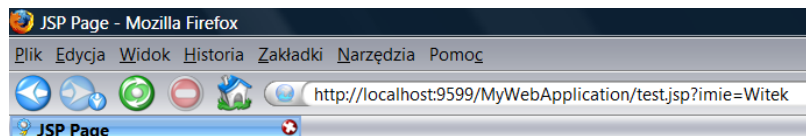
3) Umieszczanie kodu Java w stronach JSP.

a) Kolejną regułą transformacji jest: wszystko co zostanie umieszczone pomiędzy `<%` a `%>` jest przepisywane bez zmian do wynikowego serwletu. Dzięki tej regule można wstawiać kod Java bezpośrednio do strony JSP. Wstawiony kod Java może korzystać m. in z predeklarowanych zmiennych/parametrów formalnych:

- `request` – obiekt określający parametry żądania przeglądarki, dokładnie taka funkcjonalność jak parametru formalnego `request` metody `processRequest` z serwletu.
- `response` - obiekt poprzez który generowany jest wynik strony JSP/serwletu, dokładnie taka funkcjonalność jak parametru formalnego `response` metody `processRequest` z serwletu.
- `session` – zmienna zawierająca wynik działania polecenia `request.getSession(true);`, pozwala na zarządzanie sesją.
- `out` – wynik działania polecenia `response.getWriter();`, służy do wyświetlania wyników działania programu na stronie JSP.

Poniższa przykładowa wstawka kodu może zostać umieszczona gdzieś pomiędzy `<body>` a `</body>` strony JSP i działa ona w ten sposób, że pobiera ona przekazane jako parametr imię, i wyświetla je na stronie:

```
<% String imie=request.getParameter("imie");
out.println("Witaj "+imie);%>
```

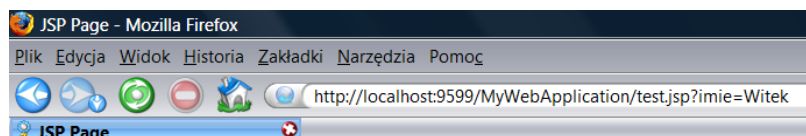


Hello World!

Witaj Witek

b) Powyższy zapis można uprościć stosując kolejną regułę: wszystko umieszczone pomiędzy `<%=` a `%>` jest umieszczane pomiędzy `out.println( a )`. Przykładowo, polecenie: `<%=2*2>` zostanie przetransformowane do `out.println(2*2);`. Uproszczony powyższy zapis można przedstawić następująco:

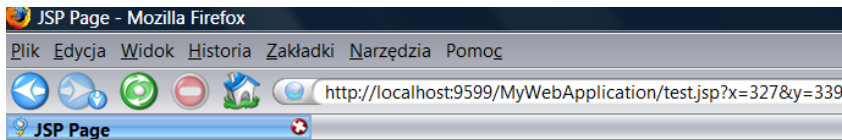
```
<% String imie=request.getParameter("imie");%>
<%= "Witaj "+imie%>
```



Hello World!

Witaj Witek

c) **Zadanie:** Napisz stronę JSP odczytującą parametry o nazwach „x” i „y” i wyświetlającą ich sumę. Nie używaj nigdzie `out.println()`. Przypomnij sobie jak wykonałeś analogiczne zadanie podczas ćwiczeń z serwletów.



Hello World!

327+339=666

- d) Czasami zachodzi konieczność zaimportowania jakiejś klasy, bądź zbioru klas z innego pakietu. W języku Java robi się to za pomocą polecenia `import`. Ponieważ strona JSP jest transformowana do kodu Java, to należy użyć specjalnego polecenia JSP w celu zaimportowania jakiejś klasy. Poleceniem tym jest: `<%@ page import="xxx" %>`. Jest ono transformowane do linijki `import xxx;` i umieszczane na początku serwletu. Przykładowo, linijka importująca klasę `Date` z pakietu `java.util`, wygląda następująco:

```
<%@ page import="java.util.Vector" %>
```

Powyższa linijka jest transformowana do polecenia:

```
import java.util.Vector;
```

Od tego momentu można tą klasę wykorzystywać we wstawkach kodu JSP. Poniższy przykład demonstruje zastosowanie importowania klas: importuje klasę `Date`, tworzy jej obiekt i wyświetla jego reprezentację łańcuchową w oknie przeglądarki.

```
<%@ page import="java.util.Date" %>
```

```
<% Date data=new Date(); //domyslnie obiekt reprezentuje aktualną datę%>
```

```
<%=data%>
```

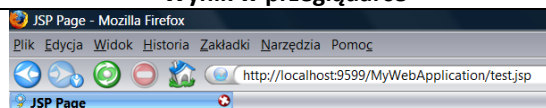
4) Mieszanie kodu HTML i kodu Java.

Możliwe jest mieszanie kodu HTML i kodu Java z pętlami i instrukcjami warunkowymi. Poniżej kilka przykładów:

- a) **Pętla.** Poniższa pętla wyświetla liczby od 1 do 10. Każda z tych liczb stanowi osobną pozycję nienumerowanej listy.

Kod JSP	Kod JSP po „transformacji” do Java	Wynik pętli w postaci HTML
<pre> <% for (int i=1;i<=10;i++) {%> <%=i%> <%}%> </pre>	<pre>out.println(""); for (int i=1;i<=10;i++) { out.println(""); out.println(i); out.println(""); } out.println("");</pre>	<pre> 1 2 3 4 5 6 7 8 9 10 </pre>

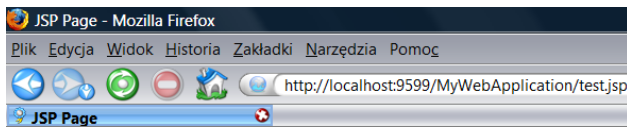
Wynik w przeglądarce



Hello World!

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

- b) **Zadanie:** Napisz stronę JSP dynamicznie generującą tabelkę z tabliczką mnożenia. Nie używaj `out.println`. Przykładowy wynik:



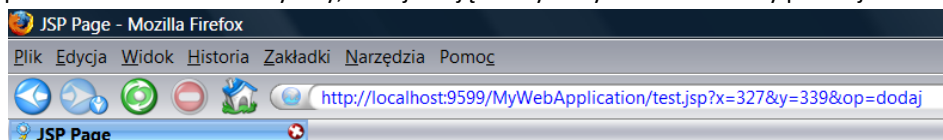
Hello World!

1 2 3 4 5 6 7 8 9 10
 2 4 6 8 10 12 14 16 18 20
 3 6 9 12 15 18 21 24 27 30
 4 8 12 16 20 24 28 32 36 40
 5 10 15 20 25 30 35 40 45 50
 6 12 18 24 30 36 42 48 54 60
 7 14 21 28 35 42 49 56 63 70
 8 16 24 32 40 48 56 64 72 80
 9 18 27 36 45 54 63 72 81 90
 10 20 30 40 50 60 70 80 90 100

- c) **Instrukcje warunkowe:** Poniższy przykład odczytuje wartość parametru „plec” i w zależności od tego czy jest on równy M, czy K, czy też coś innego, wyświetla odpowiedni komunikat.

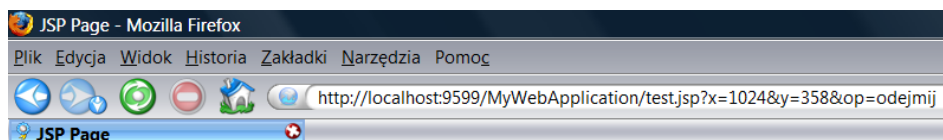
Kod JSP	Kod JSP po „transformacji” do Java
<pre><% String plec=request.getParameter("plec"); if (plec==null) plec=""; if (plec.equals("M")) {%> <h1>Witam Pana!</h1> <% } else if (plec.equals("K")) {%> <h1>Witam Panią!</h1> <% } else {%> <h1> Witam Niewiadomoco!</h1> <%}%></pre>	<pre>String plec=request.getParameter("plec"); if (plec==null) plec=""; if (plec.equals("M")) { out.println("<h1>Witam Pana!</h1>"); } else if (plec.equals("K")) { out.println("<h1>Witam Panią!</h1>"); } else { out.println("<h1>Witam Niewiadomoco!</h1>"); }</pre>
Wynik w przeglądarce	

- d) **Zadanie:** Napisz stronę JSP, która przyjmuje trzy parametry: „x”, „y” i „op”. W zależności od parametru op strona powinna albo dodać liczby x i y, albo je odjąć. Przykłady działania strony poniżej:



Hello World!

327+339=666



Hello World!

1024-358=666

5) Przekazywanie generowania wyniku z serwletu do JSP

- a) Utwórz nowy serwlet. Nowy serwlet powinien odczytywać parametry przekazane przez użytkownika x, y i z (trzy liczby). Serwlet powinien wykrywać sytuację, w której przekazane zostaną tylko x i y (z równe jest null) albo wszystkie 3 wartości. Odczytane liczby powinny być zapisane jako atrybuty żądania przeglądarki (albo 2 albo 3 liczby). Zakładając, że parametrem metody processRequest, który zawiera parametry żądania przeglądarki jest request, ustawienie parametrów możliwe jest za pomocą metody setAttribute:

```
request.setAttribute("valueX", zmiennaX);
```

W powyższym przykładzie „valueX” to etykieta nadawana przez użytkownika obiektowi zapisywanemu jako atrybut, a zmiennaX to zmienna z tym obiektem. Po zapisaniu liczb przekazanych jako parametry użytkownika jako atrybuty, w zależności od tego ile liczb przekazano przekazuje generowanie wyniku do jednej z dwóch stron JSP opisanych w kolejnym podpunkcie. **Uwaga! Przekonwertuj przekazane liczby do typu Integer zanim zapiszesz je jako atrybuty wywołania!** Przekazanie generowania wyniku wykonuje się następująco:

```
ServletContext ctx = this.getServletContext();
```

```
RequestDispatcher dispatcher = ctx.getRequestDispatcher("/simplePage.jsp");
```

```
dispatcher.forward(request, response);
```

w powyższym przykładzie przekierowanie nastąpi do strony simplePage.jsp.

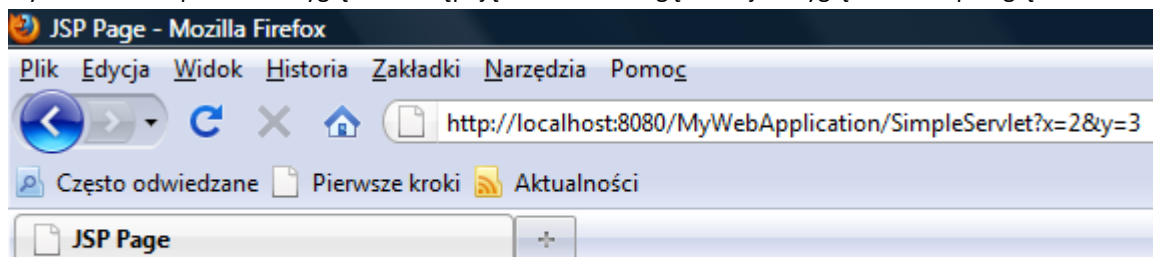
- b) Przygotuj dwie strony JSP. Pierwsza z nich powinna pobierać z atrybutu zapisanego w serwlecie dwie liczby, dodawać je i zapisywać wynik. Druga z nich powinna robić to samo z trzema liczbami. Pobranie wartości atrybutu (typu Integer) z poziomu JSP można wykonać następująco:

```
<%
```

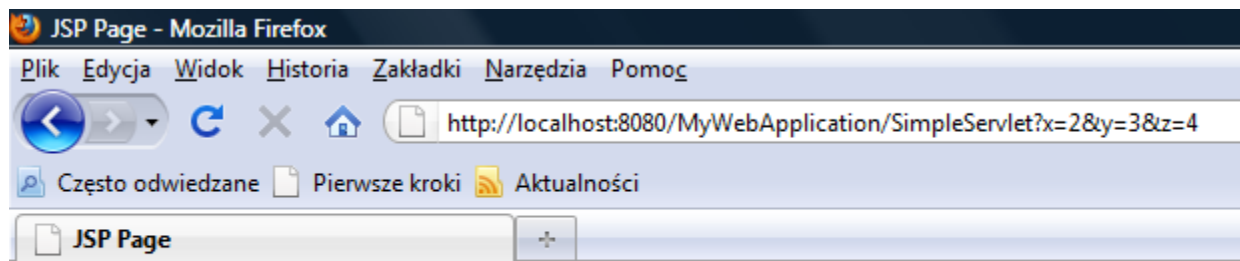
```
Integer x=(Integer)request.getAttribute("valueX");
```

```
%>
```

Wynik działania powinien wyglądać następująco. Zwróć uwagę na to jak wygląda URL w przeglądarce.



2+3=5



2+3+4=9

- c) Dopisz za wywołaniem metody `dispatcher.forward` w serwlecie polecenie `out.println("<br>TEST</br>");`. Ponownie uruchom serwlet z odpowiednimi parametrami. Czy słowo test pojawiło się w wyniku? Zamień teraz wszystkie wywołania `dispatcher.forward` na `dispatcher.include`. Czy teraz słowo test pojawiło się w wyniku? Obejrzyj źródło wygenerowanej strony. Zastanów się nad różnicami w działaniu `dispatcher.forward` i `dispatcher.include`. Jak powinny wyglądać strony JSP używane przy `dispatcher.include`?

6) Wyrażenia EL.

Zmodyfikuj strony JSP z poprzedniego podpunktu tak, aby do wypisania wartości liczbowych użyć wyrażeń JSP EL. W najprostszej sytuacji wypisanie wartości liczbowej zapisanej w atrybucie `abc` odbywa się za pomocą `${abc}`. Dodanie wartości atrybutów `x` i `y`, i wypisanie wyniku to `${x+y}`. Przykładowo, jeżeli w serwlecie wykonano polecenie:

```
request.setAttribute("wartosc", new Integer(123));
```

to w stronie JSP do której przekierowano request, użycie

```
${wartosc}
```

Spowoduje wypisanie 123. W analogiczny sposób można odwoływać się do parametrów żądania i sesji. Aby odwołać się do wartości parametru o nazwie `par1`, można napisać:

```
${param.par1}
```

Aby odwołać się do wartości zapisanej w sesji pod etykietą `ses1`, można napisać:

```
${sessionScope.ses1}
```

7) JSP Beans

- a) Utwórz klasę `pp.wsnhid.servlets.MyCounter`:

```
package pp.wsnhid.servlets;

public class MyCounterBean {
    private Integer counter;

    public MyCounterBean() {
        counter=0;
    }

    public void setCounter(Integer counter) {
        this.counter = counter;
    }

    public Integer getCounter() {
        return counter;
    }

    public void increment() {
        setCounter(getCounter()+1);
    }
}
```

- b) Utwórz nową stronę JSP, w której umieść Tag tworzący nowy JSP Bean o zasięgu strony:

```
<jsp:useBean id="counter" class="pp.wsnhid.servlets.MyCounterBean"
    scope = "page"/>
```

JSP Bean to po prostu zmienna o nazwie podanej w `id` i klasie podanej w `class`. Są jednak dwie dodatkowe zalety używania JSP beans. Po pierwsze można określić zasięg ważności tej zmiennej (pokażą to kolejne podpunkty), a po drugie zmienna ta jest dostępna dla wyrażeń EL (co można zobaczyć w następnym podpunkcie).

- c) Dopisz do strony JSP kod wyświetlający wartość licznika oraz zwiększający jego wartość:

Wartość licznika to `${counter.value}`

`<% counter.increment(); %>`

Zwróć uwagę na to, że wyświetlenie wartości licznika odbywa się analogicznie jak wyświetlenie wartości atrybutu.

Uwaga! Odwołanie `counter.value` w wyrażeniu JSP EL, jest tłumaczone na odpowiednie wywołanie metody `getValue` bądź `setValue` w zależności od kontekstu. Z tego powodu należy zwracać uwagę na nazwy metod przy tworzeniu klasy `beana`! Ponieważ `Beana` można traktować jako zmienną, to zwiększenie naszego licznika jest możliwe po prostu dzięki wywołaniu metody `increment`.

- d) Uruchom stronę JSP w przeglądarce. Odśwież stronę kilkukrotnie. Czy licznik się zwiększa?
- e) Zmień scope `Beana` na "session". Powtórz eksperyment. Czy teraz licznik się zwiększa?
- f) Uruchom drugą przeglądarkę np. jeżeli pracowałeś/aś na na Firefoxi, to uruchom również Internet Explorera i otwórz w niej tą samą stronę JSP (chodzi o to, żeby strona była otwarta w dwóch niezależnych sesjach). Powtórz eksperyment. Czy mamy jeden licznik czy dwa niezależne liczniki?
- g) Zmien scope `Beana` na "application". Odśwież licznik w obydwu przeglądarkach. Czy są dwa niezależne liczniki, czy jeden?

8) Zadania

- a) Napisz dwie strony JSP wypisujące liczby od 1 do 10. Pierwsza strona powinna wypisywać te liczby w różnych kolorach, a druga tylko na czarno. Napisz serwlet, który na podstawie parametru `color` przekazanego metodą `Get` (wartości `true` albo `false`) uruchamia odpowiednią stronę JSP.
- b) Napisz stronę JSP, która losuje i zapisuje najpierw do tablicy 75 liczb. Następnie strona JSP powinna wyświetlić te liczby w tabelce w trzech kolumnach. W ostatnim wierszu tabelki powinna się znaleźć suma wszystkich wartości z całej kolumny.
- c) Napisz stronę JSP, która pozwala podać kolor czcionki w dokumencie jako parametr. Podany kolor powinien być zapisywany w sesji. Jeżeli parametr jest nie podany i w sesji nie został zapisany, to domyślny kolor jest czarny. Jeżeli koloru nie podano, ale jest on zapisany w sesji, to ten kolor powinien być użyty.