

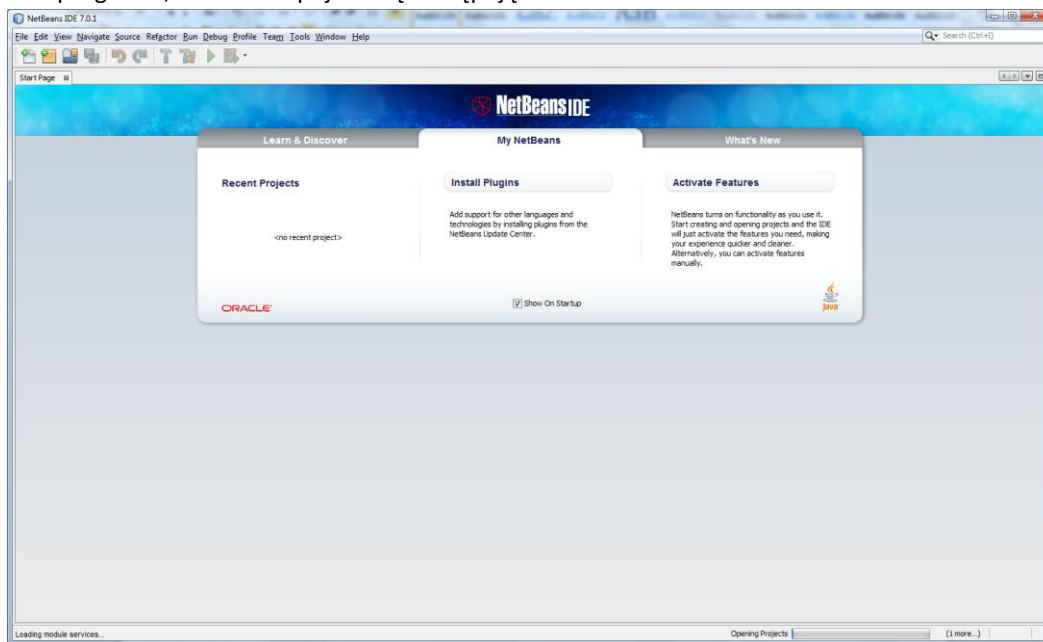
Serwlety.

Celem niniejszego ćwiczenia jest poznanie podstaw technologii tworzenia aplikacji internetowych J2EE, poprzez zapoznanie się z metodami tworzenia tzw. serwletów. W ramach niniejszego ćwiczenia wykorzystane zostanie środowisko programistyczne NetBeans 7.0.1.

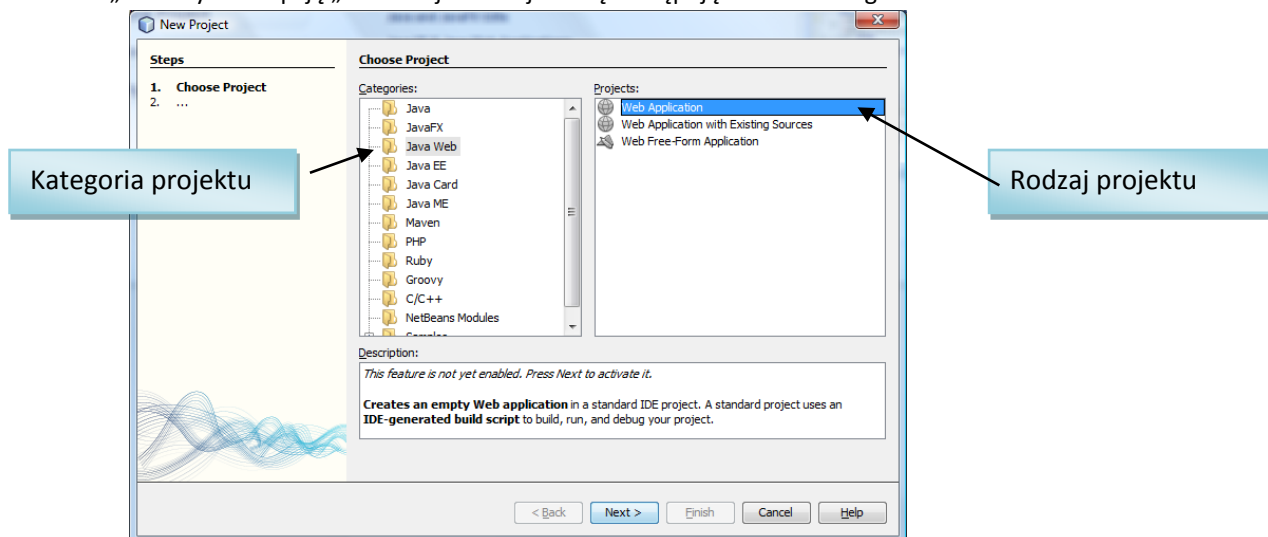
1) Przygotowanie środowiska pracy.

Celem niniejszego zadania jest przygotowanie środowiska pracy. Po ukończeniu zadania, na komputerze uruchomione będzie środowisko NetBeans 7.0.1, oraz utworzony domyślnie generowany projekt prostej aplikacji internetowej.

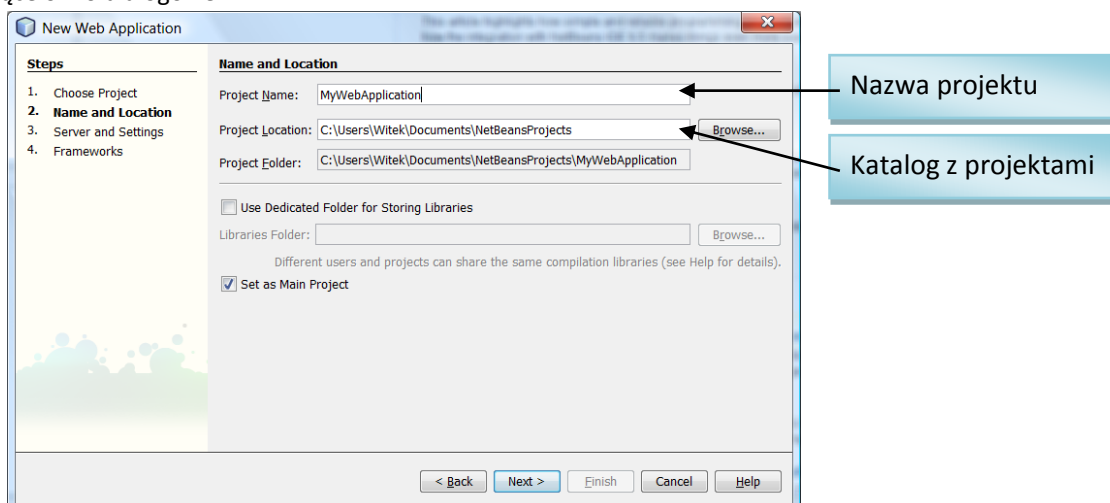
- a) Uruchom środowisko NetBeans 7.0.1. Domyślnie, ikona startująca to środowisko znajduje się w: Start/Programy/NetBeans. Po uruchomieniu programu, na ekranie pojawi się następujące okno:



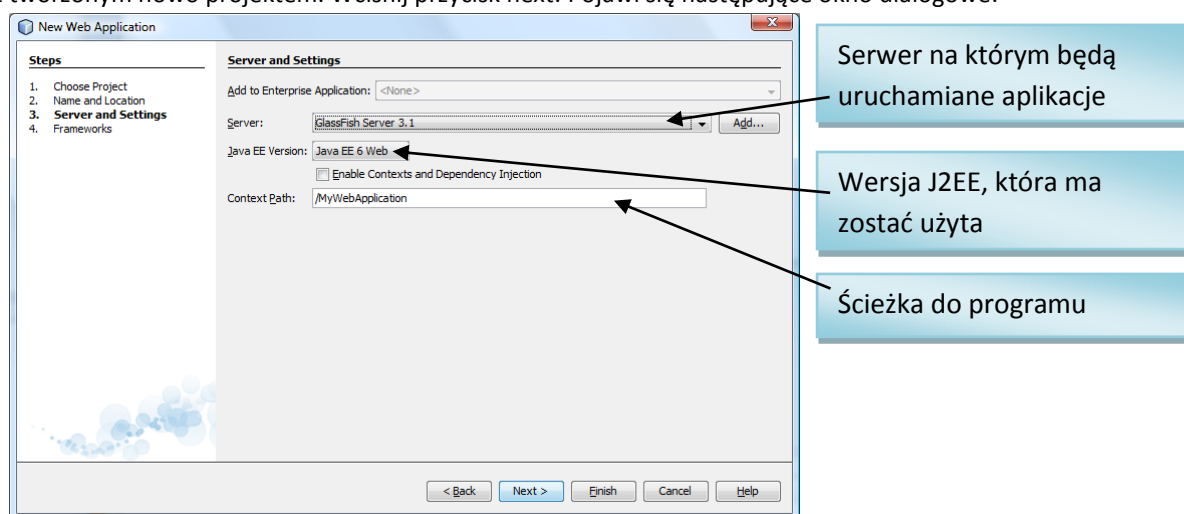
- b) Z menu „File” wybierz opcję „New Project”. Pojawi się następujące okno dialogowe:



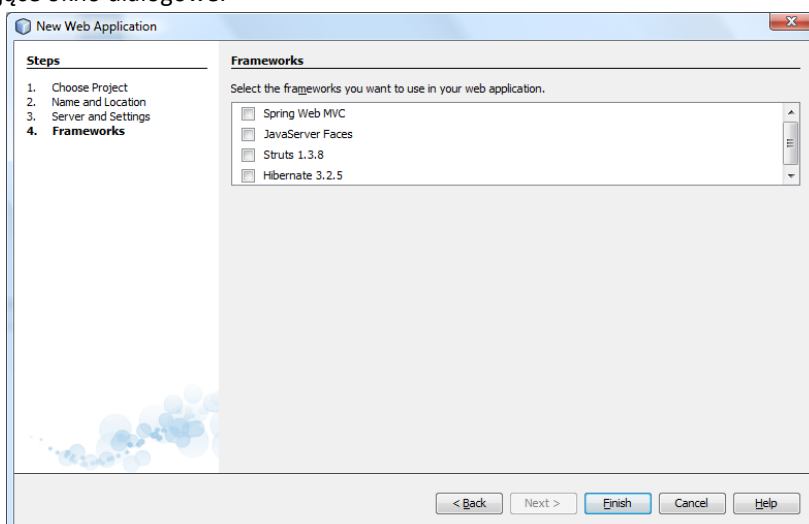
- c) Wybierz kategorię projektu „Web” i rodzaj projektu „Web application”, a następnie wciśnij „Next”. Pojawi się następujące okno dialogowe:



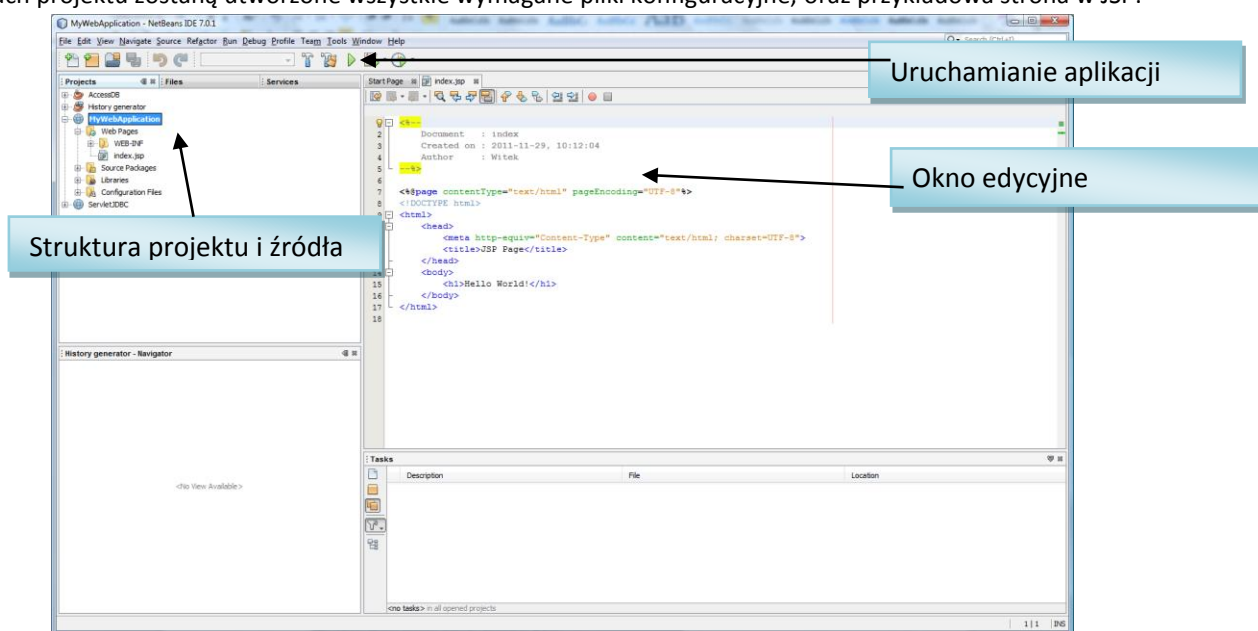
- d) W otrzymanym okienku dialogowym wypełnij pozycje dotyczące nazwy projektu i katalogu, w którym ma znaleźć się katalog z tworzoną nowo projektem. Wciśnij przycisk next. Pojawi się następujące okno dialogowe:



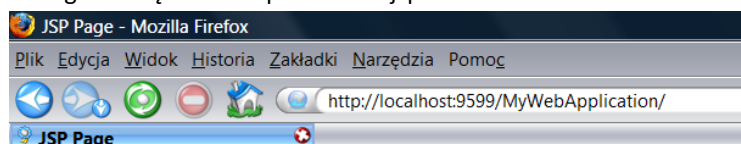
- e) W powyższym okienku dialogowym należy wybrać serwer „GlassFish” oraz wersję J2EE 6. Najciekawszym parametrem jest tutaj parametr „Context Path”. Jest to ścieżka, która umieszczona za adresem domenowym serwera będzie wskazywać na tworzoną aplikację. Uwaga! To wcale nie znaczy, że katalog odpowiadający 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?

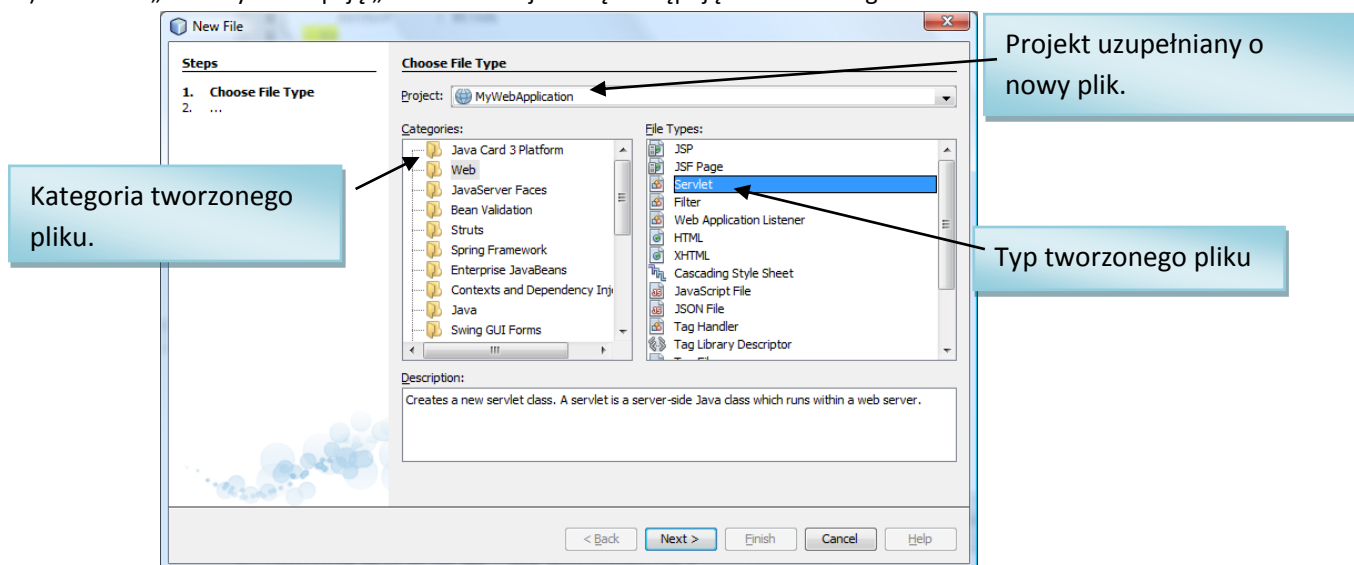


Hello World!

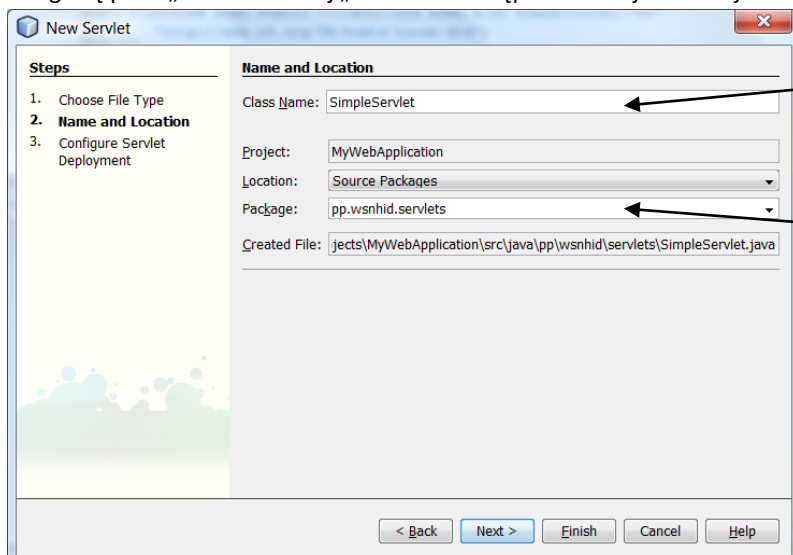
2) Utworzenie prostego serwletu.

Celem niniejszego zadania jest przedstawienie metody tworzenia nowych serwletów w ramach aplikacji internetowej i ich uruchamiania.

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



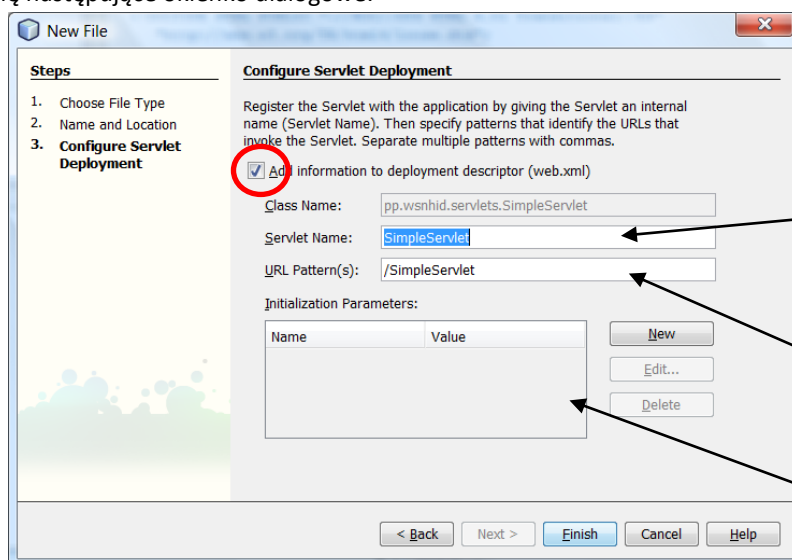
- b) W otworzonym okienku dialogowym upewnij się, że nowy plik jest dodawany do projektu nad którym pracujesz. Wybierz kategorię pliku „Web” a rodzaj „Servlet”. Następnie wciśnij next. Pojawi się następujące okienko dialogowe:



Nazwa klasy implementującej serwet.

Pakiet w którym należy umieścić klasę.

- c) W otrzymanym okienku dialogowym należy wpisać nazwę klasy implementującej nasz serwet (np. SimpleServlet) oraz pakiet, w którym ta klasa powinna się znaleźć (np. pp.wsnhid.servlets). Po uzupełnieniu ww. parametrów, naciśnij next. Pojawi się następujące okienko dialogowe:

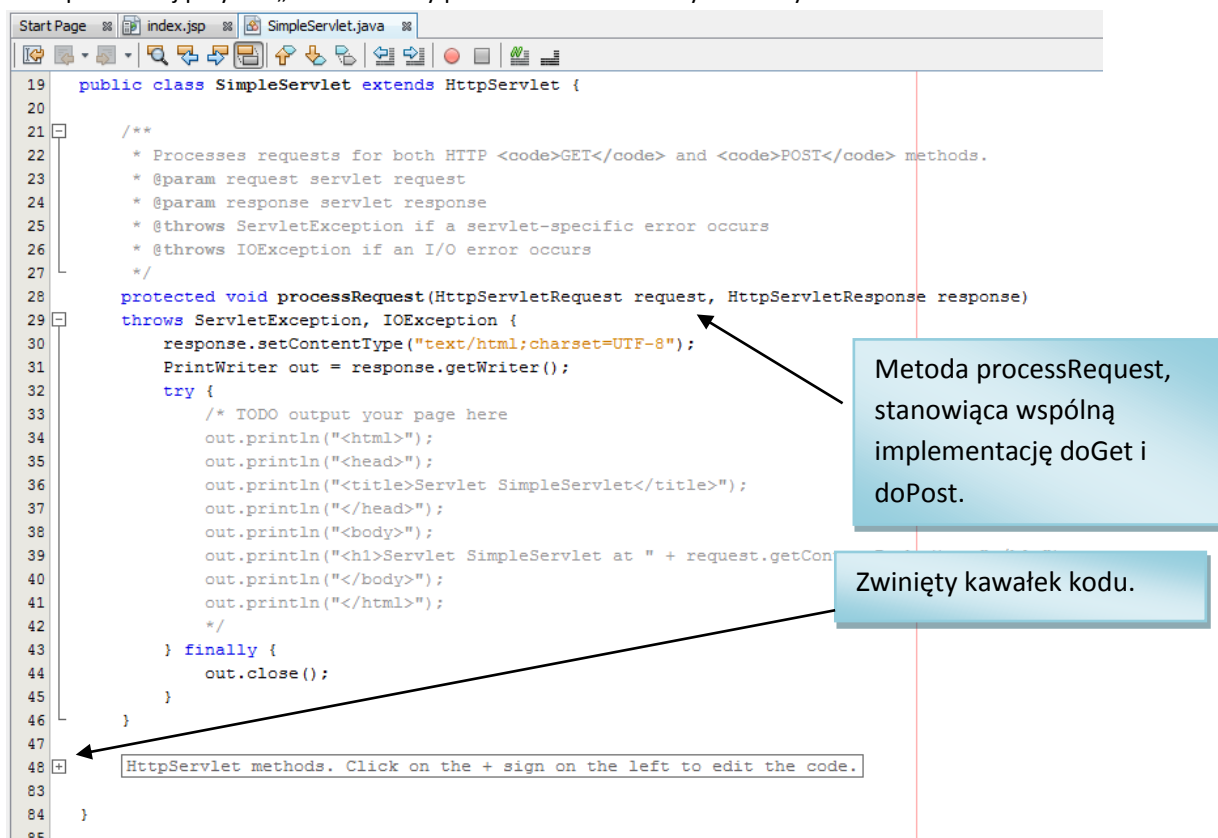


Wewnętrzna nazwa serwletu.

Wzorzec URL wywołania serwletu.

Parametry inicjalizacyjne serwletu.

- d) W otrzymanym okienku pozostaw domyślną nazwę wewnętrzną serwletu, a następnie podaj wzorec URL. Wzorec URL określa, jakie wyrażenie w ścieżce, podane za adresem aplikacji (patrz punkt 1e), uruchamia tworzony serwlet. Wyrażenie musi zaczynać się od „/” po którym może zostać podana dowolna ścieżka. Można również wykorzystywać znak „*” do oznaczenia dowolnego ciągu znaków. Sugerowane jest pozostawienie domyślnej wartości. Po wypełnieniu odpowiednich pól wciśnij przycisk „Finish”. Nowy plik zostanie utworzony i otwarty:



- e) Rozwiń zwinięty kawałek kodu wskazany na powyższym rysunku, klikając na znak „+”. Odsłonięte zostaną metody doGet i doPost:

```
// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
/**
 * Handles the HTTP <code>GET</code> method.
 * @param request servlet request
 * @param response servlet response
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Handles the HTTP <code>POST</code> method.
 * @param request servlet request
 * @param response servlet response
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}
```

- f) Jak łatwo zauważyć na powyższym rysunku, metody doGet i doPost odwołują się do metody processRequest. Można zatem zauważyć, że obsługa żądań GET i POST jest identyczna. Odkomentuj zawartość metody processRequest i usuń linijkę „TODO: output your page here”. Przeanalizuj odkomentowany kod. Zastanów się, co powinno się pojawić w przeglądarce odwołującej się do serwletu.

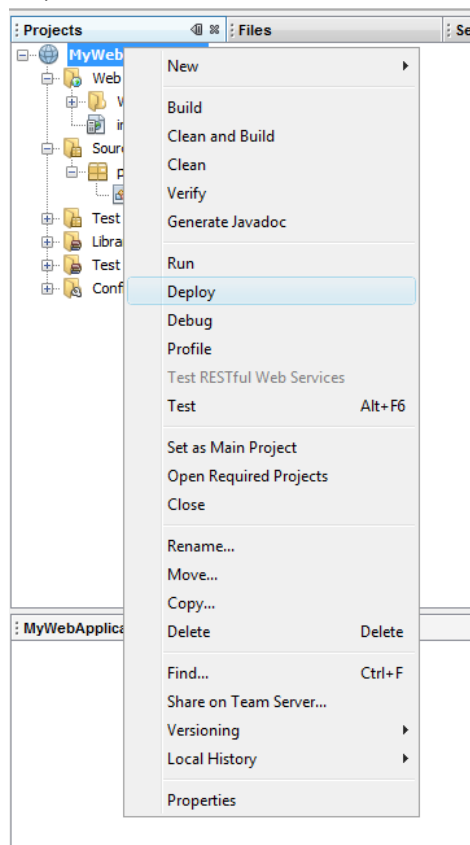
```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {

        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet SimpleServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet SimpleServlet at " + request.getContextPath () + "</h1>");
        out.println("</body>");
        out.println("</html>");

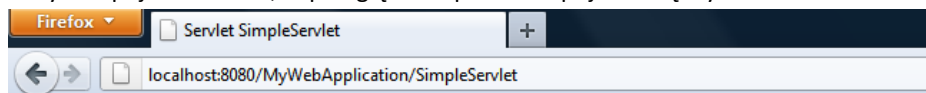
    } finally {
        out.close();
    }
}
```

g) Aby uruchomić serwlet wykonaj następujące kroki:

- Kliknij prawym klawiszem na nazwę projektu w oknie po lewej stronie ekranu. Pojawi się menu kontekstowe (patrz poniższy rysunek):



- Z menu kontekstowego wybierz „Deploy”.
- W przeglądarce wybierz adres taki sam, jaki pojawił się w momencie uruchamiania aplikacji (patrz punkt 1g), ale dodaj na jego końcu „/” i wyrażenie zgodne ze wzorce URL podanym w punkcie 2c. Dla przykładowych nazw stosowanych w niniejszym ćwiczeniu byłoby to: `http://localhost:8080/MyWebApplication/SimpleServlet`. Jeżeli wszystko pójdzie dobrze, w przeglądarce powinien pojawić się wynik działania serwletu:



Servlet SimpleServlet at /MyWebApplication

Jeżeli serwlet zostanie później zmodyfikowany, wystarczy wykonać jeszcze raz „Deploy”, oraz odświeżyć zawartość okna przeglądarki.

3) Odczytanie parametrów żądania otrzymanego z przeglądarki.

a) Zmodyfikuj metodę „processRequest” wygenerowaną automatycznie razem z serwletem w poprzednim zadaniu, tak, aby wyświetlała w oknie przeglądarki wyniki działania następujących metod interfejsu `HttpServletRequest` (należy je wywoływać poprzez parametr „request” metody `processRequest`, który jest tego typu):

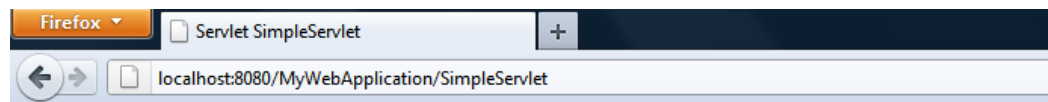
- `getMethod()` – bezparametrowa metoda zwracająca typ żądania (GET, POST, HEAD, itp.)
- `getRemoteAddr()` – bezparametrowa metoda zwracająca adres IP przeglądarki.

- `getServerName()` – bezparametrowa metoda zwracająca adres domenowy serwera.
- `getHeader(n)` – metoda o jednym parametrze typu łańcuchowego, która zwraca wartość jednego z pól żądania protokołu HTTP. Parametr zawiera nazwę pola, którego wartość chcemy odczytać. Odczytaj wartość pól: „Accept”, „Accept-Language”, „Accept-Encoding” i „User-Agent”.

Przykładowe linijki kodu, wykorzystujące metody wymienione powyżej:

```
out.println("wynik getRemoteAddr: "+request.getRemoteAddr()+ "<br>");
out.println("wynik getHeader(\"Accept\") : "+request.getHeader("Accept")+ "<br>");
```

Przykładowy wynik działania poprawnie wykonanego ćwiczenia przedstawiono poniżej:



Servlet SimpleServlet at /MyWebApplication

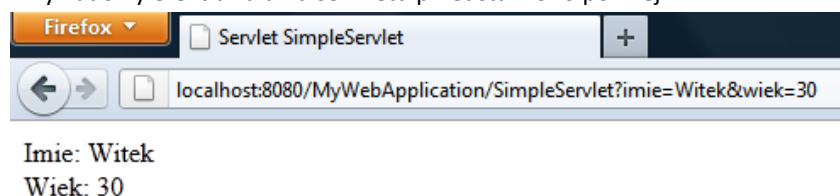
```
wynik getMethod(): GET
wynik getRemoteAddr(): 127.0.0.1
wynik getServerName(): localhost
wynik getHeader("Accept"): text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
wynik getHeader("Accept-Language"): pl,en-us;q=0.7,en;q=0.3
wynik getHeader("Accept-Encoding"): gzip, deflate
wynik getHeader("User-Agent"): Mozilla/5.0 (Windows NT 6.0; rv:7.0.1) Gecko/20100101 Firefox/7.0.1
```

- b) Zmodyfikuj adres URL wpisany do przeglądarki w ten sposób, aby podawał dwa parametry oraz ich wartości: imię i wiek. Dla przykładowego adresu, używanego w niniejszym ćwiczeniu wyglądałoby to następująco: `http://localhost:8080/MyWebApplication/SimpleServlet?imie=Witold&wiek=30`. Następnie, wzorując się na poprzednim ćwiczeniu, wykorzystaj kolejną metodę klasy `HttpServletRequest`, o nazwie `getParameter`, aby odczytać i wyświetlić wartości parametrów podanych w URL. Przykładowe zastosowanie `getParameter` poniżej:

```
out.println("wynik getParameter(\"nazwisko\") : "+request.getParameter("nazwisko")+ "<br>");
```

Uwaga! W zadaniu chodzi o parametry **imie** i **wiek**, podczas gdy powyższy przykład pokazuje odczytanie parametru **nazwisko**.

Przykładowy efekt działania serwletu przedstawiono poniżej:

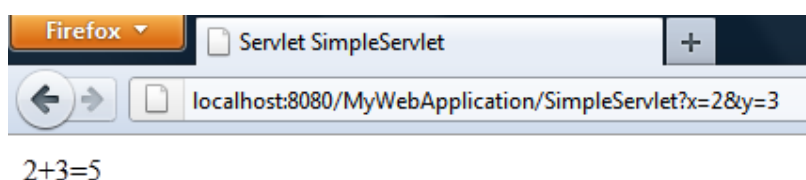


- c) Zmodyfikuj serwlet tak, by odczytywał parametry o nazwach `x` i `y` (usuń wsparcie dla parametrów `imie` i `wiek` z poprzedniego ćwiczenia). Parametry te powinny zawierać liczby. Twój program powinien odczytywać te parametry i, ponieważ wartość każdego parametru jest zwracana jako łańcuch, konwertować je do liczb. Przykładowa konwersja łańcucha do liczby (całkowitej) przedstawiona została poniżej:

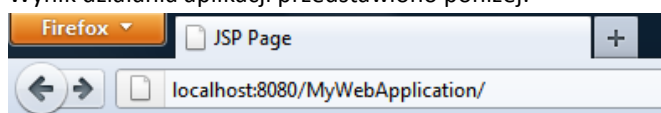
```
int liczba;
liczba=Integer.parseInt("100"); //do zmiennej liczba zapisywana jest wartość 100
```

Po odczytaniu i przekonwertowaniu, liczby przekazane przez parametry `x` i `y` powinny zostać dodane do siebie, a wynik dodawania wyświetlony przez serwlet w oknie przeglądarki.

Przykładowy efekt działania serwletu przedstawiono poniżej:

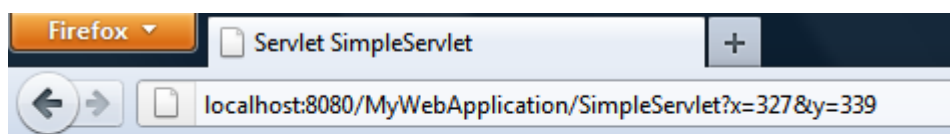


- d) Zmodyfikuj plik index.jsp, wygenerowany automatycznie w momencie tworzenia aplikacji internetowej (patrz zadanie 1) tak, aby wyświetlał formularz wczytujący wartości parametrów x i y, i przekazywał je do serwletu metodą GET. Uwaga! Tutaj tak naprawdę nie korzystamy z technologii JSP, tylko wykorzystujemy plik index.jsp jako zwykłą stronę w HTML. Wynik działania aplikacji przedstawiono poniżej:



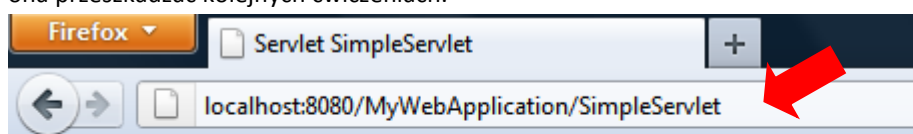
Hello World!

x:
y:



327+339=666

- e) Zmodyfikuj formularz tak, aby przekazywał parametry metodą POST (nie trzeba w ogóle modyfikować kodu serwletu). Zaobserwuj różnicę w działaniu aplikacji. Po zakończeniu ćwiczenia zakomentuj obecną funkcjonalność serwletu. Może ona przeszkadzać w kolejnych ćwiczeniach.



327+339=666

4) Zasięg widoczności zmiennych.

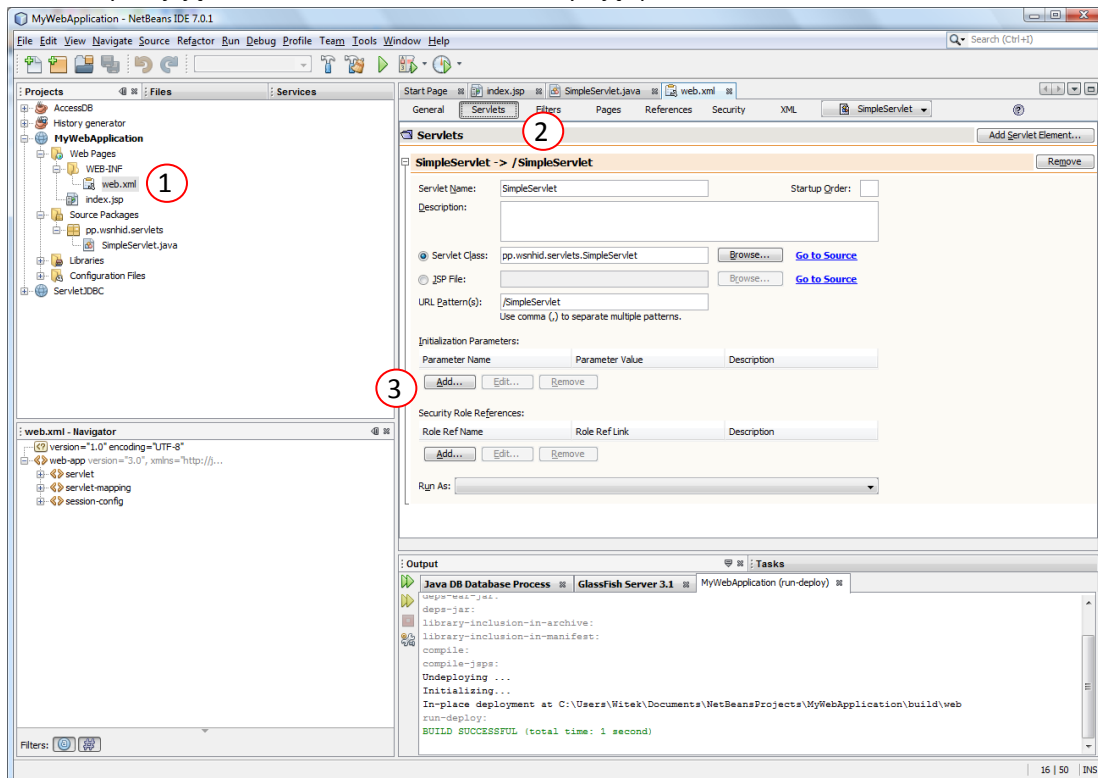
- a) Wpisz następujący kawałek kodu do metody processRequest:

```
int i=0;  
i=i+1;  
out.println("Licznik: "+i);
```

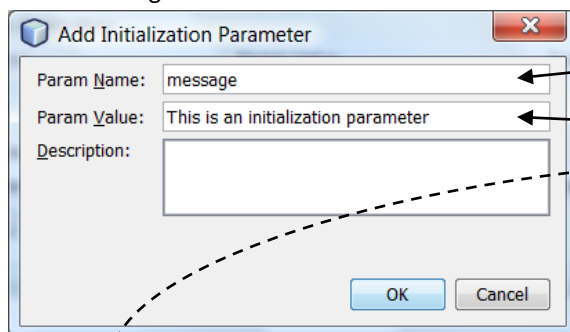
- b) Uruchom serwlet i odśwież okno przeglądarki kilka razy (czy licznik się zwiększa czy nie?)
c) Usuń deklarację zmiennej „i”, a następnie zadeklaruj ją jako pole klasy implementującej serwlet.
d) Uruchom serwlet i odśwież przeglądarkę kilka razy (czy licznik się zwiększa czy nie?)
e) Licznik się nie zwiększał za pierwszym podejściem ponieważ za każdym razem, w momencie uruchomienia processRequest, zmienna była alokowana i inicjowana na nowo. Za drugim razem zmienna została zaalokowana tylko raz, w momencie uruchomienia serwletu. Co więcej, zmienna utworzona w drugim przypadku jest współdzielona przez wszystkie wątki realizujące żądania zgłaszane do serwletu. Wynika z tego, że jeżeli wielu użytkowników równocześnie odświeża stronę wygenerowaną przez serwlet, to każdy z nich będzie widział licznik zwiększony przez innych. Należy o tym fakcie pamiętać, bo konieczna jest synchronizacja do zmiennych współdzielonych przez wiele wątków.

5) Odczytanie parametrów inicjujących serwletu.

a) Parametry inicjujące dla serwletu można ustawić edytując plik web.xml.



b) Plik web.xml można odnaleźć w katalogu „Web pages/WEB-INF”(1). Po odnalezieniu pliku wykonaj na nim double click aby go otworzyć, a po otwarciu, odnajdź w oknie, które się otworzy zakładkę „Servlets”(2). Parametry dodaje się w sekcji „Initialization Parameters”(3). Aby dodać nowy parametr, należy wcisnąć przycisk „Add”. Pojawi się następujące okienko dialogowe.



Nazwa parametru

Wartość parametru

c) Utwórz jeden parametr inicjalizacyjny (np. taki jak na rysunku powyżej) i wciśnij OK.

d) Odczytaj parametr inicjalizacyjny w metodzie „init” i zapamiętaj go w zmiennej stanowiącej pole klasy, a następnie wyświetl wartość tej zmiennej w metodzie processRequest. Wartości parametrów inicjalizacyjnych są dostępne poprzez metodę getInitParameter klasy ServletConfig. Przykładowo:

```
public void init(ServletConfig config) {
    msg = config.getInitParameter("message");
}
```

Nazwa parametru

Uwaga! należy zaimportować klasę ServletConfig: `import javax.servlet.ServletConfig;`

6) Obsługa sesji.

Sesja jest mechanizmem zapewniającym komunikację (przekazywanie obiektów) pomiędzy kolejnymi żądaniami przychodzącymi z tego samego okna przeglądarki. Sesję można traktować jako „worek” skojarzony z konkretnym oknem przeglądarki, do którego wrzucamy, i z którego wyciągamy, obiekty oznaczone odpowiednimi etykietami. Poniższe ćwiczenie demonstruje jak można użyć mechanizmu sesji do implementacji mechanizmu logowania i wylogowywania. **Uwaga! Kolejne kroki od a) do e) wykonaj bez uruchamiania serwletu.** Staraj się zrozumieć kolejne kawałki kodu. Bez tego zrobienie całego ćwiczenia może się udać tylko fuksem.

a) Sprawdź, czy w sesji zapisana jest informacja o tym, że użytkownik jest zalogowany, tworząc nową sesję, w sytuacji, kiedy jest to konieczne. Poniższy kod należy umieścić w serwlecie pomiędzy linią wyświetlającą `<body>` a linią wyświetlającą `</body>`.

Uwaga! Należy zaimportować klasę HttpSession: `import javax.servlet.http.HttpSession;`

```
HttpSession session=request.getSession(true); //Pobierz obiekt sesji, bądź utwórz
jeśli go nie ma
5 Boolean loggedIn=(Boolean)session.getAttribute("zalogowany"); //Pobierz z sesji
Obiekt oznaczony
Etykieta zalogowany
if (loggedIn==null) loggedIn=false; //Jeżeli w sesji nie było informacji o tym,
//że ktoś został zalogowany, to znaczy, że nie został.
2
if (loggedIn==true) {
4 //Użytkownik został zalogowany
} else {
1 //Użytkownik nie został zalogowany
}
```

- b) W sytuacji, gdy użytkownik nie został zalogowany, program powinien wyświetlać formularz pytający się o hasło i o użytkownika. Poniżej przykładowy kod takiego formularza. Pamiętaj, że należy kolejne wiersze tego formularza wysyłać do przeglądarki za pomocą `out.println`. Kod należy umieścić w miejscu oznaczonym przez (1) w poprzednim podpunkcie. Poniższy formularz wykorzystuje metodę GET, żeby można było zaobserwować co jest przekazywane przez przeglądarkę do serwletu. W praktycznych zastosowaniach formularz powinien wykorzystywać metodę POST. Zastanów się dlaczego.

```
<form method="get">
  <input type="text" name="user"/>
  <input type="password" name="pass"/>
  <input type="submit" value="zaloguj"/>
</form>
```

- c) Dopisz teraz w miejscu oznaczonym przez (2) w punkcie a) kod, który wykonuje następujące operacje:

- Sprawdza, czy użytkownik jest zalogowany i jeżeli nie, to czy próbuje się zalogować:

```
if (loggedIn!=true) {
  String user;
  String pass;
  user=request.getParameter("user");
  pass=request.getParameter("pass");
  if (user!=null && pass!=null) {
3    //Użytkownik próbuje się zalogować
  }
}
```

- Jeżeli tak, to sprawdza, czy hasło i użytkownik są poprawne. Poniższy kod należy wstawić w miejscu oznaczonym przez (3) poprzednim podpunkcie. Jeżeli są poprawne, to zapisuje do sesji informację o tym, że użytkownik został zalogowany, oraz przypisuje `true` to zmiennej `loggedIn`:

```
if (user.equals("witek") && pass.equals("haslo")) {
  loggedIn=true;
  session.setAttribute("zalogowany", loggedIn);
}
```

- Zwróć uwagę, że teraz kod przeznaczony dla zalogowanego użytkownika (oznaczony przez (4) w punkcie a)) wykona się w dwóch sytuacjach: zmienna `loggedIn` jest `true` bo tak zostało zapisane w sesji, albo zmienna `loggedIn` jest `true` bo użytkownik podał poprawnego użytkownika i hasło.

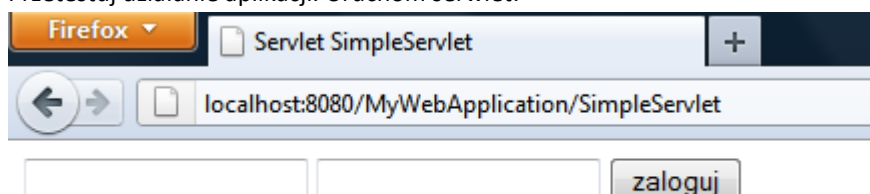
- d) Teraz należy napisać kod, który się wykonuje, kiedy użytkownik został zalogowany. W naszym prostym przypadku będzie to tylko wyświetlenie komunikatu, oraz utworzenie przycisku wylogowującego. Poniższy kawałek kodu, należy umieścić w miejscu oznaczonym przez (4) w punkcie a).

```
out.println("<h1>ZALOGOWANY</h1>");
out.println("<form method=\"get\">");
out.println("<input type=\"hidden\" name=\"akcja\" value=\"wyloguj\" \\\>");
out.println("<input type=\"submit\" value=\"Wyloguj\" \\\>");
out.println("</form>");
```

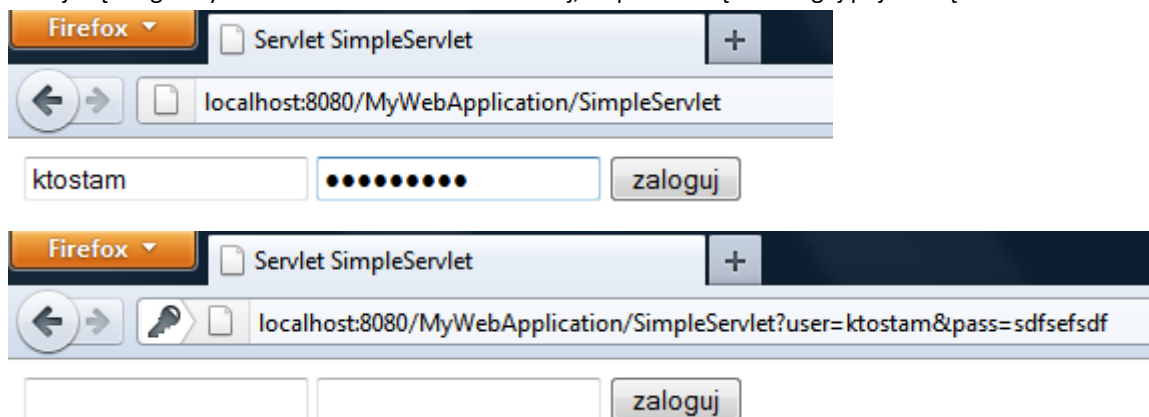
- e) Formularz wyświetlany przez powyższy kod, zawiera jedno pole typu INPUT, i jest to pole typu HIDDEN. Pola typu HIDDEN są bardzo specyficzne, bo nie można ich edytować i są one niewidoczne na stronie, ale ich wartość zostanie przekazana jako parametr w momencie wciśnięcia przycisku SUBMIT. Ostatnim krokiem, jest dopisanie kodu obsługującego wylogowanie. Wylogowanie będzie wykonywane poprzez zapisanie do sesji, do obiektu oznaczonego etykietą `zalogowany` wartości `false`. Poniższy kod należy umieścić w miejscu oznaczonym przez (5) w punkcie a).

```
String action=request.getParameter("akcja");
if (action!=null) {
    if (action.equals("wyloguj")) {
        Boolean wartosc=false;
        session.setAttribute("zalogowany",wartosc);
    }
}
```

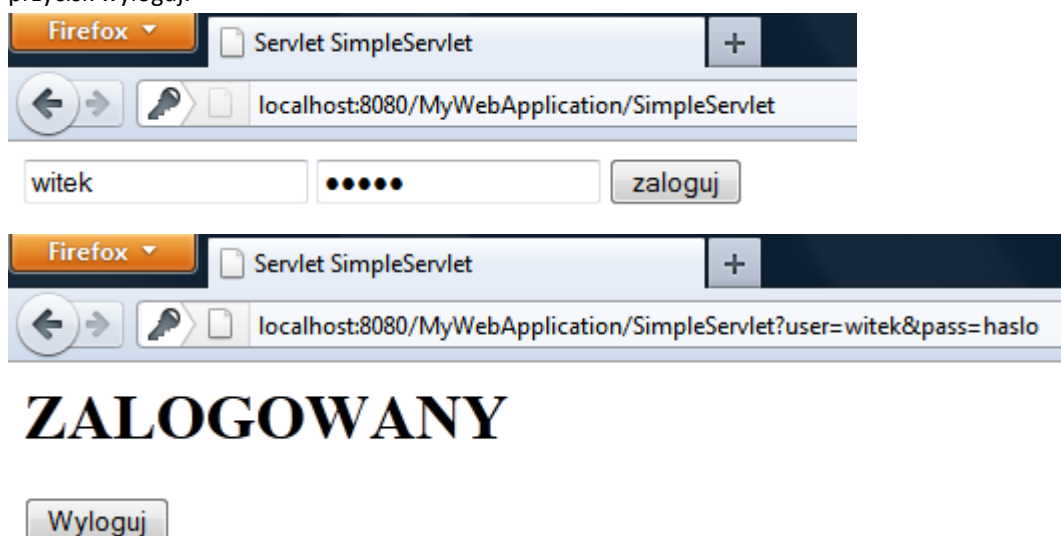
- f) Przetestuj działanie aplikacji. Uruchom serwet:



- g) Podaj błędnego użytkownika albo hasło. Zaobserwuj, że po wciśnięciu zaloguj pojawi się znowu formularz.



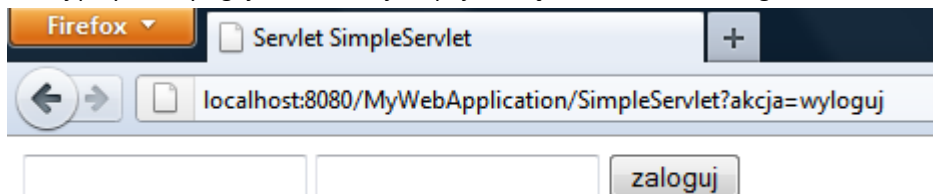
- h) Podaj poprawnego użytkownika i hasło. Po zalogowaniu powinien się pojawić komunikat o poprawnym zalogowaniu i przycisk wyloguj.



- i) Usuń z adresu w przeglądarce parametry logowania i odśwież zawartość okna przeglądarki. Zauważysz, że pomimo tego, że nie przekazujemy danych dotyczących użytkownika, to pozostajemy zalogowani. Dzieje się tak dzięki temu, że informację o zalogowaniu zapisano w sesji.



- j) Kliknij przycisk wyloguj. Zaobserwuj, że pojawi się znowu formularz logowania.



7) Ciasteczka.

- a) Napisz licznik, który liczy liczbę razy, przeglądarka odwiedziła daną stronę. Licznik powinien być przechowywany w ulotnym ciasteczku. Poniższy kod implementuje taki licznik. Przeanalizuj jego działanie. Zastanów się w szczególności po co są linijki oznaczone na czerwono.

Uwaga! Zaimportuj klasę Cookie: `import javax.servlet.http.Cookie;`

```

Cookie[] cookies=request.getCookies();
Cookie licznik=null;

if (cookies!=null) {
    for (int i=0;i<cookies.length;i++) {
        if (cookies[i].getName().equals("licznik")) {
            licznik=cookies[i];
            break;
        }
    }
}

if (licznik==null) {
    licznik=new Cookie("licznik","0");
} else {
    Integer v=Integer.parseInt(licznik.getValue());
    v++;
    licznik.setValue(v+""); //zastanów się po co do v dodajemy "" ?
}

licznik.setMaxAge(-1);
response.addCookie(licznik);

out.println(licznik.getValue());

```

- b) Po zwiększeniu licznika poprzez kilkukrotne odświeżenie przeglądarki, należy wyjść z przeglądarki i uruchomić serwet jeszcze raz, i zaobserwować, że licznik się wykasował. **Uwaga! Jeżeli Firefox spyta się czy zapamiętać karty przeglądarki, to należy odpowiedzieć NIE, bo inaczej ciasteczka też zostaną zapamiętane.**
- c) Teraz należy zwiększyć czas życia ciasteczka do 1 dnia i powtórzyć eksperyment.

8) Zadania

- Napisz serwlet generujący tabelę HTML z tabliczką mnożenia.
- Napisz serwet przyjmujący poprzez formularz łańcuch, obliczający jego długość i generujący w odpowiedzi stronę z długością łańcucha i formularzem pozwalającym na wpisanie kolejnego łańcucha.
- Napisz dwa serwlety kooperujące ze sobą. Pierwszy serwlet wyświetla aktualną datę i formularz pozwalający na wprowadzenie np. imienia. Formularz powinien przekierowywać wynik do drugiego serwletu, który wyświetla komunikat: Cześć + <imie> i pokazuje linka pozwalającego wrócić do pierwszego serwletu. Aktualną datę można pobrać w następujący sposób:

```
DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
Date date = new Date();
String dataLancuch=dateFormat.format(date);
```

- Zaimplementuj bardziej zaawansowany kalkulator niż w ćwiczeniu 3. Prócz podawania liczb dodaj jeszcze pole (np. pole combi) za pomocą którego można wybrać rodzaj operacji matematycznej.
- Napisz serwet, który losuje jakąś liczbę i zapisuje ją do sesji (jeżeli liczba jest już w sesji, to powinna być jedynie odczytywana). Następnie serwlet powinien wyświetlić formularz pozwalający użytkownikowi odgadnąć tą liczbę. Serwlet powinien następnie odpowiadać komunikatami: „za dużo”, „za mało”, „zgadłeś!”. W tej ostatniej sytuacji serwlet powinien wylosować nową liczbę i ponownie wyświetlić formularz do zgadywania.
- Zaimplementuj prosty czat:

- Zadeklaruj jako synchronizowane pole klasy jakąś kolekcję łańcuchów:

Dodaj na początku pliku 4 importy:

```
import java.util.Collection;
import java.util.Collections;
import java.util.Vector;
import java.util.Iterator;
```

Dodaj następującą deklarację pola klasy implementującej serwlet:

```
Collection<String>
    tab=Collections.synchronizedCollection(new Vector<String>());
```

- Serwlet powinien generować stronę, która się będzie regularnie odświeżać:

Dodaj w odpowiednim miejscu linijkę:

```
out.println("<META HTTP-EQUIV=Refresh CONTENT='10'>");
```

- Serwlet powinien wyświetlać prosty formularz pobierający jakiś tekst:

```
<form method='post'>
<input type='text' name='nazwa_parametru' />
<input type=submit value='ok'>
</form>
```

(zastanów się dlaczego przekazujemy dane z formularza za pomocą metody POST a nie GET)

- Jeżeli jakiś tekst został przekazany jako parametr, powinien on być dodawany do kolekcji:

```
String par=request.getParameter("tekst");
if (par!=null) tab.add(par);
```

- Serwlet powinien wyświetlać całą zawartość kolekcji:

```
Iterator<String> it=tab.iterator();

while (it.hasNext()) {
    out.println(it.next()+"<br/>");
}
```