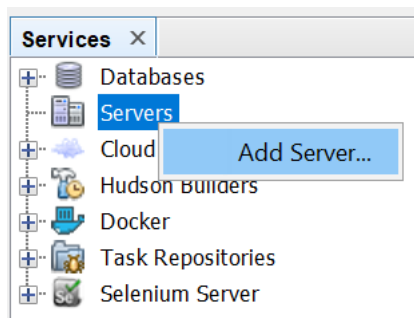


## Serwlety, JSP, JSTL

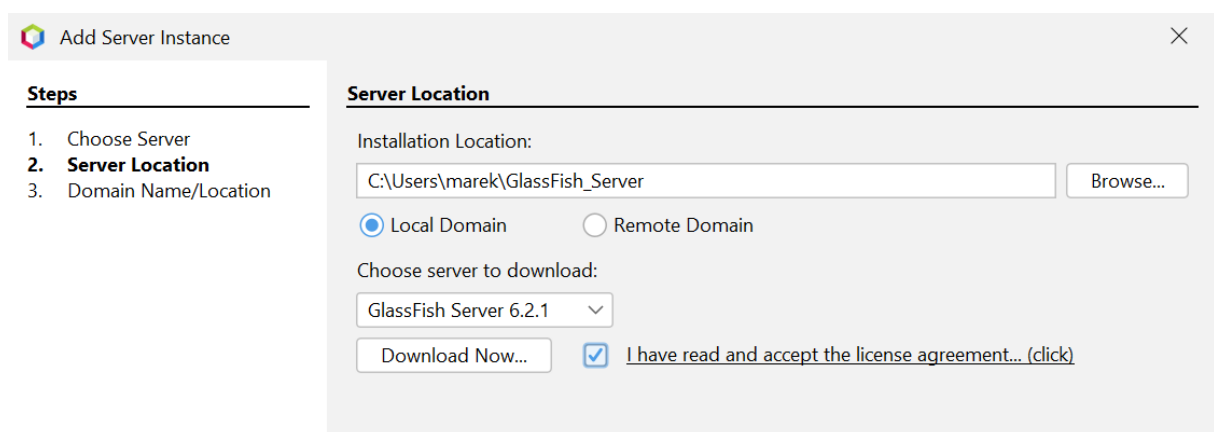
Celem ćwiczenia jest wprowadzenie do klasycznych technologii warstwy prezentacji na platformie Java EE/Jakarta EE, czyli technologii serwletów, JSP i biblioteki znaczników JSTL. Do realizacji ćwiczenia potrzebne jest zintegrowane środowisko programistyczne NetBeans w wersji 17 wraz z serwerem aplikacji GlassFish.

Przygotowanie środowiska do tworzenia i uruchamiania aplikacji Java EE:

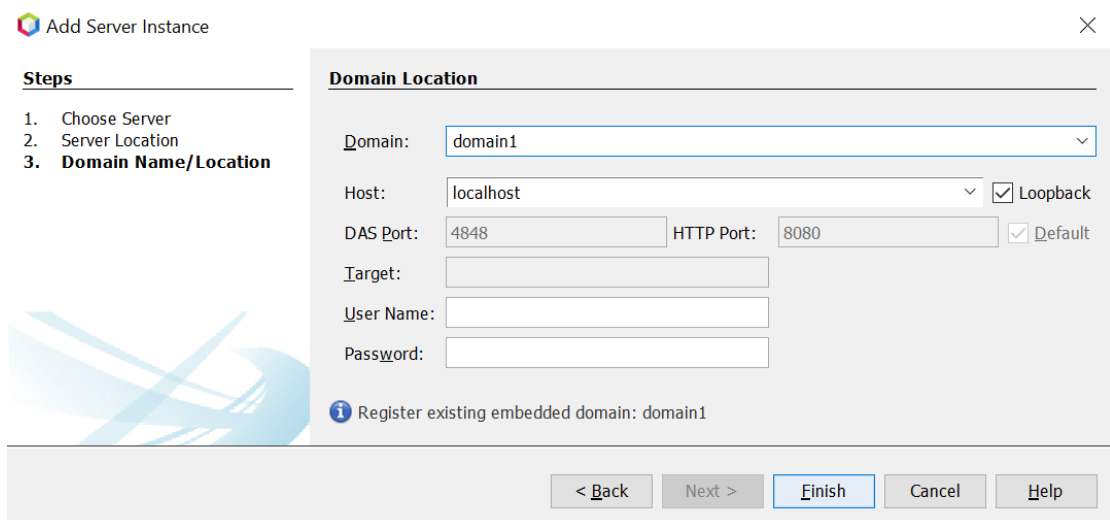
- a) Uruchom NetBeans.
- b) Przejdź do panelu Services (menu Window→Services) i wybierz gałąź Servers. Jeśli gałąź zawiera już skonfigurowany serwer GlassFish to środowisko jest skonfigurowane i można przystąpić do pierwszego kroku ćwiczenia. W przeciwnym wypadku z menu kontekstowego dla gałęzi Servers wybierz opcję Add Server.



- c) Z listy dostępnych serwerów wybierz GlassFish Server. Kliknij **Next**.
- d) Jeśli wcześniej nie była w NetBeans aktywowana opcja Java Web and EE, to pojawi się okienko informujące o jej aktywacji. Oczywiście opcja ta jest niezbędna do realizacji ćwiczeń.  
Zaakceptuj instalację wymaganych komponentów i warunki licencji.
- e) W oknie dodawania instancji serwera wybierz najnowszą dostępną wersję serwera GlassFish do pobrania, potwierdź zapoznanie się z licencją i kliknij **Download Now...**

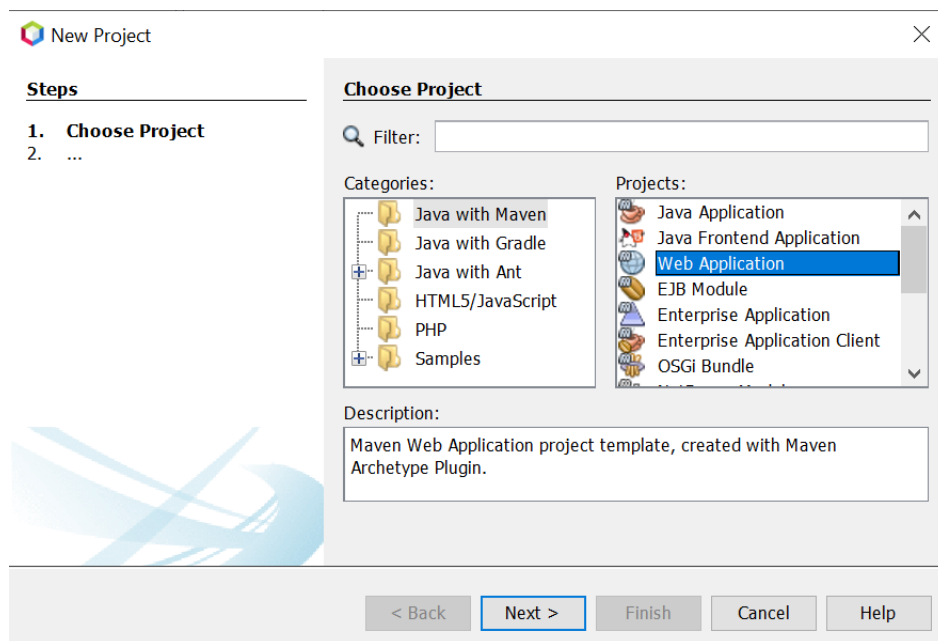


- f) Po zakończeniu pobierania i instalacji serwera kliknij **Next**.
- g) W kroku rejestracji domeny serwera pozostaw zaproponowane ustawienia (dostęp administracyjny do serwera nie będzie chroniony hasłem) i kliknij **Finish**.

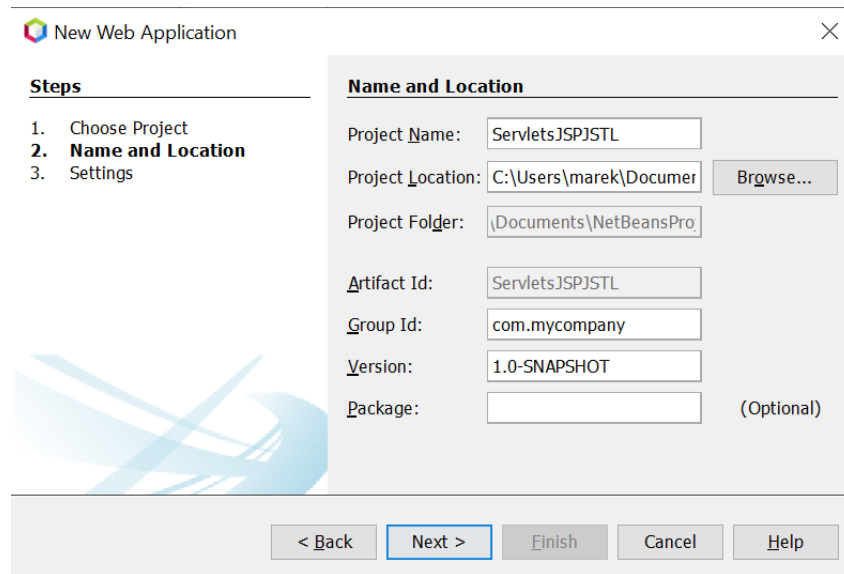


## 1. Utworzenie projektu webowego z wykorzystaniem Mavena

- a) W środowisku NetBeans z menu File wybierz opcję New Project... Wybierz kategorię Java with Maven i typ projektu Web Application. Kliknij przycisk **Next >**



- b) Jako nazwę projektu wpisz **ServletsJSPJSTL**. Nie zmieniaj wartości pozostałych opcji. Kliknij przycisk **Next >**.

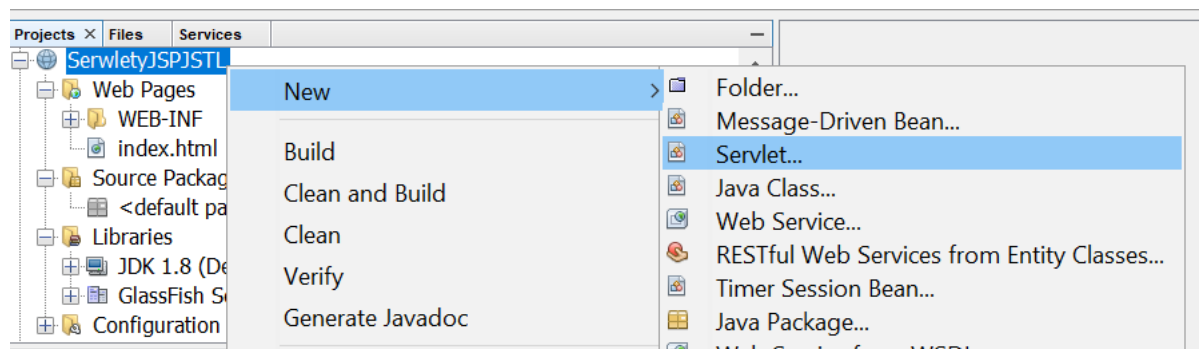


- c) Jako serwer do uruchamiania aplikacji wybierz serwer GlassFish. Jako wersję Javy EE wybierz **Jakarta EE 9.1 Web**. Kliknij przycisk **Finish**.

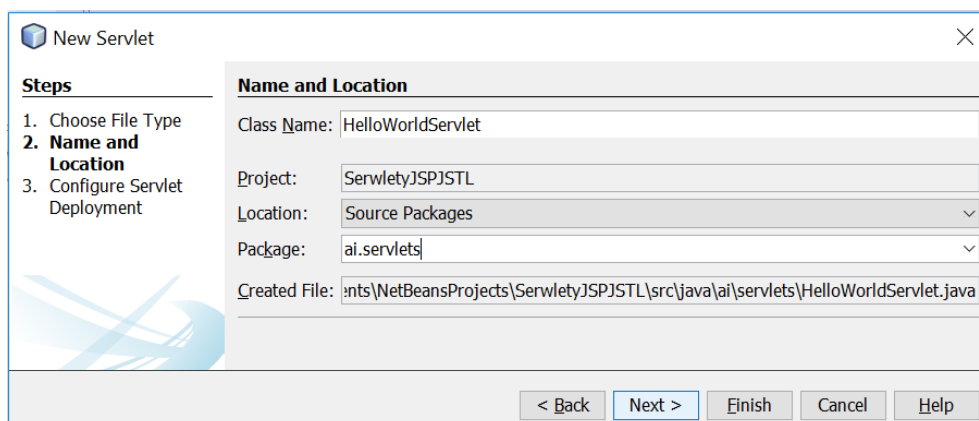
- d) Przeanalizuj strukturę projektu, jaki powstał w wyniku działania kreatora.

## 2. Utworzenie i uruchomienie pierwszego serwletu

- a) Kliknij prawym klawiszem na ikonie projektu w nawigаторze projektów, następnie z menu kontekstowego wybierz opcję **New → Servlet**



- b) Jako nazwę serwletu podaj **HelloWorldServlet**. Jako nazwę pakietu wpisz **ai.servlets**. Kliknij przycisk **Next >**



- c) W ostatnim kroku kreatora pozostaw odznaczoną opcję dodania informacji o serwlecie do pliku deskryptora instalacji `web.xml`. Zwróć uwagę na to, jaki adres będzie wywoływał serwlet. Kliknij przycisk **Finish**.

**New Servlet**

**Steps**

1. Choose File Type
2. Name and Location
3. **Configure Servlet Deployment**

**Configure Servlet Deployment**

Register the Servlet with the application by giving the Servlet an internal name (Servlet Name). Then specify patterns that identify the URLs that invoke the Servlet. Separate multiple patterns with commas.

☒ Add information to deployment descriptor (web.xml)

Class Name:

Servlet Name:

URL Pattern(s):

Initialization Parameters:

Name	Value
------	-------

New Edit... Delete

< Back Next > **Finish** Cancel Help

**Komentarz:** Do wersji Java EE 5 konfiguracja serwletów (i całego modułu webowego aplikacji Java EE) realizowana była w oparciu o plik `web.xml`. Od wersji Java EE 6 plik ten jest opcjonalny, gdyż podstawowa konfiguracja może być wyspecyfikowana poprzez adnotacje w kodzie Java. Plik `web.xml` jest używany do „nadpisania” ustawień zawartych w aplikacji po jej instalacji (bez konieczności rekompilacji aplikacji) oraz dla ustawień zaawansowanych, których nie można zrealizować poprzez adnotacje.

- d) Po zakończeniu działania kreatora kod źródłowy serwletu zostanie otwarty w edytorze kodu. Sprawdź czy kreator dodał właściwe importy dla platformy Jakarta EE. Jeśli kreator dodał importy klas z pakietów Java EE (zaczynających się prefiksem `javax`), to zmień nazwy pakietów na właściwe dla platformy Jakarta EE (zaczynające się prefiksem `jakarta`). Następnie odszukaj adnotację specyfikującą nazwę serwletu i jego odwzorowanie (`@WebServlet`). Przeanalizuj w jaki sposób kreator zapewnił, że serwlet będzie obsługiwał żądania GET i POST, reagując na nie w ten sam sposób.

**Komentarz:** Identyczna reakcja na żądania GET i POST była często spotykana w czasach pierwszych aplikacji webowych do obsługi formularzy HTML.

- e) Usuń z klasy serwletu metody `doGet`, `doPost`, `processRequest` i `getServletInfo`. Następnie dodaj poniższą implementację metody `doGet`. Wygeneruje ona dynamiczną stronę WWW, zatytułowaną „Hello World Servlet”, wyświetlającą w przeglądarce tekst „Hello World”.

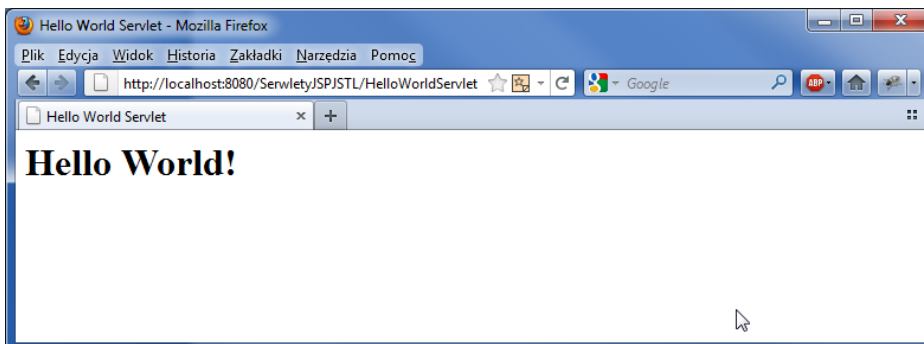
```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

    response.setContentType("text/html");
    response.setCharacterEncoding("windows-1250");

    PrintWriter out = response.getWriter();

    out.println("<html>");
    out.println("<head><title>Hello World Servlet</title></head>");
    out.println("<body>");
    out.println("<h1>Hello World!</h1>");
    out.println("</body>");
    out.println("</html>");
    out.close();
}
```

- f) Uruchom projekt. Spowoduje to zbudowanie archiwum instalacyjnego aplikacji oraz uruchomienie serwera GlassFish i zainstalowanie (deployment) jej na serwerze (możesz śledzić komunikaty w panelu Output, które to potwierdzają). NetBeans powinien też uruchomić przeglądarkę i przejść do domyślnej strony aplikacji. Popraw adres w przeglądarce na wywołujący utworzony wcześniej serwlet.



### 3. Zmiana odwzorowania serwletu na adres URL

- a) Wróć do edycji kodu klasy serwletu. Poprzez modyfikację adnotacji zmień domyślne odwzorowanie na `/hws`.

```
@WebServlet(name = "HelloWorldServlet", urlPatterns = {"/hws"})
```

Zapisz zmiany zwracając uwagę na komunikaty pojawiające się na pasku stanu środowiska NetBeans. Wróć do przeglądarki i odśwież zawartość dokumentu (nie zmieniaj adresu). Co zauważyłaś/eś? Następnie wprowadź w przeglądarce adres `http://localhost:8080/ServletsJSPJSTL/hws`

Komentarz: NetBeans domyślnie przy zapisie zmian w kodzie powinien przebudować projekt, a następnie wykonać jego deployment na serwerze. W takim wypadku zmiany w działaniu w aplikacji można zaobserwować przy odświeżeniu strony lub przejściu do nowego adresu w przeglądarce. Jeśli opisany mechanizm nie zadziała (może się tak zdarzyć w przypadku niektórych zmian), należy jawnie ponownie uruchomić projekt.

b) Powtórz kroki z punktu a) dla następujących dwóch odwzorowań: \*.hws, /hws/\* .  
Chcemy uzyskać poniższe adresy wywołania serwletu:

- `http://localhost:8080/ServletsJSPJSTL/cokolwiek.hws`
- `http://localhost:8080/ServletsJSPJSTL/hws/cokolwiek`

c) Przywróć pierwotne odwzorowanie serwletu.

#### 4. Przekazywanie parametrów z formularza HTML do serwletu

a) Kliknij prawym klawiszem myszy na ikonie projektu i wybierz New → HTML. Jako nazwę dokumentu podaj form (rozszerzenie \*.html zostanie dodane automatycznie). Umieść w dokumencie poniższy kod.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>HTML form</title>
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1250">
  </head>
  <body>
    <form action="HelloWorldServlet" method="get">
      <table>
        <tr><td>Name:</td><td><input type="text" name="name"/></td></tr>
        <tr><td>Age: </td><td><input type="text" name="age" size="2"/></td></tr>
        <tr><td colspan="2"><input type="submit" value="Send"/></td></tr>
      </table>
    </form>
  </body>
</html>
```

b) W edytorze kodu otwórz zawartość pliku HelloWorldServlet.java. Zmodyfikuj kod metody doGet() w poniższy sposób. Zwróć uwagę na konieczność jawnego rzutowania parametrów innych niż łańcuchy znaków.

```
response.setContentType("text/html");
response.setCharacterEncoding("windows-1250");

PrintWriter out = response.getWriter();

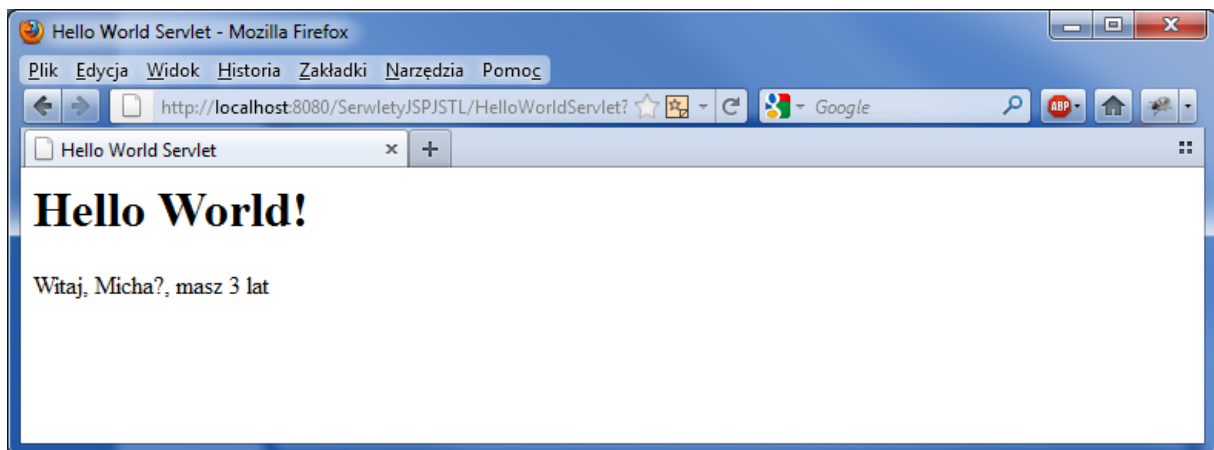
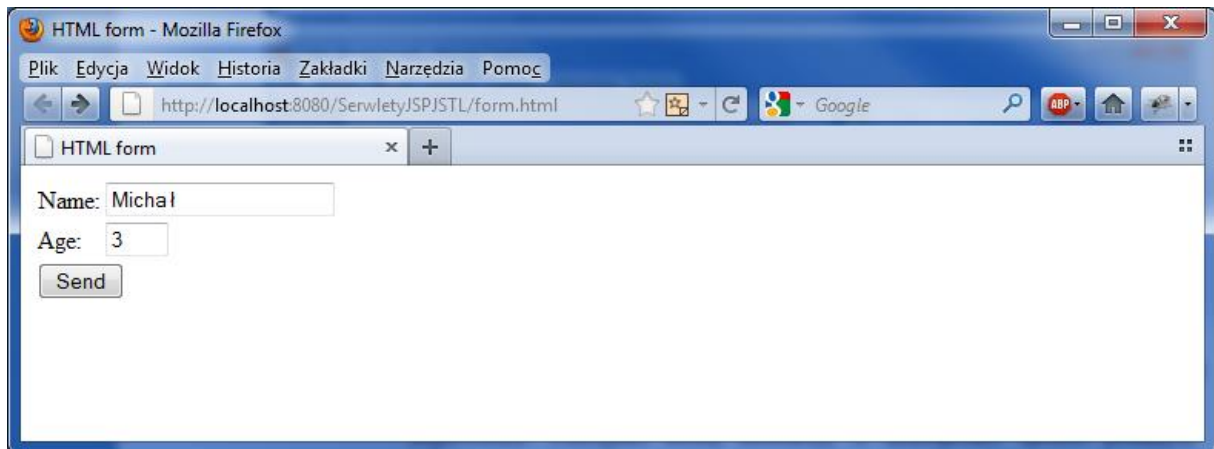
String name = request.getParameter("name");
int age = Integer.parseInt(request.getParameter("age"));

out.println("<html>");
out.println("<head><title>Hello World Servlet</title></head>");
out.println("<body>");
out.println("<h1>Hello World!</h1>");

out.println("<p>Witaj, " + name + ", masz " + age + " lat</p>");

out.println("</body>");
out.println("</html>");
out.close();
```

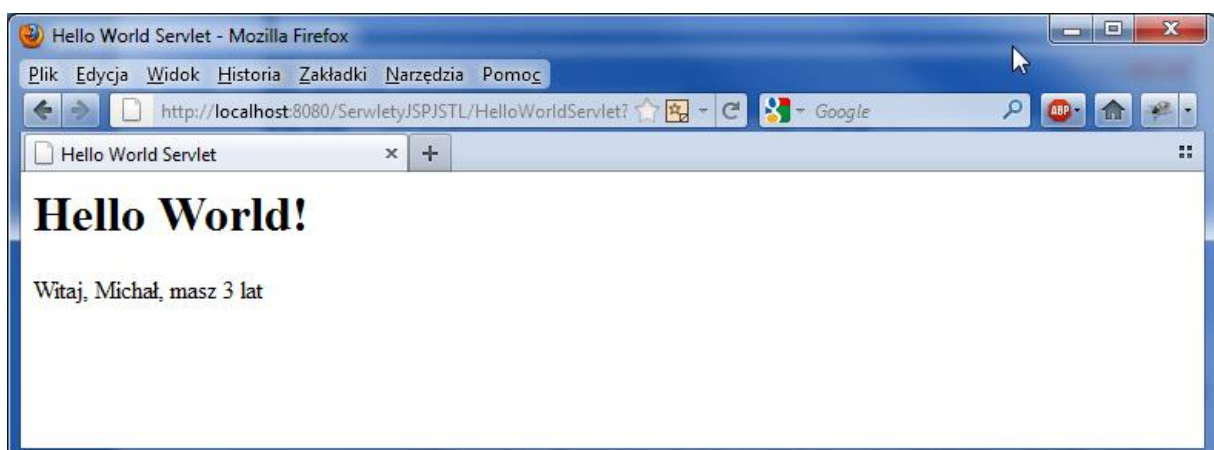
- c) Kliknij prawym klawiszem myszy na pliku `form.html` (w oknie Projects, gałąź Web Pages), z menu kontekstowego wybierz pozycję Run File. Wypełnij pola formularza i prześlij formularz do przetworzenia w serwlecie.



- d) Odszukaj w kodzie serwletu wiersz ustawiający kodowanie dla generowanej odpowiedzi. Następnie dodaj instrukcję w analogiczny sposób podającą informację o kodowaniu dla żądania (obiekt `request`). Zwróć uwagę aby informacja ta była ustawiona przed odczytem wartości parametrów z żądania.

```
request.setCharacterEncoding("windows-1250");
```

Ponownie uruchom formularz i sprawdź działanie serwletu dla imienia z polskimi znakami.



- e) Zmień w formularzu HTML metodę wywoływania serwletu na POST.

```
<form action="HelloWorldServlet" method="post">
```

Zainstaluj aplikację na serwerze. Ponownie otwórz formularz w przeglądarce i spróbuj wysłać formularz do serwletu. Zinterpretuj otrzymany komunikat.

- f) Do klasy HelloWorldServlet dodaj metodę doPost, która będzie obsługiwała metodę wywołania POST. Następnie zainstaluj aplikację na serwerze i ponownie spróbuj przesłać wypełniony formularz do serwletu.

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    this.doGet(request, response);
}
```

## 5. Obsługa sesji w serwlecie

- a) Utwórz w projekcie nowy serwlet i nazwij go HttpSessionServlet. Umieść nowo utworzony serwlet w pakiecie ai.servlets. Serwlet będzie umożliwiał przechowywanie krótkich notatek pomiędzy kolejnymi wywołaniami dokumentu. Pamiętaj, aby nie zaznaczać opcji dodania informacji o serwlecie do pliku deskryptora instalacji web.xml.
- b) Jako kod serwletu podaj poniższy kod. Zwróć szczególną uwagę na obsługę sesji, utworzenie nowego obiektu do przechowywania notatek w przypadku gdy obiekt sesyjny jest pusty oraz konieczność każdorazowego utrwalenia obiektu sesyjnego. Uruchom aplikację. Przetestuj działanie serwletu dla dwóch współbieżnych sesji (uruchom inną przeglądarkę i skopiuj adres wywołujący serwlet do paska adresu).

```
package ai.servlets;

import java.io.*;
import java.util.ArrayList;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.*;
import jakarta.servlet.http.*;

@WebServlet(name="HttpSessionServlet", urlPatterns={"/HttpSessionServlet"})
public class HttpSessionServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        HttpSession session = request.getSession();

        ArrayList notes = (ArrayList)session.getAttribute("notes");
        if (notes == null) {
            notes = new ArrayList ();
            session.setAttribute("notes",notes);
        }

        String note = request.getParameter("note");
        if (note != null)
            notes.add(note);
    }
}
```



```

        PrintWriter out = response.getWriter();

        out.println("<html>");    out.println("<body>");
        out.println("<h1>My notes</h1>");
        out.println("<ul>");

        for (int i = 0; i < notes.size(); i++)
            out.println("<li>" + notes.get(i));

        out.println("</ul>");

        out.println("<form action='HttpSessionServlet'>");
        out.println("<input type='text' name='note'><br/>");
        out.println("<input type='submit' value='Add note'>");
        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    }
}

```

9. Współpraca między formularzem, serwletem, komponentem Java Bean oraz stroną JSP (przekierowanie żądania z serwletu do JSP).

a) Utwórz w projekcie prostą stronę HTML o nazwie yesterday.html a następnie wypełnij ją poniższym kodem.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
  <head>
    <meta http-equiv="Content-Type"
          content="text/html; charset=windows-1250"/>
    <title>Data entry form</title>
  </head>
  <body>
    <form action="ControllerServlet" method="post">
      <table>
        <tr><td>tekst</td>
          <td><select name="foreColor">
            <option>red</option><option>green</option><option>black</option>
          </select></td></tr>
        <tr><td>tłó</td>
          <td><select name="backColor">
            <option>yellow</option><option>magenta</option><option>gray</option>
          </select></td></tr>
        <tr><td colspan="2"><input type="submit" value="wyślij"></td></tr>
      </table>
    </form>
  </body>
</html>

```

- b) Przygotuj klasę Java Bean, która będzie wykorzystywana do komunikacji między serwletem i stroną JSP. Dodaj do projektu nową klasę `ColorBean` (w pliku `ColorBean.java`), umieść klasę w pakiecie `ai.beans`. Zastąp kod wygenerowany automatycznie poniższym. Nie zapomnij wygenerować metod *getter* i *setter* i dla obu pól składowych (użyj mechanizmu refaktoryzacji kodu).

```
package ai.beans;

public class ColorBean {

    private String foregroundColor;
    private String backgroundColor;

    public ColorBean() {
    }

}
```

- c) W kolejnym kroku dodaj do projektu serwlet o nazwie `ControllerServlet` i umieść go w pakiecie `ai.servlets` (serwlet jest wykorzystany w metodzie `action` formularza z pliku `yesterday.html`). Kod serwletu znajduje się poniżej. Zwróć uwagę na wykorzystanie komponentu Java Bean do komunikacji między warstwami aplikacji.

```
package ai.servlets;

import java.io.*;

import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.WebServlet;

import ai.beans.ColorBean;

@WebServlet(name = "ControllerServlet", urlPatterns = {"/ControllerServlet"})
public class ControllerServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequestRequest request,
                          HttpServletResponse response)
        throws ServletException, IOException {
        doPost(request, response);
    }

    @Override
    protected void doPost(HttpServletRequestRequest request,
                          HttpServletResponse response)
        throws ServletException, IOException {
        ColorBean myBean = new ColorBean();
        myBean.setForegroundColor( request.getParameter("foreColor") );
        myBean.setBackgroundColor( request.getParameter("backColor") );
        request.setAttribute("bean", myBean);

        ServletContext ctx = this.getServletContext();
        RequestDispatcher dispatcher =
            ctx.getRequestDispatcher("/yesterday.jsp");
        dispatcher.forward(request, response);
    }

}
```

- d) W ostatnim kroku stwórz stronę JSP o nazwie yesterday.jsp i wypełnij poniższym kodem. Następnie wróć do strony yesterday.html i uruchom aplikację. Przetestuj jej działanie.

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
    <title>Yesterday</title>
    <style type="text/css">
        body { background: ${bean.backgroundColor};
                color: ${bean.foregroundColor} }
    </style>
</head>
<body>

<h2>Yesterday</h2>

    Yesterday,<br/>
    All my troubles seemed so far away,<br/>
    Now it looks as though they're here to stay,<br/>
    Oh, I believe in yesterday.<br/>
    <br/>
    Suddenly,<br/>
    I'm not half the man I used to be,<br/>
    There's a shadow hanging over me,<br/>
    Oh, yesterday came suddenly.<br/>
    <br/>
    Why she<br/>
    Had to go I don't know, she wouldn't say.<br/>
    I said,<br/>
    Something wrong, now I long for yesterday.<br/>

    <a href="yesterday.html">powrót</a>
</body>
</html>
```

## 10. JSTL – Standardowa biblioteka znaczników dla JSP

- a) W kodzie serwletu z poprzedniego zadania umieść poniższe instrukcje (przed przekierowaniem żądania do JSP) rejestrujące w zasięgu bieżącego żądania 2 obiekty: datę wydania piosenki i kolekcję ze składem zespołu:

```
ArrayList members = new ArrayList();
members.add("John Lennon");
members.add("Paul McCartney");
members.add("Ringo Starr");
members.add("George Harrison");
request.setAttribute("members", members);

Calendar calDate = new GregorianCalendar();
calDate.set(1965, Calendar.SEPTEMBER, 13);
Date releaseDate = calDate.getTime();
request.setAttribute("releaseDate", releaseDate);
```

- b) Zaimportuj wykorzystywane we wklejonym kodzie klasy ArrayList, Date, Calendar i GregorianCalendar z pakietu java.util.
- c) W kodzie strony JSP, wyświetlającej dane przygotowane przez serwlet (yesterday.jsp), dodaj (pod tekstem piosenki) poniższy kod:

```
<p>
Released: <c:out value="${releaseDate}" />
</p>
<c:if test="${!empty members}">
  <table border>
    <tr><th>The Beatles</th></tr>
    <c:forEach var="member" items="${members}">
      <tr><td>${member}</td></tr>
    </c:forEach>
  </table>
</c:if>
```

- d) Nie zapomnij dodać (jako pierwszego wiersza kodu strony JSP) dyrektywy deklarującej użycie wykorzystywanego zestawu znaczników JSTL:

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
```

- e) Uruchom i przetestuj aplikację (uruchamiając formularz HTML)
- f) Zastąp znacznik wyświetlający przekazaną datę znacznikiem formatującym:

```
<fmt:formatDate value="${releaseDate}" type="date" dateStyle="default"/>
```

- g) Nie zapomnij o dyrektywie deklarującej użycie zestawu znaczników formatujących JSTL:

```
<%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
```

- e) Przetestuj inne formaty daty: short, medium, long i full (zastępuj wartość default dla atrybutu `dateStyle`).

### **Zadanie do samodzielnego wykonania**

Dodaj na stronie `yesterday.html` pole wyboru wskazujące czy tabelka z członkami zespołu ma mieć widoczne krawędzie czy nie. Informacja o dokonanym wyborze obramowania powinna być przekazana stronie JSP tą samą drogą co informacje o kolorach.