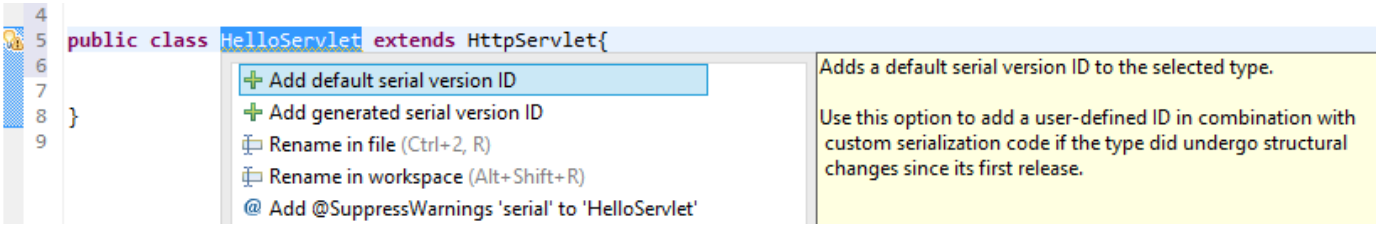
Ćwiczenie 1

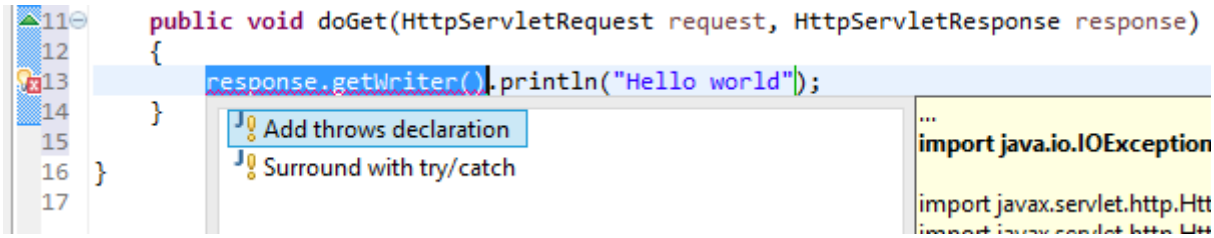
1. Konfiguracja pluginu jetty, do automatycznego odświeżania zmian
    a. W pliku pom.xml w pluginie jetty'ego dodaj konfigurację 'scanIntervalSeconds' i ustaw ją na 5

```
<plugin>
    <groupId>org.eclipse.jetty</groupId>
    <artifactId>jetty-maven-plugin</artifactId>
    <version>9.3.7.v20160115</version>
    <configuration>
      <scanIntervalSeconds>5</scanIntervalSeconds>
    </configuration>
  </plugin>
```

2. Utworzenie nowego serwletu.
    a. Do nowoutworzonego projektu dodaj nowy pakiet i nazwij go 'servlets'
    b. Do pakietu 'servlets' dodaj klasę 'HelloServlet'
        - Klasa powinna dzidziczyć po 'HttpServlet'
        - Zwróć uwagę na wyświetlone ostrzeżenie



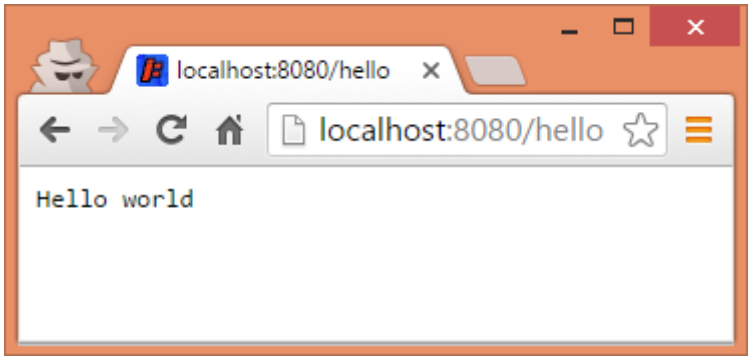
        - Wybierz pierwszą proponowaną opcję
3. Obsługa odpowiedzi http.
    a. Do serwletu 'HelloServlet' dodaj metodę o nazwie 'doGet'
        - Metoda przyjmuje dwa parametry : HttpServletRequest, HttpServletResponse
        - Do ciała metody dodaj linijkę response.getWriter().println(„Hello world")
        - Zwróć uwagę na powstały błąd i rozwiąż go dodając deklarację throws do metody



    b. Do klasy dodaj adnotację WebServlet i parametrze dodaj adres pod jakim będzie nasłuchiwał Servlet i zapisz zmiany.

```
@WebServlet("/hello")
public class HelloServlet extend
```

    c. Uruchom projekt komedą 'mvn jetty:run' i przejdź na na stronę http://localhost:8080/hello



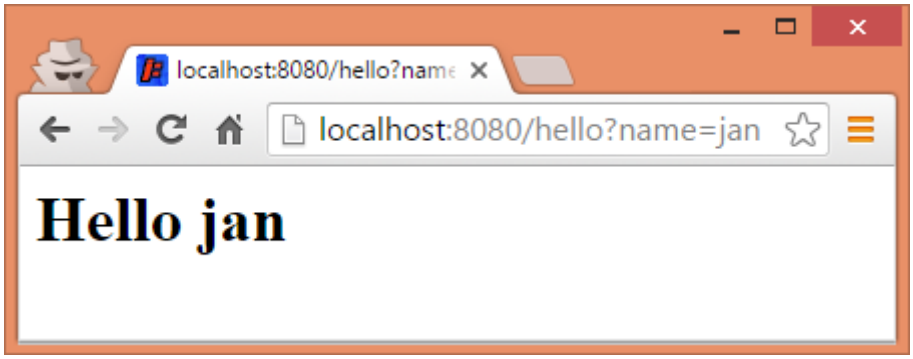
Jak widzimy obiekt response dostarcza informacji przeglądarce które ma wyświetlić.

d. Przesyłanie html
   - Obiekt response posiada metodę ‘setContentType’ dzięki której możemy ustalić typ przesyłanej zawartości
     - Zmień wysyłany text na „<h1>Hello World</h1>”
     - Oraz przetestuj metodę ‘setContentType’ dla wartości ‘text/html’ oraz ‘text/plaintext’

4. Obsługa rządań http typu GET – pobieranie parametrów z adresu url.
   a. Do adresu url dodaj paramater name i nadaj mu wartość ‘Jan’
      - http://localhost:8080/hello?name=Jan
   b. Pobieranie parametru
      - Obiekt request posiada metodę ‘getParameter’, która po wprowadzeniu nazwy parametru pobiera jego wartość
        - Zmień kod metody ‘doGet’ aby wyglądała podobnie jak na przedstawionym obrazku

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException
{
    String name = request.getParameter("name");
    response.setContentType("text/html");
    response.getWriter().println("<h1>Hello "
            + name
            + "</h1>");
}
```

- Wejdź na adres http://localhost:8080/hello?name=Jan

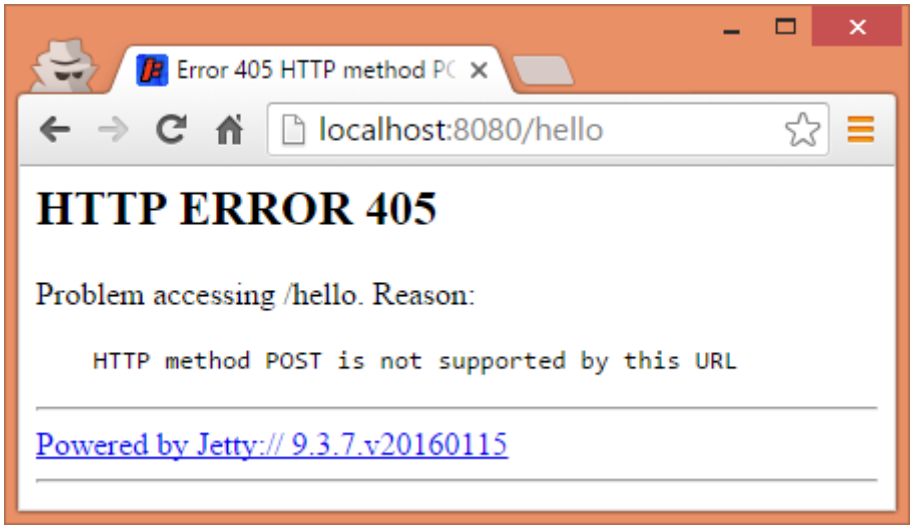


- Sprawdź co się stanie jeśli przekażesz parametr bez wartości oraz co się stanie jeśli nie przekażesz żadnego parametru

5. Utworzenie formularza
    a. W katalogu ‘src/main/webapp’ dodaj plik index.jsp
        - Napisz kod html z formularzem który będzie przesyłał imię na adres servletu

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
    adres do servletu
    <form action="hello" method="get">
        <label>Imie:<input type="text" id="name" name="name"/></label>
        <input type="submit" value="wyslij"/>
    </form>
</body>
</html>
```

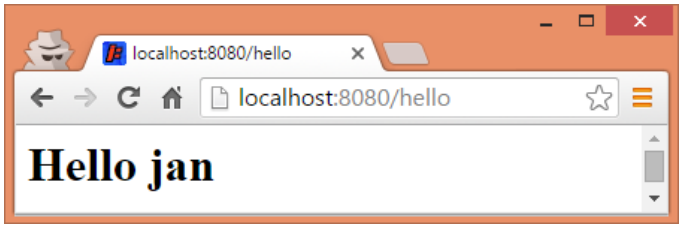
    b. Przetestuj działanie formularza na adresie http://localhost:8080
6. Przekazywanie parametrów metodą POST
    a. Sprawdź co się stanie jeśli w nowoutworzonym formularzu zmienisz metdoę przesyłania na ‘POST’.



    b. W servlecie dodaj metodę ‘doPost’ o takiej samej sygnaturze jak metoda ‘doGet’ i przekopiuj do niej kod metody ‘doGet’

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException
{
    String name = request.getParameter("name");
    response.setContentType("text/html");
    response.getWriter().println("<h1>Hello "
            + name
            + "</h1>");
}
```

Po odświeżeniu strony powinieneś otrzymać



7. Przekierowania stron
    a. Obiekt respnse dostarcza nam metody która pozwala przekierować użytkownika na inny adres. Jako parametr przyjmuje nazwę podstrony do której chcemy się dostać.
    b. Zanim wyświetlimy przywitanie użytkownikowi, chcemy sprawdzić czy do formularza zostało wprowadzone imię – jeśli pole w formularzu jest puste to servlet ma spowrotem przenieść użytkownika na stronę z formularzem.
        - Zmodyfikuj metodę 'doPost':

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException
{
    String name = request.getParameter("name");
    if(name==null || name.equals(""))
    {
        response.sendRedirect("/");
    }
    response.setContentType("text/html");
    response.getWriter().println("<h1>Hello "
            + name
            + "</h1>");
}
```

        - Przetestuj działanie formularza z pustym jak i wypełnionym polem.
8. Testy jednostkowe
    a. Skonfiguruj pom.xml (do mockowania użyj Mockito)
        - Dodatkowo będzie potrzebna paczka servlets-api

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
</dependency>

<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-all</artifactId>
    <version>1.10.19</version>
</dependency>

<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
</dependency>
```

    b. Test sprawdzający co się stanie jeśli servlet dostanie pusty parametr

```java
public class TestHelloServlet extends Mockito{

    @Test
    public void servlet_should_not_greet_the_user_if_the_name_is_null() throws IOException  {
        HttpServletRequest request = mock(HttpServletRequest.class);
        HttpServletResponse response = mock(HttpServletResponse.class);
        HelloServlet servlet = new HelloServlet();

        when(request.getParameter("name")).thenReturn(null);

        servlet.doPost(request, response);

        verify(response).sendRedirect("/");
    }

    @Test
    public void servlet_should_not_greet_the_user_if_the_name_is_empty() throws IOException  {
        HttpServletRequest request = mock(HttpServletRequest.class);
        HttpServletResponse response = mock(HttpServletResponse.class);
        HelloServlet servlet = new HelloServlet();

        when(request.getParameter("name")).thenReturn("");

        servlet.doPost(request, response);

        verify(response).sendRedirect("/");
    }
```

c. Test sprawdzający co się wyświetli na stronie

```java
@Test
public void servlet_should_greet_the_user_when_the_name_is_provided() throws IOException {

    HttpServletRequest request = mock(HttpServletRequest.class);
    HttpServletResponse response = mock(HttpServletResponse.class);
    PrintWriter writer = mock(PrintWriter.class);

    when(request.getParameter("name")).thenReturn("jan");
    when(response.getWriter()).thenReturn(writer);

    new HelloServlet().doPost(request, response);

    verify(writer).println("<h1>Hello jan</h1>");
}
```

Użyj komendy ‘mvn test’ aby uruchomić testowanie.