

Laboratorium 1. Środowisko *JEE* i interfejs serwletów

Cel zajęć

Realizacja zadań z niniejszego laboratorium pozwoli studentom zapoznać się z zasadami tworzenia aplikacji internetowych z wykorzystaniem podstawowych elementów środowiska *Java Enterprise Edition (JEE)*, klasy *HttpServlet* serwletów (ang. *Java Servlet*) [3, 13, 21] oraz z interfejsami *HttpServletRequest* i *HttpServletResponse*.

Zakres tematyczny

- Przygotowanie prostej aplikacji internetowej w *JEE*, korzystającej z klasy *HttpServlet*.
- Poznanie i zastosowanie metod interfejsów *HttpServletRequest* oraz *HttpServletResponse* do:
 - wyświetlania metadanych żądania *HTTP*,
 - pobrania i walidacji parametrów przekazanych z formularza w żądaniu *HTTP*,
 - zarządzania ciasteczkami i sesją *HTTP*.

Wprowadzenie

Podstawą aplikacji internetowych tworzonych w języku *Java* jest serwlet (ang. *Java Servlet*). Serwlet jest programem napisanym w języku *Java*, wykonywanym na kontenerze aplikacji *JEE* (np. na serwerze *Apache Tomcat*). Serwlety stanowią warstwę pośrednią pomiędzy żdaniami przesyłanymi przez przeglądarkę (klienta *HTTP*) oraz aplikacjami działającymi po stronie serwera. Serwlety obsługują żądania *HTTP* (ang. *request*) i generują odpowiedź *HTTP* (ang. *response*) przesyłaną do klienta.

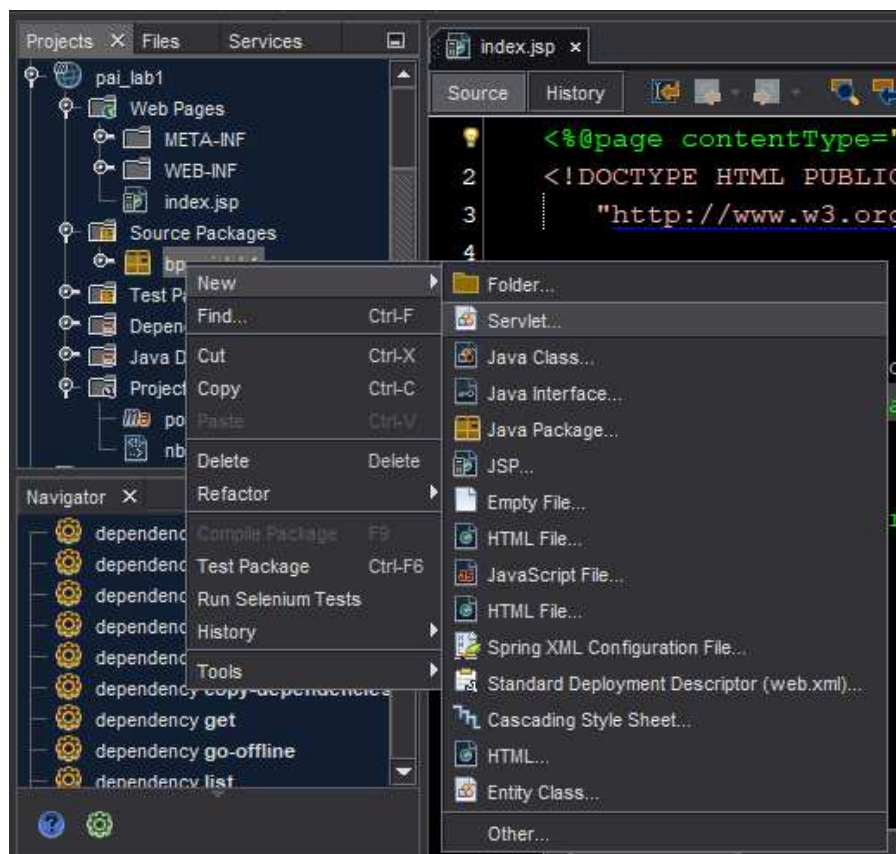
Serwlet umożliwia:

- odczytywanie jawnych informacji przesyłanych przez użytkownika (np. danych z formularza *HTML*),
- odczytywanie niejawnych informacji przesyłanych przez przeglądarkę w żądaniu *HTTP* (np. nagłówków żądania *HTTP*),
- generowanie wyników w różnych formatach (np. *text/html*, *text/json*, *image/jpg* itp.),
- przesyłanie jawnych informacji w różnych formatach do klienta,
- przesyłanie niejawnych informacji w odpowiedzi *HTTP* (np. nagłówków odpowiedzi *HTTP*).

Podstawą działania serwletów są dwa interfejsy: ***HttpServletRequest*** i ***HttpServletResponse***. Całą aplikację internetową można utworzyć korzystając jedynie z możliwości serwletów, jednak przy rozbudowanych aplikacjach nie jest to efektywne. Poznanie interfejsów klasy serwletu pomoże zrozumieć kolejne poznawane w tym skrypcie technologie, które dodają pewien poziom abstrakcji, umożliwiając łatwiejsze i szybsze tworzenie kodu w języku *Java*.

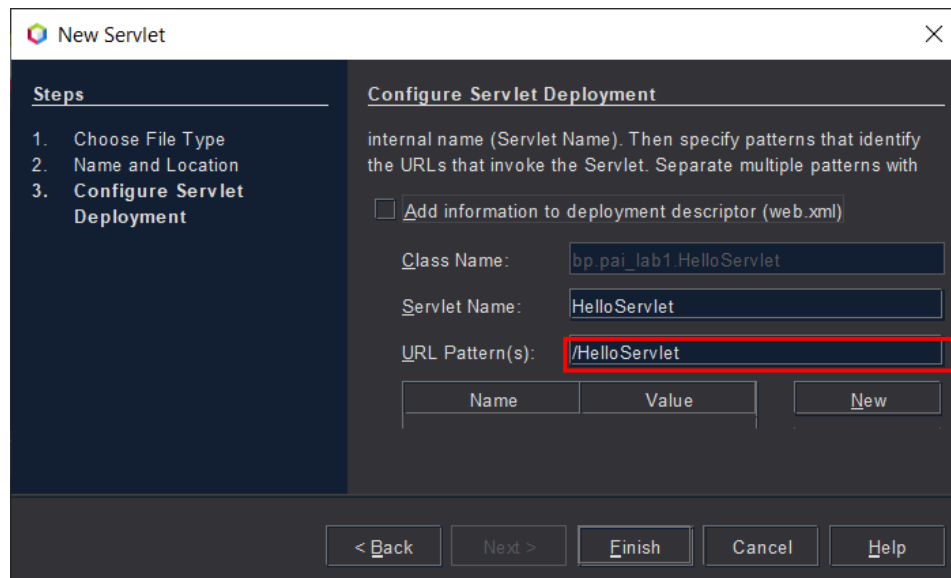
Zadanie 1.1. Pierwszy serwlet

Przygotuj środowisko programistyczne do pracy, jak pokazano na przykładzie *Netbeans 12.6* w rozdziale wstępnym. Utwórz nowy projekt aplikacji webowej o nazwie ***pai_lab1*** (*File* → *New Project* → *Java with Maven/Web Application*). Do projektu dodaj nowy plik z kategorii serwletu o nazwie *HelloServlet* (menu kontekstowe projektu lub *File* → *New File* → *Servlet* – Rys. 1.1). Tworzony serwlet (klasę dziedziczącą po klasie bazowej serwletu *HttpServlet*) dodaj do pakietu (w przykładzie jest to pakiet *bp.pai_lab1*).



Rys. 1.1. Tworzenie serwletu *HelloServlet* jako plik w projekcie

Przy tworzeniu serwletu zwróć uwagę na konfigurację związaną z jego wdrożeniem (Rys. 1.2) i sprawdź ustawienie dla **URLPattern**.



Rys. 1.2. Konfiguracja serwletu

Sprawdź, jak wygląda wygenerowany kod serwletu *HelloServlet* oraz usuń zbędne komentarze. Sprawdź kod pomocniczej metody *processRequest()* oraz rozwiń fragment kodu na końcu serwletu i dokładnie przejrzyj umieszczone tam metody (*doGet()*, *doPost()*, *getServletInfo()*) – Rys. 1.3. **Nie usuwaj ani nie modyfikuj metod *doGet()* i *doPost()*.** Zwróć uwagę, że obie metody wywołują pomocniczą metodę *processRequest()*, wykorzystywanej w kolejnych zadaniach. Parametrami tych metod są dwa ważne obiekty:

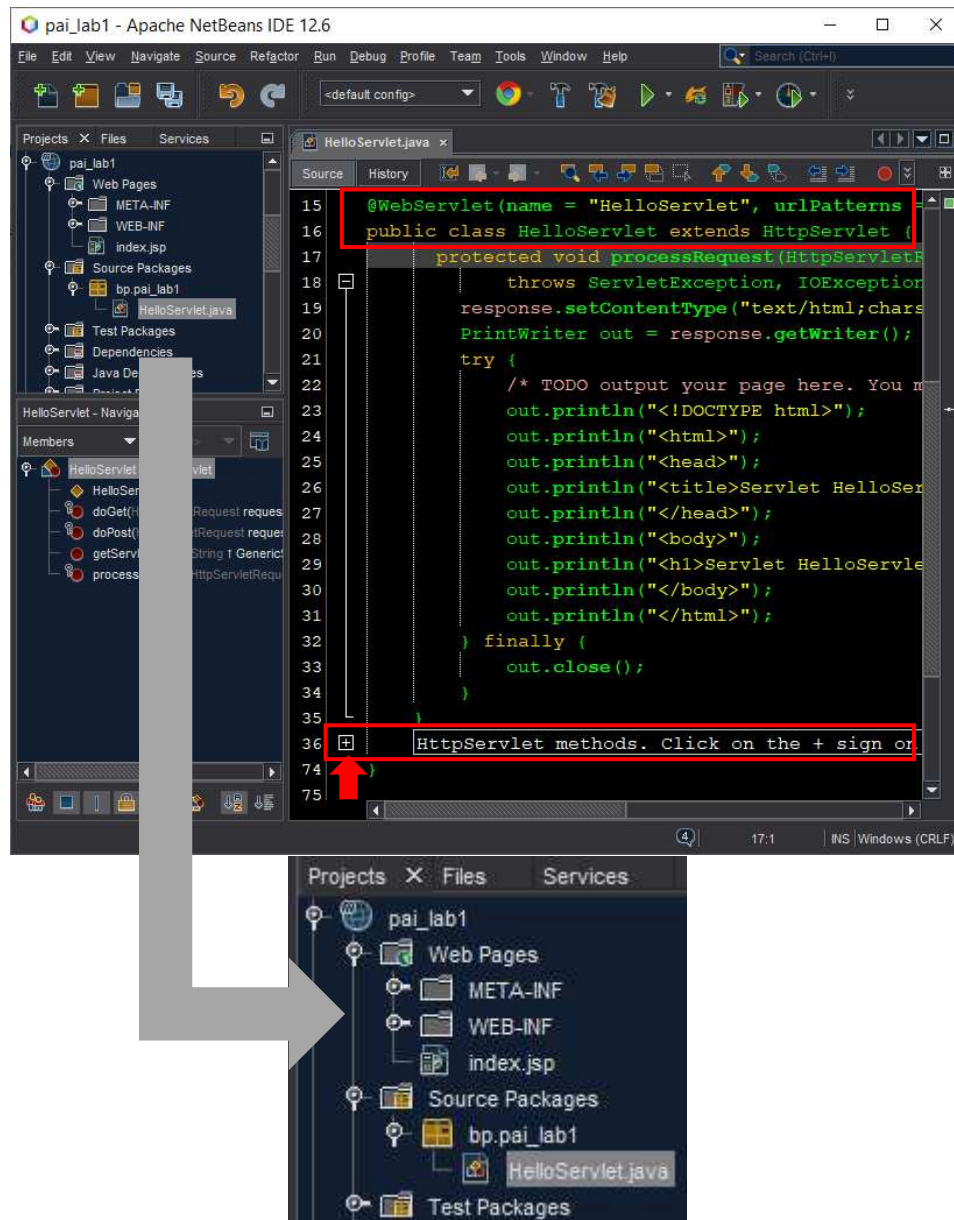
- *request* klasy *HttpServletRequest*, reprezentujący przychodzące żądanie HTTP,
- *response* klasy *HttpServletResponse*, reprezentujący obiekt odpowiedzi HTTP.

Zwróć uwagę na adnotację poprzedzającą definicję klasy serwletu:

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

Na pasku narzędziowym *NetBeans* ustaw wybraną przeglądarkę, a następnie uruchom serwlet *Run*→*Run File*. Sprawdź adres URL w pasku adresowym przeglądarki.

W okienku *Output* kontroluj proces budowania, kompilacji i wdrażania projektu w celu uruchomienia na serwerze *Tomcat*, który wskazano w trakcie tworzenia nowego projektu.



Rys. 1.3. Kod serwletu z pomocniczą metodą *processRequest()*

Zadanie 1.2. Metadane żądania HTTP

Do metody `processRequest()` serwletu *HelloServlet* dopisz (w odpowiednim miejscu w bloku `try`) kod wyświetlający wybrane informacje z obiektu *request* (klasy *HttpServletRequest*), który w serwlecie reprezentuje obiekt żądania HTTP (Przykład 1.1).

Przykład 1.1. Obiekt request i metadane żądania HTTP

```
out.println("<h2>Dane serwera</h2>");
out.println("<p>request.getServerName(): " + request.getServerName() +
    "</p>");
out.println("<p>request.getServerPort(): " + request.getServerPort() +
    "</p>");
out.println("<p>request.getRemoteHost(): " + request.getRemoteHost() +
    "</p>");
out.println("<p>request.getRemoteAddr(): " + request.getRemoteAddr() +
    "</p>");
out.println("<h2>Szczegóły żądania</h2>");
out.println("<p>request.getMethod(): " + request.getMethod() + " </p>");
out.println("<p>request.getQueryString(): " + request.getQueryString() +
    "</p>");
```

Uruchom projekt oraz serwlet wpisując w przeglądarce URL:

http://localhost:8080/pai_lab1/HelloServlet. Przeanalizuj otrzymane wyniki.

Zadanie 1.3. Metoda `init()` w serwlecie

Do klasy serwletu:

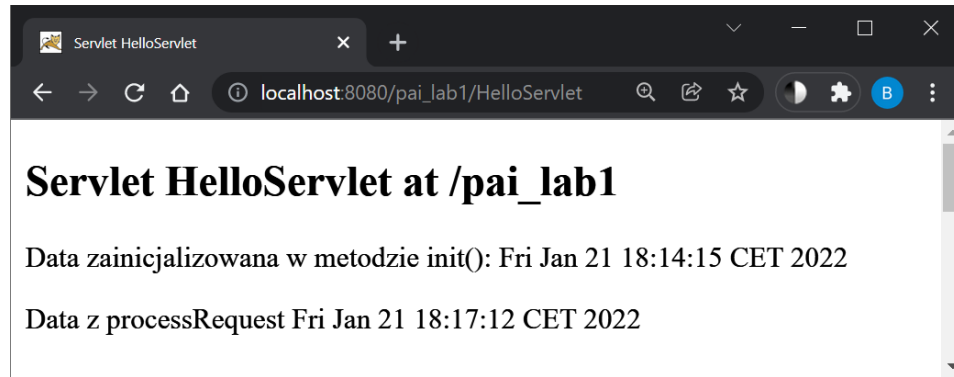
- dodaj deklarację pola *data1* typu *Date* (zaimportuj bibliotekę *java.util.Date*),
- dodaj specjalną (nadpisaną) metodę: `public void init() { }`, w której zainicjalizuj wartość daty: `data1 = new Date();`

Wyświetl wartość *data1* w metodzie `processRequest()`. Uruchom serwlet i sprawdź, czy data zmienia się po odświeżeniu strony w przeglądarce.

Następnie w metodzie `processRequest()` dodaj następujący kod:

```
out.println("<p>data z processRequest " + new Date() + "</p>");
```

Która data zmienia się po ponownym uruchomieniu i odświeżeniu strony w przeglądarce? (Rys. 1.4).



Rys. 1.4. Data z metody *init* i *processRequest* w serwlecie *HelloServlet*

Korzystając z klas *java.text.SimpleDateFormat* i *java.text.DateFormat* przekształć wyświetlaną datę do formatu „yyyy-MM-dd”:

```
DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
Date d = new Date();
// zastosowanie formatu daty:
// dateFormat.format(d)
```

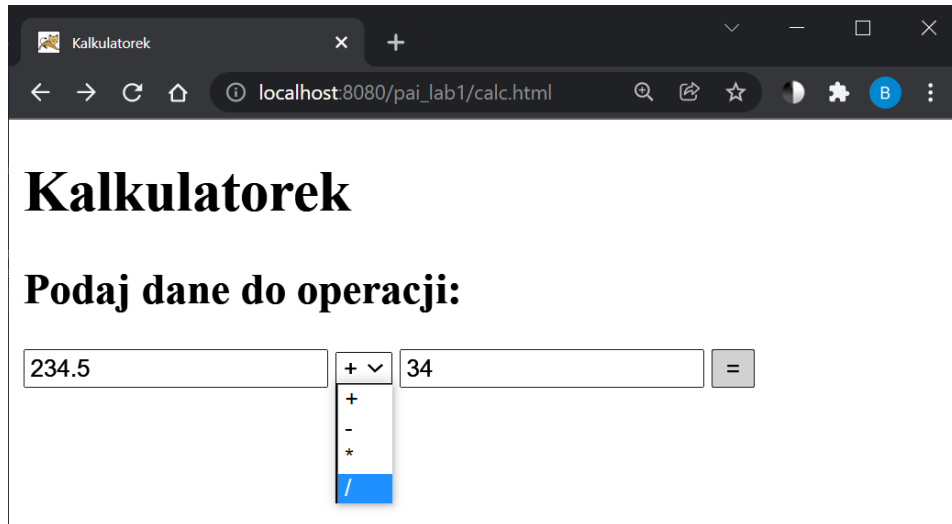
Zadanie 1.4. Dostęp do parametrów żądania

W projekcie utwórz nowy serwlet *CalcServlet*, a następnie statyczną stronę *calc.html* (w głównym folderze *Web Pages* projektu, wybierz kategorię tworzonego pliku *HTML*) z formularzem kalkulatora jak na rysunku 1.5. Formularz powinien być przesyłany za pomocą metody *POST* do serwletu *CalcServlet*:

```
<form method="POST" action="CalcServlet" > ... </form>
```

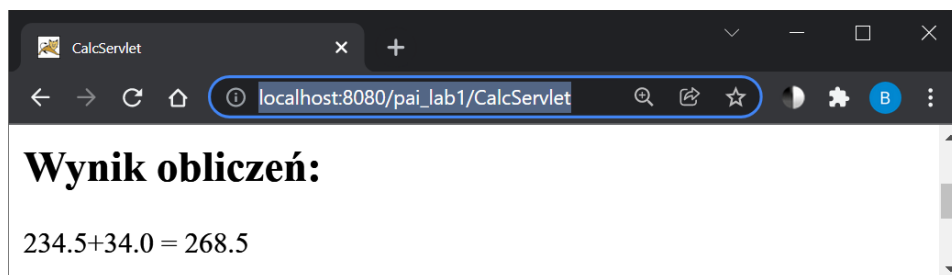
Kliknięcie na przycisk ze znakiem równości ma wysłać żądanie z danymi do serwletu *CalcServlet*, który powinien pobrać wartości parametrów i wyświetlić wynik obliczeń (Rys. 1.6). Do obliczeń wykorzystaj pomocniczą metodę zdefiniowaną w klasie serwletu, np. *oblicz(...)*. Metodzie tej należy przekazać w parametrze obiekt *request*. Pamiętaj, że wartości parametrów w żądaniu są wysyłane jako typ *String*. Wartość parametru przesłanego z danymi z formularza można uzyskać za pomocą metody *getParameter* obiektu *request*:

```
String param=request.getParameter("param");
```



The screenshot shows a web browser window with the title 'Kalkulator'. The address bar shows 'localhost:8080/pai_lab1/calc.html'. The page content includes the heading 'Kalkulator' and the instruction 'Podaj dane do operacji:'. Below this, there are two input fields: the first contains '234.5' and the second contains '34'. Between these fields is a dropdown menu with a '+' icon and a downward arrow, showing a list of operators: '+', '-', '*', and '/'. To the right of the second input field is an '=' button.

Rys. 1.5. Formularz kalkulatora



The screenshot shows a web browser window with the title 'CalcServlet'. The address bar shows 'localhost:8080/pai_lab1/CalcServlet'. The page content includes the heading 'Wynik obliczeń:' and the result '234.5+34.0 = 268.5'.

Rys. 1.6. Wynik działania kalkulatora

Zadanie 1.5. Walidacja danych

Uzupełnij (jeśli jeszcze tego nie ma) serwlet *CalcServlet* tak, aby w przypadku podania niepoprawnego formatu liczby lub w przypadku próby dzielenia przez 0, pojawiał się stosowny komunikat. Walidację zrealizuj po stronie serwletu.

Pamiętaj też, że w wyniku próby pobrania parametru, który nie został przekazany w żądaniu, pojawi się **błąd serwera: *NullPointerException***, dlatego dla każdego parametru należy sprawdzać przynajmniej warunek:

```
if ( (param == null) || (param.trim().equals("")) )
{
    obsługa_Brakujacej_Wartosci_Parametru(...);
}
```

Zadanie 1.6. Mechanizm sesji i ciasteczko

Zmodyfikuj serwlet *CalcServlet* tak, aby wyświetlał historię operacji na kalkulatorze, korzystając z obiektu sesji (w tym celu wygodne jest przechowywanie danych historii w obiekcie *ArrayList*).

Do pracy z sesją w serwlecie skorzystaj z obiektu *HttpSession*:

```
HttpSession session=request.getSession(true);
```

i jego metod *getAttribute* i *setAttribute* (Przykład 1.1).

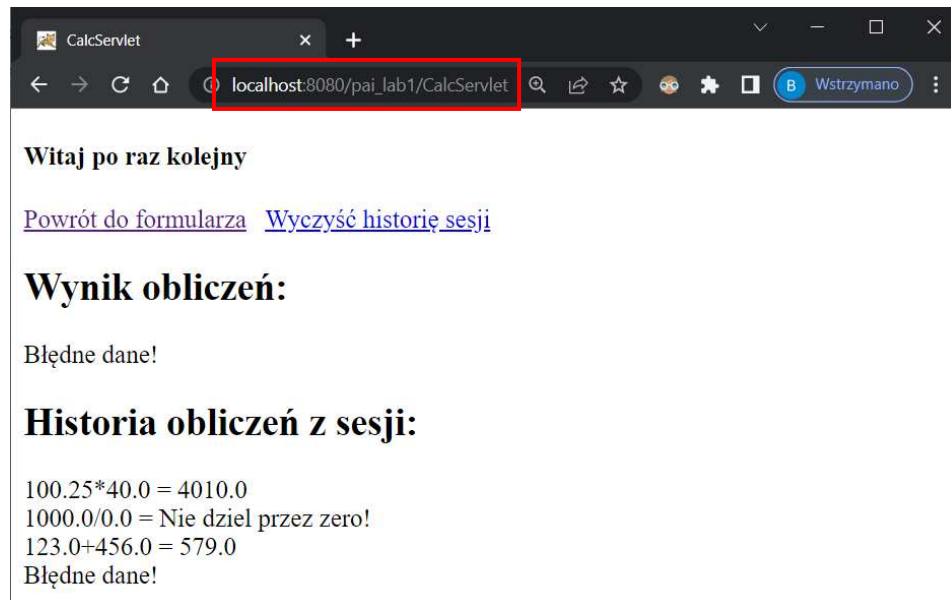
Przykład 1.1. Schemat pracy z obiektem sesji w serwlecie

```
HttpSession sesja = request.getSession(true);
JakasKlasa wartosc = (JakasKlasa) sesja.getAttribute("jakis_id");
if (wartosc==null) { //nie ma szukanego obiektu w sesji
    wartosc = new JakasKlasa(...);
    sesja.setAttribute("jakis_id",wartosc);
}
zrobCosZ(wartosc);
```

Dodaj także dwa linki: powrotny do formularza i do usunięcia operacji z historii (Rys. 1.7). W metodzie *processRequest* wykorzystaj ciasteczko (Przykład 1.2) z powitaniem „Witaj po raz pierwszy” lub „Witaj po raz kolejny” (Rys. 1.7).

Przykład 1.2. Praca z ciasteczkami w serwlecie

```
String nazwaCookie = "UserId";
Cookie [ ] cookies = request.getCookies();
if ( cookies != null )
{
    for (int i=0; i<cookies.lenght; i++) {
        Cookie c=cookies[i];
        if (nazwaCookie.equals(c.getName()))
            jakasOperacjaNaCookie(c.getValue());
    }
}
```

Rys. 1.7. Kalkulator z historią operacji