Programowanie WWW

Servlety

Przypomnienie problemu

- Aplikacja do liczenia kredytów
- Klasa Kredyt
- Formatka do wprowadzania danych (czysty HTML)
- Skrypt liczący ratę (JSP wykorzystujące klasę Kredyt)

Klasa Kredyt

```
public class Kredyt {
    double procent;
    double kwota;
    double lat;

    public double getProcent() {
        return procent;
    }

    public void setProcent(double procent) {
        this.procent = procent;
    }
...
```

Klasa Kredyt cd

```
public double getKwota() { return kwota;}

public void setKwota(double kwota) { this.kwota = kwota; }

public double getLat() { return lat; }

public void setLat(double lat) { this.lat = lat; }

public double getRata() {
    double rata = kwota * (procent/12)/
                  (1-(1/Math.pow(1.0+procent/12,lat*12)));
    return rata;
}
//nie ma setRata() !!!
}
```

Formatka

```
<form action="result.jsp">
kwota: <input type="text" name="kwota"><br/>
ile lat: <input type="text" name="lat"><br/>
procent: <input type="text" name="procent"><br/>
<input type="submit"/>
</form>
```

Skryplet z setterami

```
<%
pl.edu.swsim.Kredyt kredyt = new pl.edu.swsim.Kredyt();
kredyt.setKwota(Double.parseDouble(
                        request.getParameter("kwota")));
kredyt.setLat(Double.parseDouble(
                        request.getParameter("lat")));
kredyt.setProcent(Double.parseDouble(
                        request.getParameter("procent")));
out.write("<p>Rata wynosi: "+kredyt.getRata()+"</p>");
%>
```

JSP z tagami

```
<jsp:useBean id="credit" class="pl.edu.swsim.Kredyt">
 <jsp:setProperty name="credit" property="kwota" param="kwota"/>
 <jsp:setProperty name="credit" property="procent" param="procent"/>
 <jsp:setProperty name="credit" property="lat" param="lat"/>
</jsp:useBean>
```

Rata: <jsp:getProperty name="credit" property="rata"/>

JSP z tagami

```
<jsp:useBean id="credit" class="pl.edu.swsim.Kredyt">
 <jsp:setProperty name="credit" property="kwota" param="kwota"/>
 <jsp:setProperty name="credit" property="procent" param="procent"/>
 <jsp:setProperty name="credit" property="lat" param="lat"/>
</jsp:useBean>
```

Rata: ${credit.rata}

Użycie tagów

- Unikamy kodu w Javie na stronie JSP
- Problem: niektóre rzeczy łatwiej napisać w Javie
- Zalety czystej Javy (względem tagów na stronie JSP)
  - Łatwiejszy dostęp do zasobów
  - Bieżąca kontrola poprawności kodu
  - Łatwiejsze debugowanie
  - Możliwość testowania poza serwerem WWW
  - Prosta obsługa błędów
- Plan na dziś:
  - Tworzenie servletów
  - Połączenie servletu i strony JSP

Servlet

- Obiekt umieszczony w kontenerze servletów i uruchamiany wywołaniem
- Kontener dba o przydział servletów do wywołań (pula servletów)
- Klasy bazowe:
- GenericServlet
  - Metoda service(request, response)
- HttpServlet
  - Metody doXXX(request, response) dla każdego typu wywołania

Tworzenie servletu

- Servlet to klasa dziedzicząca po HttpServlet
- Dwie podstawowe metody:
  - doGet(HttpServletRequest req, HttpServletResponse resp)
  - doPost(HttpServletRequest req, HttpServletResponse resp)
- Wypisanie tekstu HTML:
  - PrintWriter out = response.getWriter();
  - out.println("Hello");

Przykład servletu

```
public class HelloServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req,
                         HttpServletResponse res)
        throws ServletException, IOException {

        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<h1>Hello World!</h1>");
        out.close();
    }
}
```

Struktura aplikacji

- Servlety w WEB-INF/classes
- Deskryptor aplikacji
  - WEB-INF/web.xml
- Definicja jakie są servlety i do jakich adresów URL aplikacji są przypięte

Najprostszy web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-
     app_2_4.xsd"
   version="2.4">
<!-- definicja servletu -->
   <servlet>
      <servlet-name>Hello</servlet-name>
      <servlet-class>pl.kurs.HelloServlet</servlet-class>
   </servlet>
<!-- mapowanie servletu -->
   <servlet-mapping>
      <servlet-name>Hello</servlet-name>
      <url-pattern>/hello</url-pattern>
   </servlet-mapping>
</web-app>
```

ćwiczenie

Stworzenie servletu

- Tworzymy bezpośrednio kod servletu zamiast strony JSP

```
public class MyController extends HttpServlet {
public void doGet(HttpServletRequest request,
    HttpServletResponse response) throws IOException,
    ServletException {
        response.getWriter().println("Hello from GET");
}
public void doPost(HttpServletRequest request,
    HttpServletResponse response) throws IOException,
    ServletException {
        response.getWriter().println("Hello from POST");
}
```

Dostęp do zasobów

- getServletContext()
  - Wspólny kontekst servletów (poziom aplikacji)
- request.getSession()
  - Sesja
- request
  - Żądanie
- Zmienne lokalne
  - Poziom strony (page)

Przekierowania

- Servlet może przekierować żądanie dalej
  - Do następnego serwletu
  - Do strony JSP
- Do znalezienia servletu używany jest RequestDispatcher
  - getServletContext().getRequestDispatcher("adres")
- Metody:
  - forward(request,response)
  - include(request,response)
- Kolejny servlet używa tego samego requesta
  - W ten sposób można zrealizować komunikację

Tworzenie łańcucha servletów

- Servlet może pobrać z kontekstu obiekt RequestDispatcher:
  - getServletContext().getRequestDispatcher("url")
- Dwie metody:
  - forward(request, response)
  - include(request, response)
- Przekierowanie wywołania:
  - getServletContext().getRequestDispatcher("/form.jsp")
    .forward(request, response);
- Przed przekierowaniem można umieścić dane w request:
  - request.setAttribute("klucz",dowolnaDana);

Przykład

- Servlet TimeTable
- Servlet Time używane kilkukrotnie z różnymi parametrami

Servlet TimeTable

```
PrintWriter out = res.getWriter();
out.print("Lokalny czas: ");
getServletContext().getRequestDispatcher("/time")
    .include(req,res);
out.print("Czas w Londynie: ");
request.setAttribute("zone", "GMT");
getServletContext().getRequestDispatcher("/time")
    .include(req,res);
out.print("Czas w Tokyo: ");
request.setAttribute("zone", "Japan");
getServletContext().getRequestDispatcher("/time")
    .include(req,res);
```

Servlet Time

```
PrintWriter out = res.getWriter();
Date date = new Date();
DateFormat df = DateFormat.getInstance();
String zone = (String) request.getAttribute("zone");
if (zone != null) {
    TimeZone tz = TimeZone.getTimeZone(zone);
    df.setTimeZone(tz);
}
out.println(df.format(date));
```

ćwiczenie

Przekierowanie do JSP

- Servlet wykonuje pracę i wynik umieszcza w request
- Następnie przekierowuje sterowanie do strony JSP
  - getServletContext().getRequestDispatcher("/page.jsp")
    .forward(request,response);
- Strona JSP wyświetla wynik

Przykład - servlet

```
public class MyServlet extends HttpServlet {
public void doGet(
    HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException {
    String naglowek = "Nagłówek pliku";
    String imie = "Andrzej";
    String komunikat = "Proszę się zalogować!";

    request.setAttribute("header", naglowek);
    request.setAttribute("name", imie);
    request.setAttribute("info",komunkat);

    getServletContext().getRequestDispatcher("/strona.jsp")
            .forward(request,response);
}
```

Przykład - jsp

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

<h1>${header}</h1>

<p>Witaj ${name}!</p>

Informacja dla Ciebie: ${info}
```

ćwiczenie

Podsumowanie

- Serwlet
  - Dostaje wszystkie wywołania
  - Sprawdza parametry
  - Przygotowuje dane
  - Umieszcza w *request* lub *session*
- Strona JSP
  - Odczytuje informacje z *request* i *session*
  - Wyświetla je na ekranie
- Co dalej:
  - Logikę (obliczenia) wydobywamy z serwletu