Сервлети

У овој вежби су приказани основни принципи писања и употребе сервлета у веб апликацијама.

За тестирање примера треба креирати веб апликацију **learningservlets** на Tomcat веб серверу. Контекст апликације треба додати у конфигурциону датотеку **server.xml**:

```
<Context path="/learningservlets" docBase="learningservlets"
      debug="0" reloadable="true"/>
```

У фолдеру који се додељује веб апликацији треба обезбедити типичну структуру фолдера и датотека, при чему треба креирати датотеку **web.xml**, са следећим почетним садржајем:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID"
version="3.1">

  <display-name> learningservlets</display-name>
  <welcome-file-list>
    <welcome-file> index.html </welcome-file>
  </welcome-file-list>

</web-app>
```

Сваки сервлет који буде написан ће бити додат у датотеку **web.xml**, да би могао да се изврши на веб серверу.

Датотека **index.html** ће бити коришћена за преглед садржаја лекције и позивање сервлета помоћу линкова. Стил приказа веб странице је уређен у датотеци **stilovi.css**. Веб апликација је доступна на адреси:

http://localhost/learningservlets/

НАПОМЕНА: У зависности од подешавања HTTP конекције не веб серверу треба у адресу веб апликације додати порт 8080.

ПРИМЕР (learningservlets)

Основна структура сервлета

Креирати сервлет са најједноставнијом структуром који приказује текстуалну поруку (**`SimpleServlet`**) . Сервлет текстуалну поруку приказује помоћу методе **`doGet()`**. Изворни код основног сервлета је:

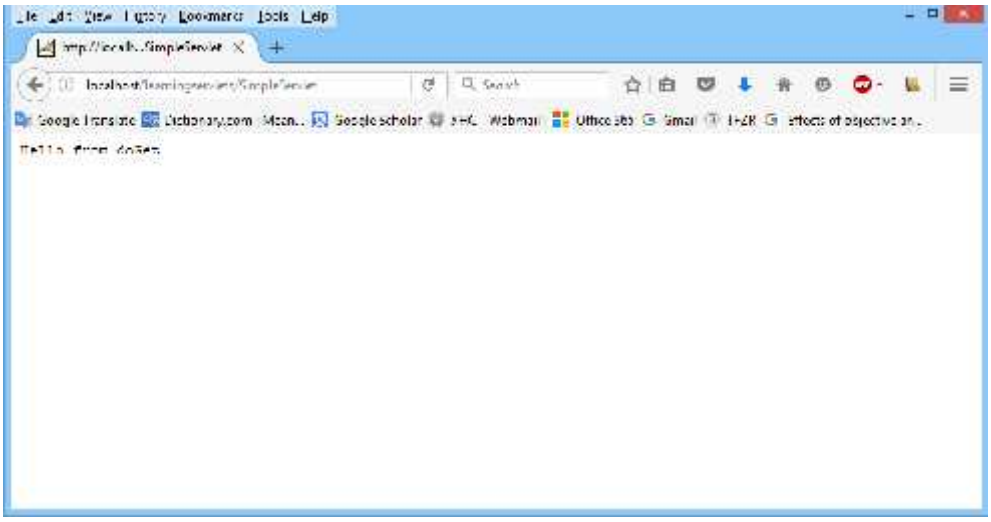
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class SimpleServlet extends HttpServlet {

      public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
                  throws ServletException, IOException {
            // print simple text
            PrintWriter out = response.getWriter();
            out.println("Hello from doGet");
      }

      protected void doPost(HttpServletRequest request,
                  HttpServletResponse response)
                  throws ServletException, IOException {
            // call doGet
            doGet(request, response);
      }
}
```

Измена датотеке **web.xml** подразумева додавање информација о сервлету као што је приказано у следећем листингу:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID"
version="3.1">

  <display-name> learningservlets</display-name>
  <welcome-file-list>
    <welcome-file> index.html </welcome-file>
  </welcome-file-list>

  <servlet>
      <servlet-name>SimpleServlet</servlet-name>
      <servlet-class>SimpleServlet</servlet-class>
  </servlet>
  <servlet-mapping>
      <servlet-name>SimpleServlet</servlet-name>
      <url-pattern>/SimpleServlet</url-pattern>
  </servlet-mapping>

</web-app>
```

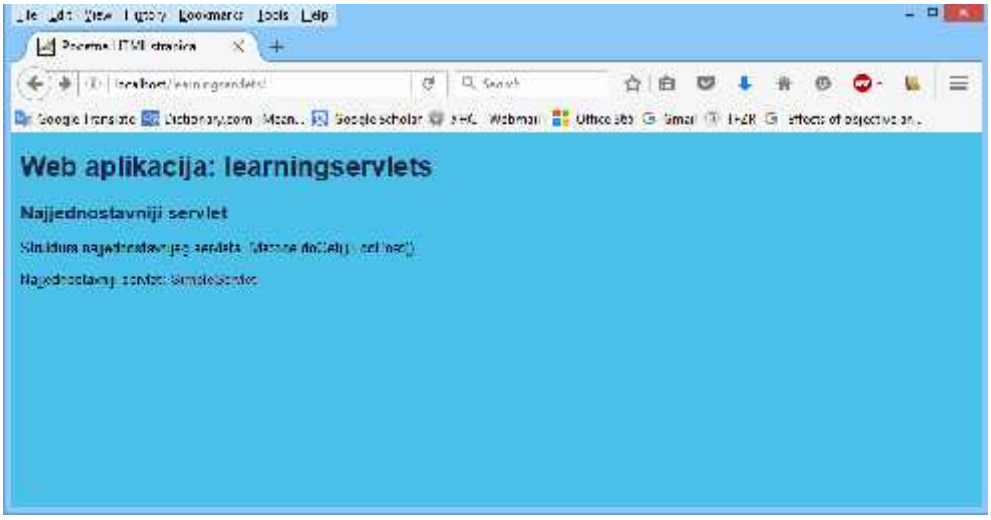
Извршавање сервлета у веб претраживачу се може реализовати директним уносом адресе сервлета у адресну линију претраживача

http://localhost/learningservlets/SimpleServlet



или позивом помоћу линка из веб странице

SimpleServlet



Сервлет који генерише HTML

Сервлети најчешће генеришу HTML садржај, што подразумева да се веб претраживач обавести да ће бити послат HTML документ, затим да се модификује штампање излаза помоћу **`PrintWriter`**, и на крају треба проверити валидност генерисаног HTML садржаја.

Да би се веб претраживач обавестио да ће бити послат HTML садржај треба поставити у заглављу одзива (**`response`**) одговарајући тип, у овом случају за генерисање HTML5 документа.

```
response.setContentType("text/html");
```

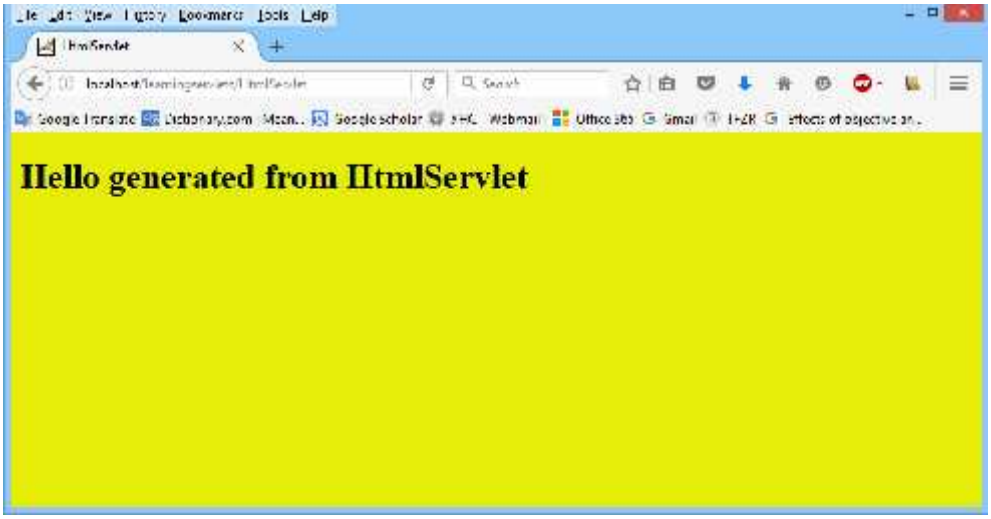
Ово се може реализовати и позивом методе **`setHeader`** класе **`HttpServletResponse`**, али се с обзиром на уобичајеност посла то може урадити директно у сервлету позивом **`setContentType`** методе која исписује тип садржаја **`text/html`**.

Након креирања сервлет класе, сервлет треба пријавити и у **web.xml** датотеци додавањем следећег кода:

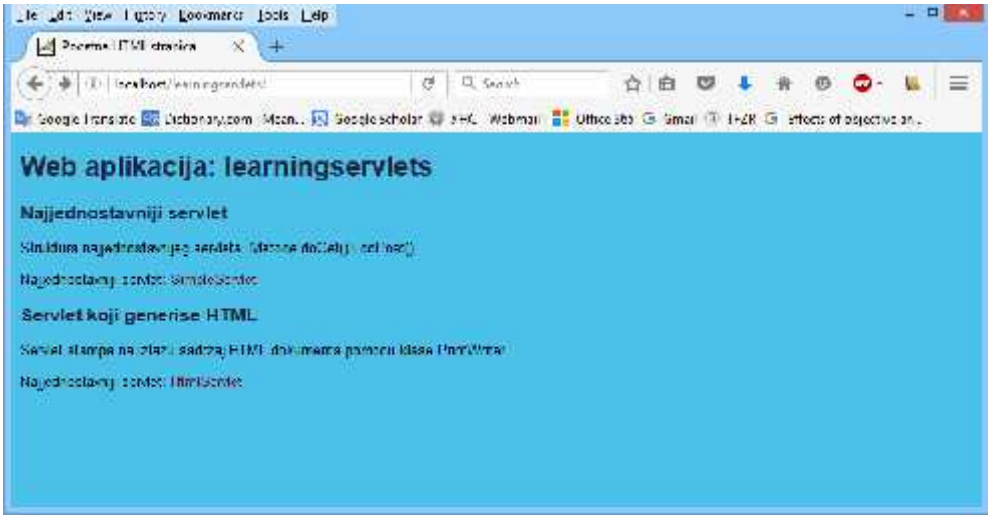
```
<servlet>
    <servlet-name>HtmlServlet</servlet-name>
    <servlet-class>HtmlServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>HtmlServlet</servlet-name>
    <url-pattern>/HtmlServlet</url-pattern>
</servlet-mapping>
```

Извршавањем сервлета се добија форматирана HTML страница

http://localhost/learningservlets/HtmlServlet



Сервлет се такође може покренути помоћу линка из веб странице:



Организација сервлета у пакете

Писање већих апликација подразумева већи броој класа и сервлета које тада треба организовати у пакете ради лакшег организовања развоја и одржавања софтвера. Сервлете је као и све остале Јава класе могуће организовати у пакете који одсликавају логичку структуру апликације.

Приликом организовања сервлета у пакете потребно је урадити следеће:

- Рапоредити сервлете у поддиректоријуме који одговарају називима пакета.
- У датотеку са изворним кодом сервлета треба додати наредбу која означава пакет у којем се налази.
- У **web.xml** датотеци треба нагласити да се сервлет налази у пакет.

Пример кода сервлета који је распоређен у пакету **`hello`**. Назив пакета се налази на почетку датотеке, као и код било које друге Јава класе.

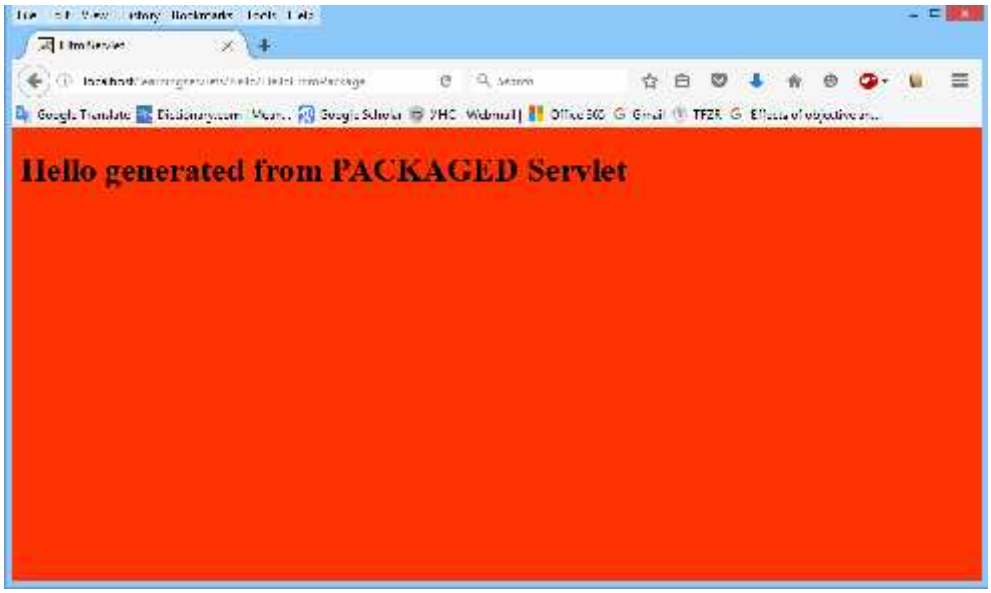
```
package hello;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/HelloFromPackage")
public class HelloFromPackage extends HttpServlet {
      private static final long serialVersionUID = 1L;

      public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
                              throws ServletException, IOException {
            // declare sending HTML content to web browser
            response.setContentType("text/html");
            // print HTML content
            PrintWriter out = response.getWriter();
          String docType = "<!DOCTYPE html>\n";
          out.println(docType +
                      "<HTML>\n" +
                      "<HEAD><TITLE>HtmlServlet</TITLE></HEAD>\n" +
                      "<BODY BGCOLOR=\"#ff3300\">\n" +
                      "<H1>Hello generated from PACKAGED Servlet</H1>\n" +
                      "</BODY></HTML>");
      }

      protected void doPost(HttpServletRequest request,
                        HttpServletResponse response)
                              throws ServletException, IOException {
            // call doGet
            doGet(request, response);
      }

}
```

Додавање информације о сервлету у **web.xml** датотеци:

```
<servlet>
      <servlet-name>HelloFromPackage</servlet-name>
      <servlet-class>hello.HelloFromPackage</servlet-class>
</servlet>
<servlet-mapping>
      <servlet-name>HelloFromPackage</servlet-name>
      <url-pattern>/hello/HelloFromPackage</url-pattern>
</servlet-mapping>
```

Извршавањем сервлета се добија форматирана HTML страница

http://localhost/learningservlets/hello/HelloFromPackage



Сервлет се такође може покренути помоћу линка из веб странице

ServletFromPackage

Креирање услужних класа за рад са HTML кодом

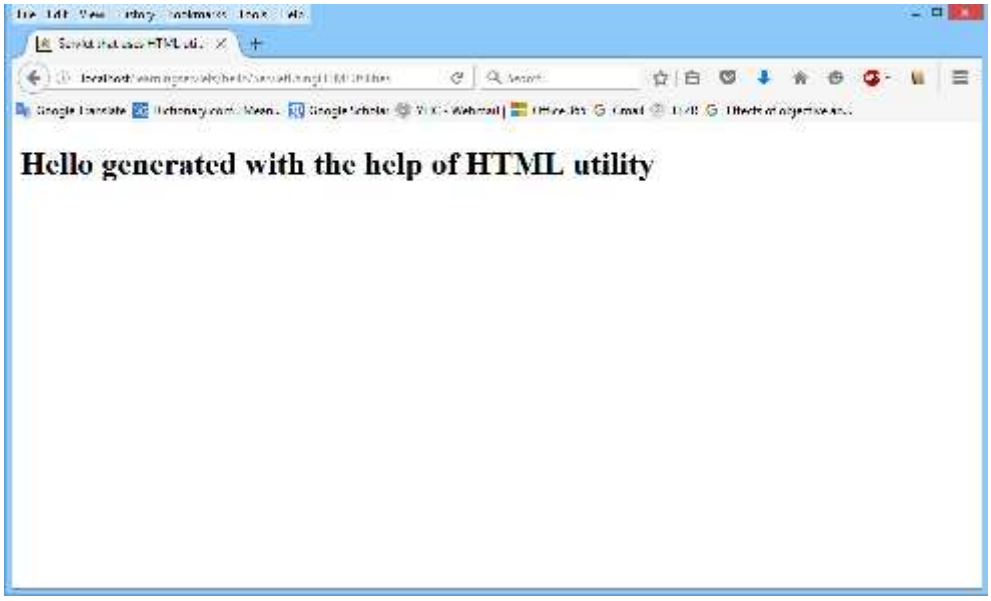
Када се део кода често понавља, што јесте случај када се генерише HTML садржај, тај део кода треба реализовати помоћу услужних класа које поједностављују креирање HTML садржаја и писање и сервлетских класа. Поред тога сервлетске класе имају значајно мање кода па је сам код једноставнији за одржавање.

Код сервлета се приликом креирања веб страница делови **DOCTYPE** и **HEAD** ретко мењају, па се њихово штампање може реализовати у услужној класи коју потом могу позивати сви сервлети у оквиру веб апликације. У овом примеру је креирана класа **`ServletHtmlUtilities`** која омогућује штампање HTML заглавља за неколико различитих типова HTML докумената. Услужне класе пошто нису сервлети се не морају пријавити у **web.xml** датотеци. Сервлет **`ServletUsingHTMLUtilities`** који користи услужну класу треба пријавити у **web.xml** датотеци.

```
<servlet>
      <servlet-name>ServletUsingHTMLUtilities</servlet-name>
      <servlet-class>hello.ServletUsingHTMLUtilities</servlet-class>
 </servlet>
<servlet-mapping>
      <servlet-name>ServletUsingHTMLUtilities</servlet-name>
      <url-pattern>/hello/ServletUsingHTMLUtilities</url-pattern>
</servlet-mapping>
```

Извршавањем сервлета се добија форматирана HTML страница

http://localhost/learningservlets/hello/ServletUsingHTMLUtilities



Сервлет се такође може покренути помоћу линка из веб странице

ServletUsingHTMLUtilities

Животни циклус сервлета

На веб серверу се увек креира само једна инстанца сваког сервлета, док се кориснички захтеви (*user requests*) реализују тако да се сваки позив метода **`doGet`** или **`doPost`** реализује као засебна нит.

Приликом првог позива, сервлет се креира и позива се метода **`init`** која се користи за конфигурисање сервлета. Сви остали позиви сервлета се усмеравају на методу **`service`** која позиве усмерава на методе **`doGet`**, **`doPost`**, или остаале **`doXxx`** (**`doPut`**, **`doDelete`**) методе. Када сервлет више није потребан контејнер позива методу **`destroy`** која уништава сервлет. Метода за иницијализацију се користи за иницијализацију применљивих, учитавање датотека, успостављање конекције са базом података итд., док се метода за уништавање сервлета користи за затварање датотека, затварање конекција ка базама података итд.

НАПОМЕНА: Препорука је да се методе животног цикуса **`init`** и **`destroy`** не позиваху из кода, већ да се у њима изврши минимални посао иницијализације или уклањања објеката. За такве послове је увек боље креирати специјалне функције у којима се има потпуна контрола кода, док методама **`init`** и **`destroy`** управља контејнер сервлета (нпр. Tomcat).

У примеру који илуструје употребу методе **`init`**, у сервлету **`ServletLifeCycleMethods`** се иницијлизују променљиве које служе за праћење приступа сервлету:

```
private static int accessCounter = 0;
private static boolean initialized = false;
```

Помоћу ових статичких промнљивих се потом управља радом методе **`doGet`** која је задужена за руковање HTTP захтевима (*requests*) и одзивима (*responses*).

Пре употребе, сервлета **`ServletLifeCycleMethods`** треба пријавити у **web.xml** датотеци.

```
<servlet>
      <servlet-name>ServletLifeCycleMethods</servlet-name>
      <servlet-class>servletlife.ServletLifeCycleMethods</servlet-class>
</servlet>
<servlet-mapping>
      <servlet-name>ServletLifeCycleMethods</servlet-name>
      <url-pattern>/servletlife/ServletLifeCycleMethods</url-pattern>
 </servlet-mapping>
```

Важно је напоменути да се сваки приказ извршавања сервлета реализује у **`doGet`**, па и одмах након саме иницијализације сервлета методом **`init`** која садржи код који не производи приказ на екрану.

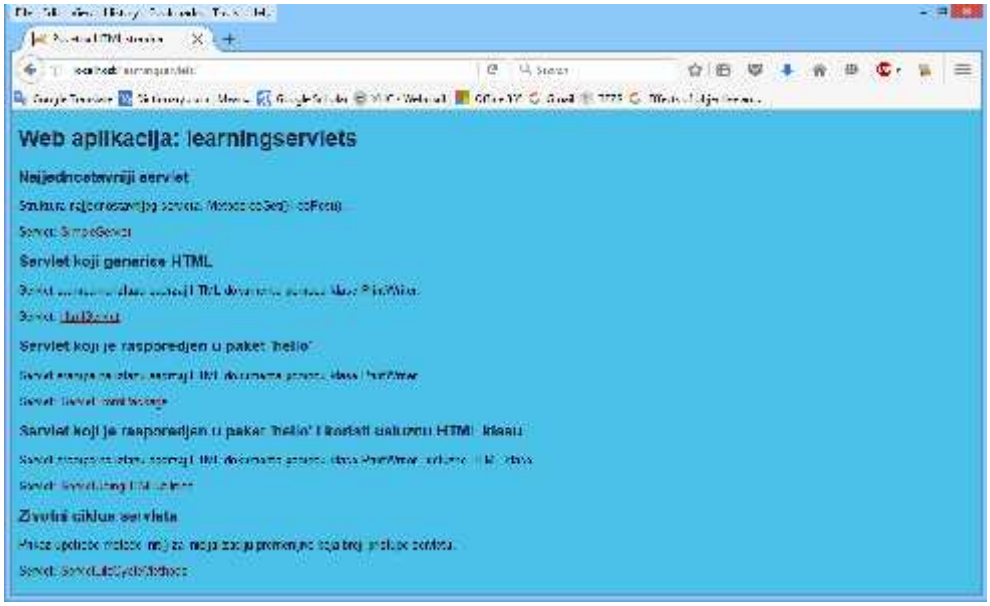
Пример вишеструког позивања сервлета **`ServletLifeCycleMethods`** након иницијализације, тј. меоде **`doGet`** је приказан на следећим сликама, где се може пратити иницијализација и бројање приступ сервлету (најједноставније је остварити вишеструки узастопни приступ освежавањем веб странице коју генерише сервлет).

Извршавањем сервлета се добија форматирана HTML страница

http://localhost/learningservlets/servletlife/ServletLifeCycleMethods

Сервлет се такође може покренути помоћу линка из веб странице

ServletLifeCycleMethods



Задаци за самостални рад

Задатак 1

Написати и тестирати сервлет који приказује све елементе Фибоначијевог низа који су мањи од 1000. приказ Фибоначијевог низа форматирати као HTML страницу на коју се могу применити стилови за форматирање садржаја (CSS).

Задатак 2

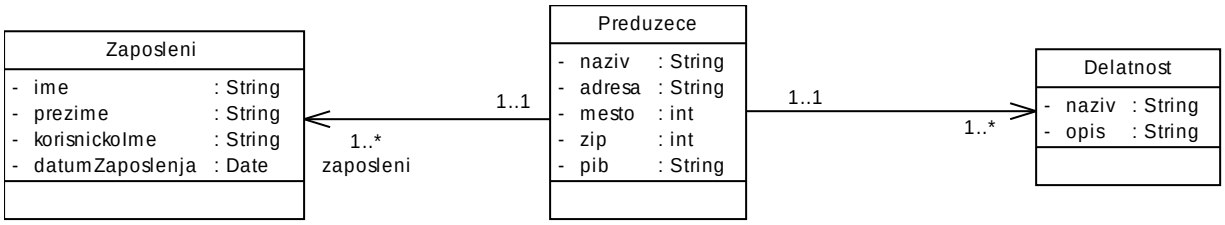
Написати и тестирати сервлет који приказује полазни низ целих бројева и потом сортирани низ у растућем редолседу. Приказ полазног и сортираног низа форматирати као HTML страницу на коју се могу применити стилови за форматирање садржаја (CSS).

Задатак 3

Проширити класу **`ServletHtmlUtilities`** тако да омогућује додавање линка **ПОВРАТАК** (*Back*) на дну сваке веб странице коју креира сервлет тако да омогућује пвратак на полазну страницу веб апликације **index.html**. Класу тестирати на неколико већ урађених сервлета.

Задатак 4

Написати и тестирати сервлет који приказује листу запослених у предузећу. Запослени је представљен класом **`Zaposleni`** која је приказана на слици:



У горњем делу странице (заглављу) формиране веб странице, пре приказа података о запосленима, треба приказати податке о предузећу које је представљено класом **`Preduzece`**.

Линкови

[1] **The Apache Tomcat**. http://tomcat.apache.org/

[2] Marty Hall and Larry Brown. *Core Servlets and JavaServer Pages, Free Online Version of Second Edition*. http://pdf.coreservlets.com/

[3] https://tomcat.apache.org/tomcat-9.0-doc/servletapi/overview-summary.html

[4] https://tomcat.apache.org/tomcat-9.0-doc/servletapi/index.html.