Dodatni Materijal

Korišćenje *third party* biblioteka koje omogućuju logovanje – Apache Log4j

- Biblioteka za logovanje (zapisivanje dnevnika)
- Open source
- http://logging.apache.org/log4j/2.x/
- Ispis bitnih stvari u konzolu/fajl/nešto drugo
- Dodavanje u projekat
  - log4j api
  - log4j core

Korišćenje *third party* biblioteka koje omogućuju logovanje – Apache Log4j

- Osnovna upotreba:

```
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
public class HelloWorld {
  private static Logger logger =
     LogManager.getLogger("HelloWorld");
  public static void main(String[] args) {
    logger.error("Hello, World!");
  }
}
```

Korišćenje *third party* biblioteka koje omogućuju logovanje – Apache Log4j

- Moguće je postaviti različite nivoe za lovovanje:
  - off – nema logovanja (0)
  - fatal – greška, ali će aplikacija najverovatnije stati (100)
  - error – greška, ali aplikacija bi trebalo da nastavi (200)
  - warn – upozorenje (300)
  - info – informacija (400)
  - debug – debagiranje (500)
  - trace – debagiranje, finije od debug (600)
  - all – sve se loguje (Integer.MAX_VALUE)

Korišćenje *third party* biblioteka koje omogućuju logovanje – Apache Log4j

- Moguće je konfigururisati rad biblioteke na više načina:
  - Logger traži konfiguracioni fajl u "log4j.configurationFile" environment varijabli
  - zatim traži "log4j2-test.json" ili "log4j2-test.jsn" datoteku
  - zatim traži "log4j2-test.xml" datoteku
  - zatim traži "log4j2.json" ili "log4j2.jsn" datoteku
  - zatim traži "log4j2.xml" datoteku, i, na kraju
  - koristi klasu DefaultConfiguration

Korišćenje *third party* biblioteka koje omogućuju logovanje – Apache Log4j

- Primer log4j2.xml datoteke:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
  <Appenders>
    <Console name="Console" target="SYSTEM_OUT">
      <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-
   5level %logger{36} - %msg%n" />
    </Console>
  </Appenders>
  <Loggers>
    <Root level="all">
      <AppenderRef ref="Console" />
    </Root>
  </Loggers>
</Configuration>
```

Korišćenje *third party* biblioteka koje omogućuju logovanje – Apache Log4j

- Doda se File appender:

```
<Appenders>
  <Console name="Console" target="SYSTEM_OUT">
    <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-
   5level %logger{36} - %msg%n" />
  </Console>

  <File name="MyFile" fileName="./logs/app.log">
    <PatternLayout>
      <Pattern>%d %p %c{1} [%t] %m%n</Pattern>
    </PatternLayout>
  </File>
</Appenders>
```

Korišćenje *third party* biblioteka koje omogućuju logovanje – Apache Log4j

- Primer:

```
<PatternLayout pattern="
%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n
" />
```

- Neki najčešći parametri:
  - %d – datum i/ili vreme
  - %t – ime niti (thread) iz iz koje se loguje
  - %logger (ili %c) – klasa koja se loguje
  - %c{1} – samo ime klase, bez imena paketa
  - %msg (ili %m) – poruka
  - %n – novi red

primerDodatnoLogovanje

Korišćenje *third party* biblioteka koje omogućuju čitanje i pisanje u Excel datoteke – *Apache POI*

- Biblioteka za rad sa Microsoft Office dokumentima
- Open source
- http://poi.apache.org/
- Čitanje i pisanje MSOffice dokumenata
- Dodavanje u projekat:
  - poi-3.10-FINAL-20140208.jar
  - poi-ooxml-3.10-FINAL-20140208.jar
  - poi-ooxml-schemas-3.10-FINAL-20140208.jar
  - xmlbeans-2.3.0.jar
  - dom4j-1.6.1.jar
  - po potrebi dodati još...

Korišćenje *third party* biblioteka koje omogućuju čitanje i pisanje u Excel datoteke – *Apache POI*

- Osnovna klasa je Workbook
- Dobija se iz fabrike:

```
Workbook wb = WorkbookFactory.create(new
    FileInputStream("racuni.xlsx"));
Workbook wb = WorkbookFactory.create(new
    File("racuni.xlsx"));
```

- Kreiranje novog/overwrite postojećeg Excel fajla

```
Workbook wb = new XSSFWorkbook();
FileOutputStream fileOut = new
    FileOutputStream("racuni.xlsx");
wb.write(fileOut);
fileOut.close();
```

Korišćenje *third party* biblioteka koje omogućuju čitanje i pisanje u Excel datoteke – *Apache POI*

- Rad sa listovima
  - Kreiranje:

```
Sheet sheet = wb.createSheet("new sheet");
```

  - Pristup:

```
//na osnovu indeksa u nizu listova, ideksi počinju od 0
Sheet sheet = wb.getSheetAt(0);
//ili na osnovu imena
wb.getSheet("Sheet1")
```

Korišćenje *third party* biblioteka koje omogućuju čitanje i pisanje u Excel datoteke – *Apache POI*

- Rad sa redovima
  - Kreiranje:

```
Row row = sheet.createRow(0);
```

  - Pristup:

```
//na foreach petlje
 for (Row row : sheet) {
   // izbegnemo prvu vrstu (zaglavlje)
   if (row.getRowNum() == 0)
     continue;
   ...
 }
//ili na osnovu pozicije
 Row row = sheet.getRow(i);
```

Korišćenje *third party* biblioteka koje omogućuju čitanje i pisanje u Excel datoteke – *Apache POI*

- Rad sa ćelijama koje se nalaze u određenoj koloni
  - Kreiranje:

```
Cell cell = row.createCell(0);
```

  - Pristup:

```
//na foreach petlje
for (Cell cell : row) ...
//ili na osnovu pozicije
r = row.getCell(i);
d = row.getCell(i, Row.RETURN_BLANK_AS_NULL);
```

  - Čitanje:

```
d = ocena.getNumericCellValue();
s = grad.getStringCellValue();
```

  - Upis nove vrednosti:

```
cell.setCellValue(1);
```

Korišćenje *third party* biblioteka koje omogućuju čitanje i pisanje u Excel datoteke – *Apache POI*

- Brisanje reda

```
/** Remove a row by its index
* @param sheet a Excel sheet
* @param rowIndex a 0 based index of removing row */
public static void removeRow(HSSFSheet sheet, int
rowIndex) {
  int lastRowNum=sheet.getLastRowNum();
  if(rowIndex>=0 && rowIndex<lastRowNum){
    sheet.shiftRows(rowIndex+1,lastRowNum, -1);
  }
  if(rowIndex==lastRowNum){
    HSSFRow removingRow=sheet.getRow(rowIndex);
    if(removingRow!=null){
      sheet.removeRow(removingRow);
    }
  }
}
```

primerDodatnoEksel