

# Iznimke u Javi

---

AUDITORNE VJEŽBE

# Sadržaj

---

Ispis staze stoga iznimke

Zapisivanje podataka o iznimkama u log datoteke – LogBack

Apache Maven

Primjer zadatka s iznimkama – pogađanje brojeva

Pitanja s certifikata

# Ispis staze stoga iznimke

---

- Staza stoga iznimke (engl. *stack trace*) sadržava ključne informacije o razlogu bacanja neke iznimke i često se ispisuje u konzolu razvojnog okruženja
- Ispis staze stoga u konzolu moguće je obaviti i pozivom metode „`printStackTrace`” iz objekta koji predstavlja iznimku, a najčešće se koristi unutar „`catch`” bloka
- Osim ispisa staze stoga, iz objekta koji predstavlja iznimku moguće je dohvatiti i poruku koja detaljnije opisuje razlog nastanka iznimke korištenjem metode „`getMessage`”:

```
catch(ArithmeticException ex1) {  
    ex1.printStackTrace();  
    String poruka = ex1.getMessage();  
    ...  
}
```

[java.lang.ArithmeticException: / by zero](#)  
/ by zero

# Zapisivanje podataka o iznimkama u log datoteke - LogBack

---

- Umjesto da se informacije o iznimkama zapisuju u konzolu koja ne sprema podatke, iste je moguće zapisivati u log datoteke
- Kako je to česta praksa radi omogućavanja naknadnog analiziranja uzroka problema u radu aplikacije, za to se koristi vanjska biblioteka LogBack
- Ona omogućava da se na jednostavan način konfiguriraju detalji koji se zapisuju u log datoteke i obavi samo zapisivanje
- Osim pogrešaka u log datoteke je moguće zapisivati i informativne poruke koje dokumentiraju aktivnosti korisnika u aplikaciji, sve u svrhu lakše rekonstrukcije i reproduciranja problema, a samim time i njihovog ispravljanja
- Biblioteka LogBack se konfigurira korištenjem XML datoteke koja se mora nalaziti unutar projekta koji koristi „logiranje”

# Zapisivanje podataka o iznimkama u log datoteke - LogBack

---

- Primjer te XML datoteke izgleda ovako:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <appender name="FILE" class="ch.qos.logback.core.FileAppender">
    <file>logs/pogreske.log</file>
    <encoder>
      <pattern>%date %level [%thread] %logger{10} [%file:%line] %msg%n</pattern>
    </encoder>
  </appender>
  <root level="debug">
    <appender-ref ref="FILE" />
  </root>
</configuration>
```

# Zapisivanje podataka o iznimkama u log datoteke - LogBack

---

- LogBack omogućava kreiranja nekoliko razina log zapisa u ovisnosti o njihovoj važnosti i detaljnosti:
  - ERROR
  - WARN
  - INFO
  - DEBUG
  - TRACE
- Java naredbe koje omogućavaju zapisivanje informacija u log datoteke ovise o „razini logiranja” što se očituje u nazivu metode koja se poziva, npr.:  
*`Logger.error("Došlo je do pogreške u radu aplikacije!", ex);`*
- U metodu za „logiranje” se često predaje i objekt koji predstavlja iznimku

# Zapisivanje podataka o iznimkama u log datoteke - LogBack

---

- Primjer sadržaja log datoteke:

```
2014-10-18 18:57:55,482 INFO [main] h.t.j.v.g.Glavna [Glavna.java:23] Unos prve knjige
2014-10-18 18:58:42,194 INFO [main] h.t.j.v.g.Glavna [Glavna.java:26] Unos druge knjige
2014-10-18 18:59:13,656 ERROR [main] h.t.j.v.g.Glavna [Glavna.java:33] Pogreška prilikom unosa knjige!
hr.tvz.java.vjezbe.iznimke.DuplikatPublikacijeException: Publikacija već postoji!
at hr.tvz.java.vjezbe.glavna.Utils.provjeriDuplikate(Utils.java:162) ~[classes/:na]
at hr.tvz.java.vjezbe.glavna.Glavna.main(Glavna.java:29) ~[classes/:na]
```

# Apache Maven

---

- Alat za pojednostavljenje upravljanja ovisnostima (engl. *dependencies*) o vanjskim bibliotekama
- Umjesto dodavanja kopija JAR datoteka koje predstavljaju vanjske biblioteke u sam Eclipse projekt, Apache Maven omogućava kreiranje lokalnog repozitorija koji se automatski ažurira resursima iz globalnog repozitorija na Internetu
- Ovisnostima se upravlja iz datoteke „pom.xml”
- Razvojno okruženje Eclipse Luna je opremljeno potrebnim dodacima za korištenje Mavena
- Da bi se projekt konfigurirao pomoću Mavena, na početku je potrebno pretvoriti ga u „Maven Project” korištenjem opcije „Configure->Convert to Maven Project”



# Apache Maven

---

- Nakon toga pojavljuje se ekran koji zahtijeva definiranje parametara za konfiguriranje Mavena:

**Create new POM**

**Maven POM**  
This wizard creates a new POM (pom.xml) descriptor for Maven.

Project: /Iznimke

Artifact

Group Id: Iznimke

Artifact Id: Iznimke

Version: 0.0.1-SNAPSHOT

Packaging: jar

Name: Iznimke

Description:

Finish Cancel

# Apache Maven

---

- Potvrđivanjem konfiguracijskih parametara unutar projekta kreira se „pom.xml” datoteka sa sljedećim sadržajem:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>Iznimke</groupId>
  <artifactId>Iznimke</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Iznimke</name>
  <build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

# Apache Maven

- Da bi se definirala vanjska biblioteka koju Maven mora dodati u „Build Path” projekta, prvo je potrebno u centralnom Maven repozitoriju pronaći odgovarajući dio XML konfiguracije koja se mora ubaciti u datoteku „pom.xml”
- To je moguće postići da se na stranici „<http://search.maven.org/>” u tražilicu upiše naziv biblioteke i odabere odgovarajuća verzija:

## The Central Repository

[SEARCH](#) | [ADVANCED SEARCH](#) | [BROWSE](#) | [QUICK STATS](#)

SEARCH

[New: About Central](#)

[Advanced Search](#) | [API Guide](#) | [Help](#)

### Search Results

< 1 > displaying 1 to 11 of 11

GroupId	ArtifactId	Latest Version	Updated	Download
<a href="#">ch.qos.logback</a>	<a href="#">logback-classic</a>	<a href="#">1.1.2</a> <a href="#">all (56)</a>	02-Apr-2014	<a href="#">pom</a> <a href="#">jar</a> <a href="#">javadoc.jar</a> <a href="#">sources.jar</a> <a href="#">tests.jar</a>
<a href="#">ch.qos.logback.contrib</a>	<a href="#">logback-mongodb-classic</a>	<a href="#">0.1.2</a> <a href="#">all (2)</a>	03-Feb-2013	<a href="#">pom</a> <a href="#">jar</a> <a href="#">javadoc.jar</a> <a href="#">sources.jar</a>
<a href="#">ch.qos.logback.contrib</a>	<a href="#">logback-json-classic</a>	<a href="#">0.1.2</a> <a href="#">all (2)</a>	03-Feb-2013	<a href="#">pom</a> <a href="#">jar</a> <a href="#">javadoc.jar</a> <a href="#">sources.jar</a>

# Apache Maven

- Dio konfiguracije koji je potrebno prebaciti u „pom.xml” označen je „tagovima” pod nazivom „dependency”:

The screenshot shows the 'The Central Repository' website. At the top, there's a search bar with 'logback classic' entered and a 'SEARCH' button. Below the search bar, there are links for 'New: About Central', 'Advanced Search', 'API Guide', and 'Help'. The main heading is 'Artifact Details For [ch.qos.logback](#) : [logback-classic](#) : [1.1.2](#)'. Below this, it says 'Click on a link above to browse the repository.' There are two main sections: 'Project Information' and 'Project Object Model (POM)'. The 'Project Information' section has fields for GroupId (ch.qos.logback), ArtifactId (logback-classic), and Version (1.1.2). The 'Project Object Model (POM)' section shows the XML code for the artifact. In the 'Dependency Information' section, the 'Apache Maven' dependency is highlighted with a red box, showing the XML code for the dependency. Below this, there are links for 'Apache Builder', 'Apache Ivy', 'Groovy Grape', 'Gradle/Grails', 'Scala SBT', and 'Leiningen'.

**The Central Repository**

SEARCH | ADVANCED SEARCH | BROWSE | QUICK STATS

logback classic

[New: About Central](#) | [Advanced Search](#) | [API Guide](#) | [Help](#)

SEARCH

Artifact Details For [ch.qos.logback](#) : [logback-classic](#) : [1.1.2](#)

Click on a link above to browse the repository.

**Project Information**

GroupId:

ArtifactId:

Version:

**Dependency Information**

**Apache Maven**

```
<dependency>
  <groupId>ch.qos.logback</groupId>
  <artifactId>logback-classic</artifactId>
  <version>1.1.2</version>
</dependency>
```

**Project Object Model (POM)**

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-parent</artifactId>
    <version>1.1.2</version>
  </parent>
  <artifactId>logback-classic</artifactId>
  <packaging>jar</packaging>
  <name>Logback Classic Module</name>
  <description>logback-classic module</description>
  <url>http://logback.qos.ch</url>
  <licenses>
    <license>
      <name>Eclipse Public License - v 1.0</name>
      <url>http://www.eclipse.org/legal/epl-v10.html</url>
    </license>
    <license>
      <name>GNU Lesser General Public license</name>
```

Apache Builder  
Apache Ivy  
Groovy Grape  
Gradle/Grails  
Scala SBT  
Leiningen

# Apache Maven

---

- XML konfiguraciju potrebno je ubaciti između „tagova” pod nazivom „dependencies” koje je potrebno smjestiti u „pom.xml” datoteku npr. između „tagova” „name” i „build”:

```
<name>Iznimke</name>
<dependencies>
    <dependency>
        <groupId>ch.qos.logback</groupId>
        <artifactId>logback-classic</artifactId>
        <version>1.1.2</version>
    </dependency>
</dependencies>
<build>
```

# Primjer zadatka s iznimkama – pogađanje brojeva

---

```
public class PremaliBrojException extends Exception {  
  
    public PremaliBrojException(String poruka) {  
        super(poruka);  
    }  
  
    public PremaliBrojException(Throwable uzrok) {  
        super(uzrok);  
    }  
  
    public PremaliBrojException(String poruka, Throwable uzrok) {  
        super(poruka, uzrok);  
    }  
}
```

# Primjer zadatka s iznimkama – pogađanje brojeva

---

```
public class PrevelikiBrojException extends Exception {  
  
    public PrevelikiBrojException(String poruka) {  
        super(poruka);  
    }  
  
    public PrevelikiBrojException(Throwable uzrok) {  
        super(uzrok);  
    }  
  
    public PrevelikiBrojException(String poruka, Throwable uzrok) {  
        super(poruka, uzrok);  
    }  
}
```

# Primjer zadatka s iznimkama – pogađanje brojeva – Glavna.java (1)

---

```
package primjer.glavna;
```

```
import java.util.Random;
```

```
import java.util.Scanner;
```

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
```

```
import primjer.iznimke.PremaliBrojException;
```

```
import primjer.iznimke.PrevelikiBrojException;
```

```
public class Glavna {
```

```
    public static final int MAX_BROJ = 100;
```

```
    private static int generiraniBroj = 0;
```

```
    private static final Logger logger = LoggerFactory.getLogger(Glavna.class);
```



# Primjer zadatka s iznimkama – pogađanje brojeva – Glavna.java (2)

---

```
public static void main(String[] args) {  
  
    Random dajBroj = new Random();  
  
    generiraniBroj = dajBroj.nextInt(MAX_BROJ);  
    Logger.info("Generirao si broj " + generiraniBroj);  
  
    Scanner unosBroja = new Scanner(System.in);  
  
    boolean pogodio = false;  
    int uneseniBroj = 0;
```

# Primjer zadatka s iznimkama – pogađanje brojeva – Glavna.java (3)

---

```
do {
    System.out.println("Unesite broj");
    uneseniBroj = unosBroja.nextInt();
    logger.info("Unesen je broj " + uneseniBroj);

    try {
        provjera(uneseniBroj);
        pogodio = true;
    } catch (PremaliBrojException e) {
        logger.info(e.getMessage(), e);
        System.out.println(e.getMessage());
    } catch (PrevelikiBrojException ex) {
        logger.info(ex.getMessage(), ex);
        System.out.println(ex.getMessage());
    }
}while(pogodio == false);
```

# Primjer zadatka s iznimkama – pogađanje brojeva – Glavna.java (4)

---

```
System.out.println("BRAVOOOO!!! Pogodili ste traženi broj: " + generiraniBroj);
unosBroja.close();
}

public static void provjera(int broj) throws PremaliBrojException, PrevelikiBrojException {
    if (broj < generiraniBroj) {
        throw new PremaliBrojException("Unijeli ste premali broj!!!");
    }
    else if (broj > generiraniBroj) {
        throw new PrevelikiBrojException("Unijeli ste preveliki broj!!!");
    }
}
}
```

# Pitanja s certifikata (1)

---

- I. Which class has the fewest subclasses: `Exception`, `RuntimeException`, or `Error`? Select the correct statement.
  - A. The `Exception` class has fewer subclasses than the `RuntimeException` and `Error` classes.
  - B. The `RuntimeException` class has fewer subclasses than the `Exception` and `Error` classes.
  - C. The `Error` class has fewer subclasses than the `Exception` and `RuntimeException` classes.

# Pitanja s certifikata (2)

---

Given:

```
public static void test() throws FileNotFoundException {  
    try {  
        throw FileNotFoundException();  
    } finally {  
    }  
}
```

Determine why it will not compile. Which statement is correct?

- A. The code will not compile without a catch clause.
- B. The code needs the new keyword after the throw keyword.
- C. The finally clause should be the final clause.
- D. There is no class called `FileNotFoundException`.

# Pitanja s certifikata (3)

---

What new features came with Java 7 to enhance exception handling capabilities? (Choose all that apply.)

- A. The multi-catch feature
- B. The boomerang feature
- C. The try-with-resources feature
- D. The try-with-riches feature

# Pitanja s certifikata (4)

---

Given:

```
String typeOfDog = "Mini Australian Shepherd";  
typeOfDog = null;  
System.out.println(typeOfDog.length);
```

Which of the following is true?

- A. A `NullPointerException` will be thrown.
- B. A `RuntimeException` will be thrown.
- C. An `IllegalArgumentException` will be thrown.
- D. A compilation error will occur.

# Pitanja?

---