



## Testování – TestNG

TestNG

Results of this test run

Suites: 1/1

Tests: 22/22

Methods: 115/115

Passed: 115

Failed: 0

Skipped: 0

Failed tests

All tests

Failure exception

TestNG JDK 1.5 ( 115/0/0/0 )

Nopackage ( 1/0/0/0 )

Regression1 ( 17/0/0/0 )

Regression2 ( 16/0/0/0 )

Basic ( 3/0/0/0 )

test.sample.Basic1.basic1

test.sample.Basic2.basic2

test.Misc.makeSureSetUpWithParameterWithNoParametersF

Exclusion ( 4/0/0/0 )

Dependents ( 23/0/0/0 )

Inheritance ( 4/0/0/0 )

JUnit ( 3/0/0/0 )

Test outer scope ( 1/0/0/0 )

Test inner scope ( 1/0/0/0 )

AfterClassCalledAtEnd ( 3/0/0/0 )

Triangle ( 3/0/0/0 )

CheckTrianglePost ( 1/0/0/0 )

Test class groups 1 ( 3/0/0/0 )

Test class groups 2 ( 3/0/0/0 )

Factory ( 5/0/0/0 )



```
public class SimpleTest {  
  
    @BeforeClass  
    public void setUp() {  
        // code that will be invoked  
        // when this test is instantiated  
    }  
  
    @Test  
    public void aFastTest() {  
        System.out.println("Fast test");  
    }  
  
    @Test  
    public void aSlowTest() {  
        System.out.println("Slow test");  
    }  
}
```

Testovací  
metoda

Testovací  
metoda



```
public class SimpleTest {  
  
    @BeforeClass  
    public void setUp() {  
        // code that will be invoked  
        // when this test is instantiated  
    }  
  
    @Test(groups = { "fast" })  
    public void aFastTest() {  
        System.out.println("Fast test");  
    }  
  
    @Test(groups = { "slow" })  
    public void aSlowTest() {  
        System.out.println("Slow test");  
    }  
}
```

Testovací  
skupina

Testovací  
skupina



```
<test name="T1" >  
  <classes>  
    <class name="SimpleTest">  
      <methods>  
        <include name="aFastTest" />  
      </methods>  
    </class>  
  </classes>  
</test>
```



Test



```
<test name="T1" >  
  <classes>  
    <class name="SimpleTest">  
      <methods>  
        <include name=".*Fast.*" />  
      </methods>  
    </class>  
  </classes>  
</test>
```



Test



```
<test name="T2" >  
  <classes>  
    <class name="SimpleTest" />  
  </classes>  
</test>
```



Test



```
<test name="T3">  
  <packages>  
    <package name="test.sample" />  
  </packages>  
</test>
```



Test





```
<test name="T4">  
  <groups>  
    <run>  
      <exclude name="fast" />  
      <include name="slow" />  
    </run>  
  </groups>  
  
  <classes>  
    ...  
  </classes>  
</test>
```



Test



```
<suite name="S1" verbose="1">  
  <test name="T1">  
    ...  
  </test>  
  <test name="T2">  
    ...  
  </test>  
  <test name="T3">  
    ...  
  </test>  
</suite>
```



Test  
Suite



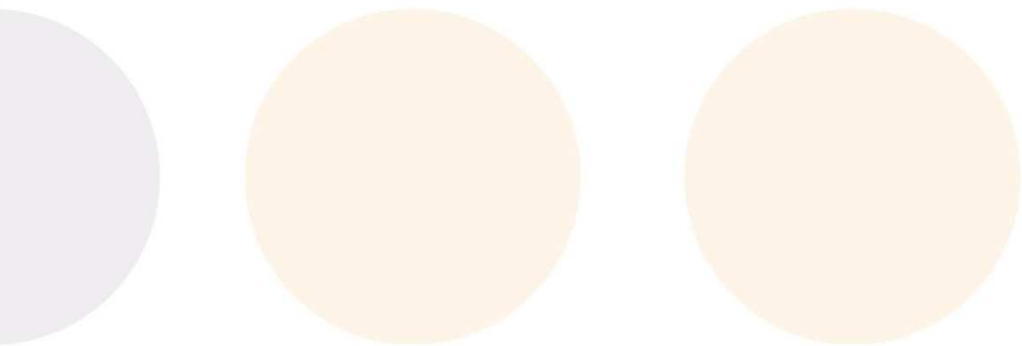
# Shrnutí

- Konfigurace
  - anotacemi
    - `@Test`
  - xml souborem
- Granularita
  - testovací metody ~> skupiny
  - testovací metody / skupiny ~> testy ~> suits



# @Test

- Označuje spustitelnou testovací metodu
  - lze použít u celé třídy





# @Test

- Označuje spustitelnou testovací metodu
  - lze použít u celé třídy
    - tzv. class-level annotation
  - pak jsou implicitně označeny všechny public metody

```
@Test  
public class SimpleTest {  
  
    public void aFastTest() {  
        System.out.println("Fast test");  
    }  
  
}
```


Testovací  
metoda



## Jiný příklad

- Přidávat metody do skupiny na úrovni třídy
  - všechny testovací metody pak patří do této skupiny

```
@Test(groups = { "integration-test" })  
public class SimpleTest {  
  
    @Test(groups = { "fast" })  
    public void aFastTest() {  
        System.out.println("Fast test");  
    }  
}
```





# @Test – Atributy

groups	seznam skupin, do kterých testovací metoda patří
enabled	je testovací metoda povolena? (přepínač)
expectedExceptions	seznam očekávaných výjimek – pro testování okrajových podmínek
description	popisek testovací metody

# Co dál umí TestNG?

- Závislosti mezi testovacími metodami

```
@Test
```

```
public void startServer() {  
    System.out.println("Starting server.");  
}
```

```
@Test(dependsOnMethods = { "startServer" })
```

```
public void doSomethingWithServer() {  
    System.out.println("Doing things on server.");  
}  
}
```



# Co dál umí TestNG?

- Závislosti mezi testovacími metodami

```
@Test
public void startServer() {
    System.out.println("Starting server.");
}

@Test(dependsOnMethods = { "startServer" })
public void doSomethingWithServer() {
    System.out.println("Doing things on server.");
}
}
```

hard  
dependency

# Co dál umí TestNG?

- Závislosti mezi testovacími metodami

```
@Test
public void startServer() {
    System.out.println("Starting server.");
}

@Test(dependsOnMethods = { "startServer" },
      alwaysRun = true)
public void doSomethingWithServer() {
    System.out.println("Doing things on server.");
}
}
```

soft  
dependency

# Co dál umí TestNG?

- Závislosti mezi testovacími metodami
  - lze definovat i závislost na skupině

```
@Test(groups = { "init" })  
public void startServer() {  
    System.out.println("Starting server.");  
}
```

```
@Test(dependsOnGroups = { "init" })  
public void doSomethingWithServer() {  
    System.out.println("Doing things on server.");  
}
```



# Co dál umí TestNG?

- Obecnou inicializaci před testem a uklízení po testu
  - velmi jemná granularita
    - @BeforeSuite / @AfterSuite
    - @BeforeTest / @AfterTest
    - @BeforeGroups / @AfterGroups
    - @BeforeClass / @AfterClass
    - @BeforeMethod / @AfterMethod
  - takto anotovaných metod může být více
    - použijí se všechny



# Co dál umí TestNG?

- Pouštět testy ve více vláknech

```
<suite name="S1" parallel="methods" thread-count="5">
```

```
<suite name="S2" parallel="tests" thread-count="5">
```

```
<suite name="S3" parallel="classes" thread-count="5">
```





# Co dál umí TestNG?

- Pouštět test vícekrát z různých vláken
  - vlákna běží současně

```
@Test(threadPoolSize = 3,  
      invocationCount = 10)  
public void doSomethingWithServer() {  
    System.out.println("Doing things on server.");  
}
```



# Co dál umí TestNG?

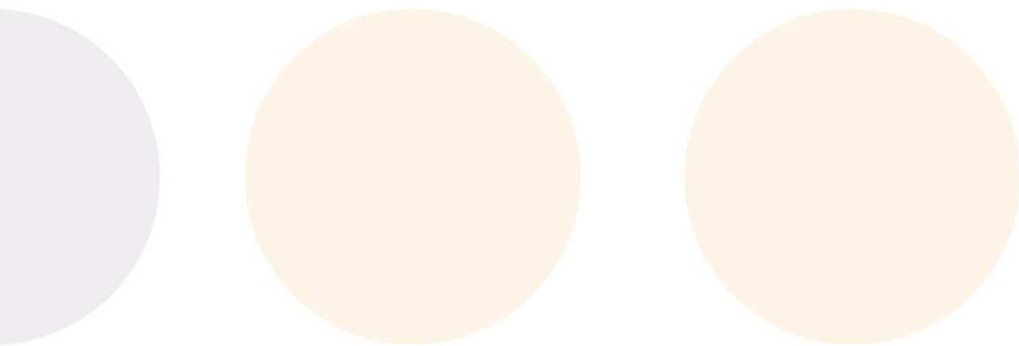
- Specifikovat timeout (v milisekundách)
  - už žádné rogue testy, které blokují vlákno navždy

```
@Test(threadPoolSize = 3,  
      invocationCount = 10,  
      timeout=10000)  
public void doSomethingWithServer() {  
    System.out.println("Doing things on server.");  
}
```



# Co dál umí TestNG?

- Parametrizovat testy

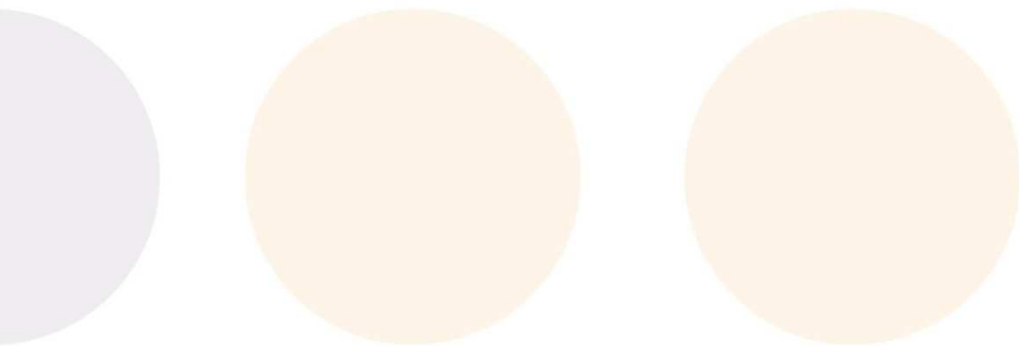






# Co dál umí TestNG?

- Parametrizovat testy **přímo v konfiguraci**
  - jednoduché a rychlé
  - ale méně flexibilní
    - smíte mít jenom String
  - parametry mohou být nepovinné
    - anotace **@Optional**





# Parametry testů

- Speciální anotace **@Parameters**
  - parametry se mapují podle pořadí, ne podle jmenné konvence

```
@Test
```

```
@Parameters({ "first-name", "last-name" })
```

```
public void doSomethingWithServer(
```

```
    String firstName,
```

```
    String lastName) {
```

```
    System.out.println(firstName + " " + lastName +  
                        " is doing things on server.");
```

```
}
```



# Parametry testů


- Definice hodnot parametrů v ngconfig.xml

```
<suite name="S1">  
  <parameter name="first-name" value="Erik" />  
  <parameter name="last-name" value="Kratochvíl" />  
  ...  
</suite>
```

```
<suite name="S2">  
  <parameter name="first-name" value="Martin" />  
  <parameter name="last-name" value="Hlavatý" />  
  ...  
</suite>
```



# Co dál umí TestNG?

- Parametrizovat testy **přes tzv. DataProvider**
    - java metoda, která vrací **pole polí** obsahující jednotlivé hodnoty parametrů
      - testy se pouští opakovaně – pro každý řádek parametrů
    - velmi silný nástroj
      - testovací data lze načítat z DB, XML
      - prostě odkudkoli
- 



# Data Provider

- Speciální anotace **@DataProvider**
  - metoda vrací **Object[][]** nebo **Iterator<Object[]>**

```
@Test(dataProvider = "namesProvider")
public void doesPersonLive(
    String firstName,
    String lastName) {
    ...
}
```

```
@DataProvider(name = "namesProvider")
public Object[][] provideNames() {
    return new Object[][] {
        { "Erik", "Kratochvíl" },
        { "Thales", "Milétský" }
    }
}
```



# DataProvider

- Alternativní způsob specifikace parametrů.
- Použití:
  - @Test(dataProvider = "jmeno")
- Zdroj dat - @DataProvider
  - Statická metoda nebo metoda třídy, ve které je provider použit.
  - Vrací pole polí objektů.
- Testy se pouští opakovaně – pro každý řádek (pole parametrů) vrácený DataProviderem.





# Co dál umí TestNG?

- Žít s JDK 1.4
  - používají se anotace v JavaDoc
- Definovat vlastní interceptory a listeners
  - měnit nebo monitorovat chování TestNG
    - sledovat volání testovacích metod
    - měnit pořadí testovacích metod
- "Dynamické" testování
  - Factory
    - vyrábí testovací třídy s danými parametry
  - vhodné, pokud je obor hodnot parametrů velký



## Bodované úkoly na zápočet







# Testovací aplikace

- Checkout (swicviceni/profinit)  
[https://subversion.assembla.com/svn/swicviceni/trunk/04\\_testng](https://subversion.assembla.com/svn/swicviceni/trunk/04_testng)
- Importovat do workspace
  - File -> Import -> Existing projects into workspace
- Lze spouštět buď pomocí ANT nebo Eclipse pluginu (doporučeno)
  - Help -> Install new software
  - Work with = „<http://beust.com/eclipse>“ a „Add“
  - Zaškrtnout TestNG a klikat na Next
  - Restartovat Eclipse



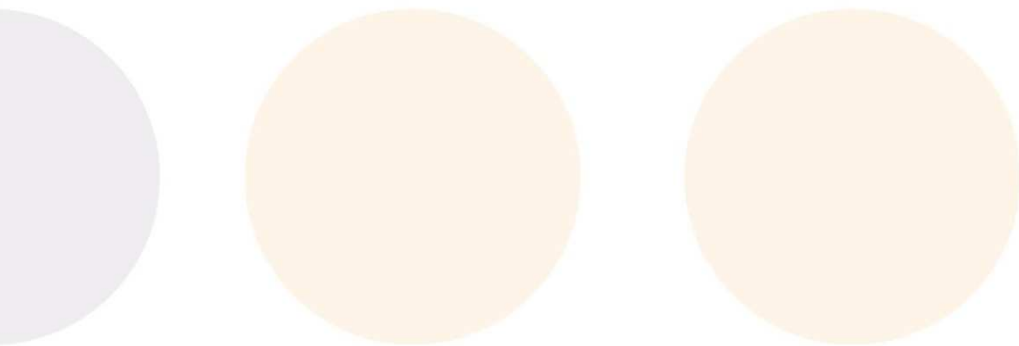
# Úkol 1

- **Spust'te všechny testy** a
  - prohlédněte si výstup v konzoli
  - pokud používáte Eclipse plugin tak si prohlédněte i výstup v TestNG view
- Zajistěte, aby **se nespouštěly testy, které neprocházejí**
  - Musí být možné opět rychle tyto testy spustit (**bez zásahu do kódu**)



# Úkol 2

- **Upravte Ukol1Test** tak, aby byly otestovány všechny dle vašeho názoru důležité vstupy
- **Upravte Ukol2Test** – napište test metody Ukol2.krat





# Úkol 3

- Upravte **Ukol2Test** tak, že
  - Implementujete metodu **testKrat**
    - využijte DataProvider
  - Otestujete minimálně, že:
    - $2 * 3 = 6$
    - $0 * 3 = 0$
    - $-1 * -1 = 1$
    - $1\,000\,000\,000 * 1 = 1\,000\,000\,000$



# Úkol 4

- Zjistěte pomocí TestNG, která ze standardních Java tříd `StringBuilder` a `StringBuffer` je thread-safe.





# Odevzdávání

- Na cvičeních
- Mailem na [martin.hlavaty@profinit.eu](mailto:martin.hlavaty@profinit.eu)
  - soubor **ngconfig.xml**
  - **zdrojové soubory**, které jste měnili
    - měly by to být jen
      - Ukol1Test.java
      - Ukol2Test.java
      - Ngconfig.xml
  - deadline – příští praktická cvičení



# Diskuse

- Komentáře
- Otázky
- Připomínky
- Upřesnění
- Poznámky
- ...

