

IŠ PRAEITOS PASKAITOS

- kaip išvengti history objekto perdavimo per props?
 - react-router 4.4.0-beta+ bus `__RouterContext`
 - o dabar galima pavyzdžiui `withRouter`

```
import { withRouter } from 'react-router';
<...>
export default withRouter(AppContainer)
// arba jeigu komponentas tame pačiame modulyje
var AppContainer = withRouter((props) => {
  return (<div>
    <button onClick={() => props.history.push("products"); } />
    {props.children}
  )
})
```

- į props objektą `history/match/location` gaus tik šis komponentas bet kuriame komponentų medžio lygyje



IŠ PRAEITOS PASKAITOS

- arrow funkcijos metodas klasėje gauna objektą `this`

```
class C extends Component {  
  handleChange = (event) => {  
    this.setState({value: event.target.value});  
  }  
}
```

- be arrow funkcijos reiktų naudoti `bind()`:

```
class C extends Component {  
  constructor(props) {  
    super(props);  
    this.handleChange = this.handleChange.bind(this);  
  }  
  handleChange(event) {  
    this.setState({value: event.target.value});  
  }  
}
```



IŠ PRAEITOS PASKAITOS

- formose `<input type="text">`, `<textarea>` ir `<select>` visi gali turėti `value={this.state...}`
 - arba keli pasirinkimai `<select multiple={true} value={['B', 'C']}>`
- vienos funkcijos pvz. visiems elementams formoje:

```
handleChange = (event) => {  
  const target = event.target;  
  const value = target.type==='checkbox'?target.checked:target.value;  
  const name = target.name;  
  this.setState({ [name]: value });  
}  
// formos laukai turi turėti name, pvz. <input name="kaip state">
```



IŠ PRAEITOS PASKAITOS

- jeigu linux'uose:
 - neatsinaujina po pakeitimų react aplikacija
 - nepasistartuoja su `npm start` ir iškrenda klaida, susijusi su WATCH
 - kažkuriam laikui pradėjus neatsinaujina react vaizdas naršyklėje
- galimai pasiekėte linux watch (failų stebėjimų) limitą, o jį padidinti galima taip:

```
echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.conf && sudo sysctl -p
```





AKADEMIJA.IT

INFOBALT IR TECH CITY

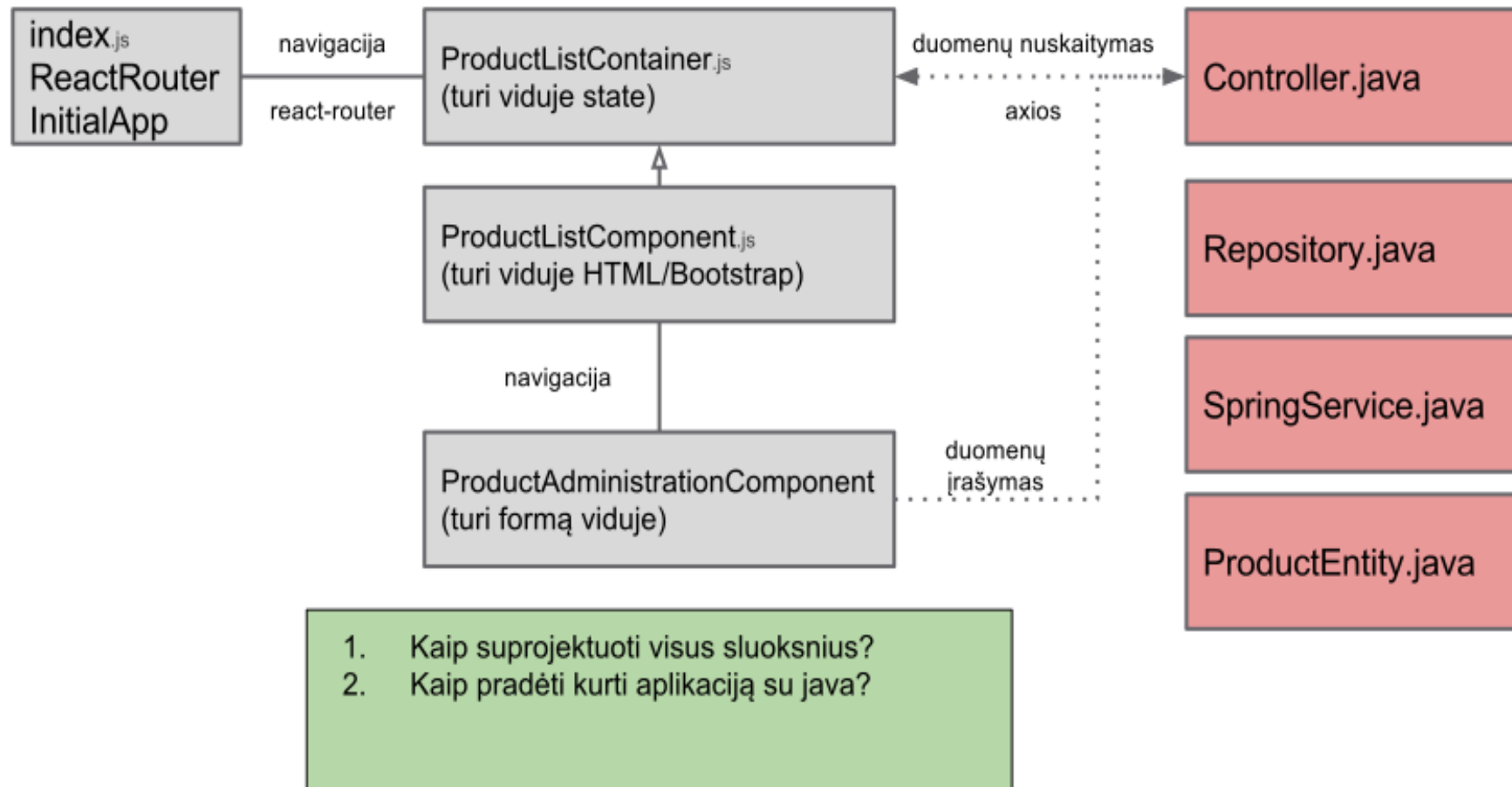
TOMCAT. MAVEN. SPRING BOOT APLIKACIJA

Andrius Stašauskas

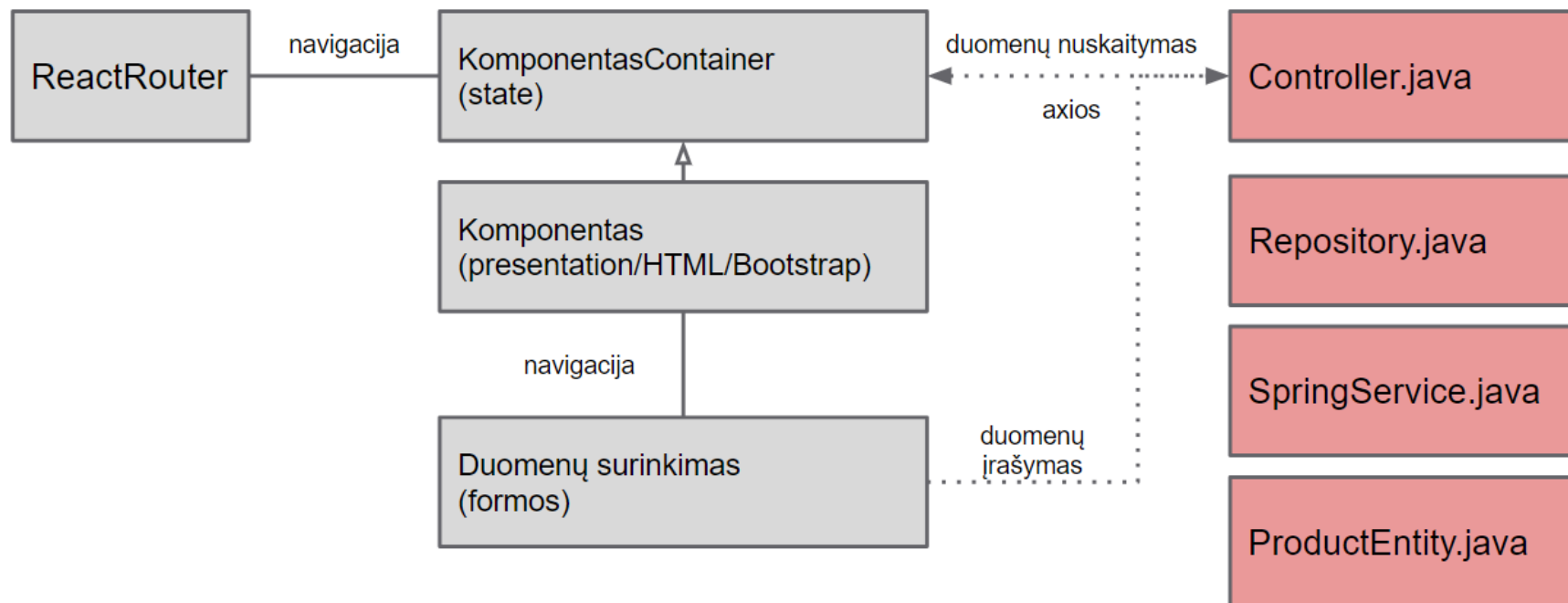
andrius@stasauskas.lt

<http://stasauskas.lt/itpro2018/>

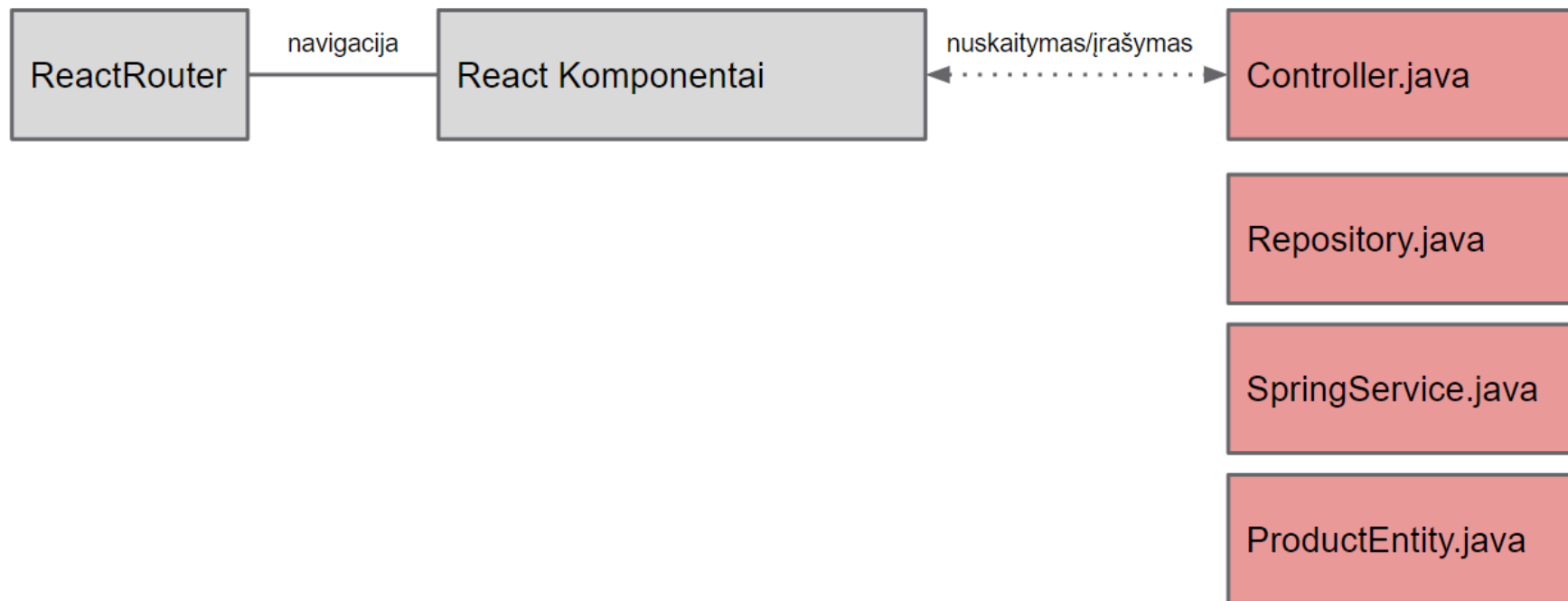
KĄ JAU MOKAME IR KO DAR NE



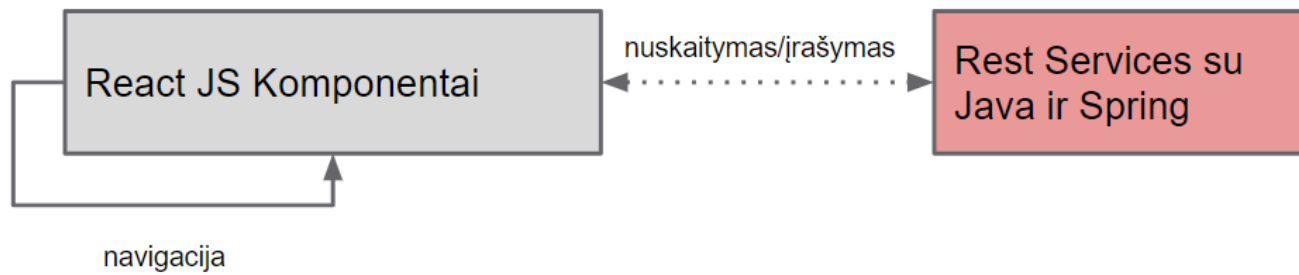
KĄ JAU MOKAME IR KO DAR NE



KĄ JAU MOKAME IR KO DAR NE



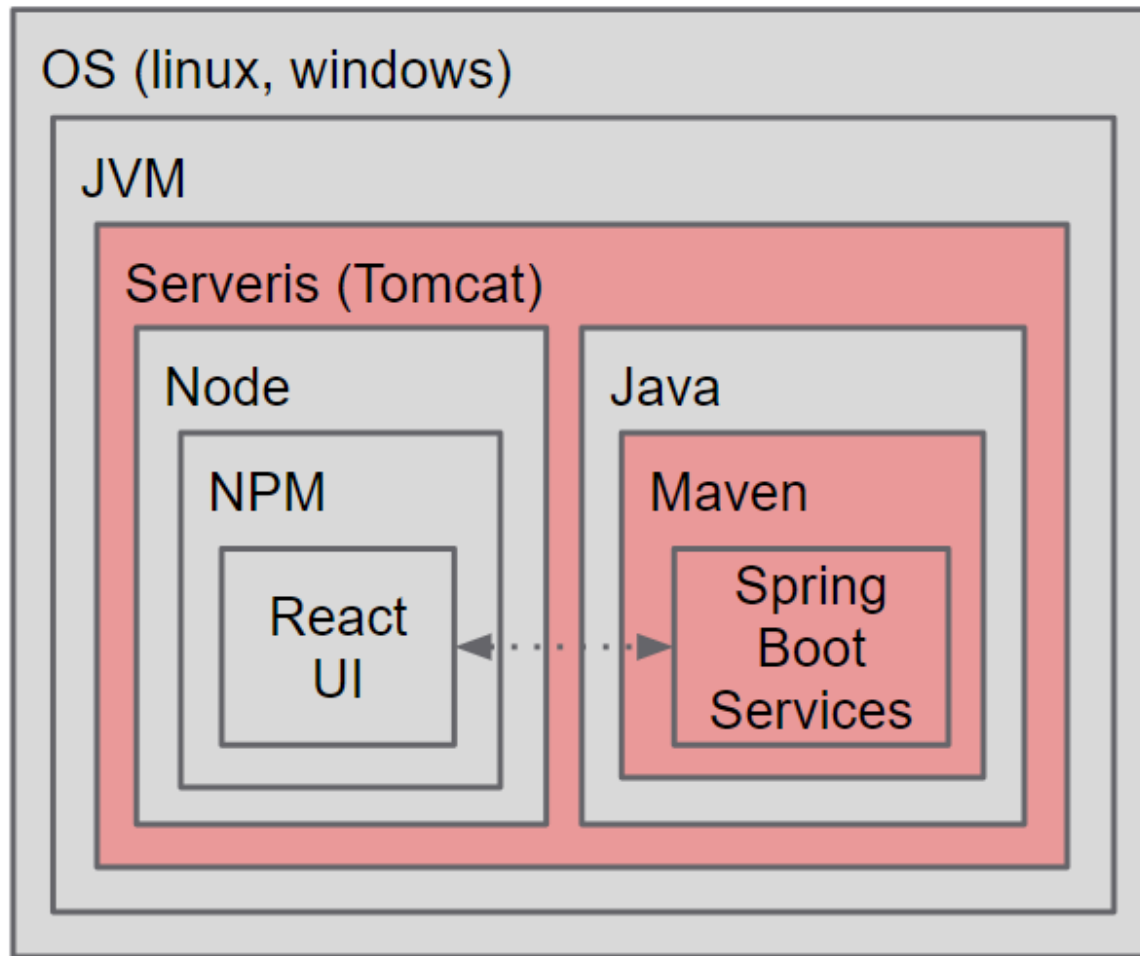
KĄ JAU MOKAME IR KO DAR NE



KĄ JAU MOKAME IR KO DAR NE



KĄ JAU MOKAME IR KO DAR NE



TURINYS

- Tomcat
 - WAR failai
- Maven
 - POM failai
 - Archetipai
- kaip sukurti Spring Boot aplikaciją
 - kaip prie jos prijungti React



TOMCAT



AKADEMIJA.IT

INFOBALT IR TECH CITY

TOMCAT

- **Apache Tomcat** - tai žiniatinklio konteineris, kuris įgalina aplikacijų, sukurtų Java Servlet arba JavaServer Pages (JSP) pagrindu, veikimą. Dauguma modernių žiniatinklio aplikacijų kūrimo karkasų yra įgyvendinti Java Servlet pagrindu: JavaServer Faces, Struts, Spring.
- Šis serveris gali būti naudojamas ir kaip HTTP serveris.



APACHE TOMCAT KATALOGŲ STRUKTŪRA

Katalogas	Aprašymas
/bin	Skriptai
/conf	Konfigūracija
/lib	Bibliotekos
/logs	Įvykių žurnalai
/temp	Laikų failų direktorija
/webapp	Aplikacijų direktorija
/work	Darbinė direktorija



UŽDUOTIS 1 - PARUOŠIMAS DARBUI

```
$ wget http://apache.mirror.vu.lt/apache/tomcat/\
> tomcat-8/v8.5.35/bin/apache-tomcat-8.5.35.tar.gz
$ tar xvzf apache-tomcat-8.5.35.tar.gz
$ cd apache-tomcat-7.0.92/bin
$ ./startup.sh
$ ./shutdown.sh - išjungimas
```

- Atidaryti: `http://localhost:8080`
- Tomcat serveris startuoja HTTP konektorių, kurio portas yra 8080 (portas leidžia skirtingoms aplikacijoms toje pačioje mašinoje dirbti drauge vienu metu).



UŽDUOTIS 1 - APLIKACIJŲ TVARKYKLĖ

- Apache Tomcat serveryje yra sudiegta aplikacijų tvarkyklė, kuri leidžia atlikti tam tikrus aplikacijų administravimo darbus. Norint ją pasinaudoti, pirmiausia reikia atlikti saugumo nustatymus
- Faile `/conf/tomcat-users.xml`:



```
<user username="tomcat" password="tomcat" roles="manager-gui"/>
```

- Vartotojui tomcat, kurio slaptažodis tomcat, yra suteikta rolė manager-gui.



APLIKACIJŲ TVARKYKLĖ

← → ↻ localhost:8080/manager/html



Tomcat Web Application Manager

Message: OK

Manager
[List Applications](#) [HTML Manager Help](#) [Manager Help](#) [Server Status](#)

Applications

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy
Deploy directory or WAR file located on server

Context Path (required):
XML Configuration file URL:
WAR or Directory URL:



APLIKACIJŲ TVARKYKLĖ

- Naudojantis šia tvarkykle galime atlikti tokias funkcijas:
 - Sustabdyti aplikaciją
 - Iš naujo paleisti aplikaciją
 - Panaikinti aplikaciją
 - Pridėti naują aplikaciją
 - Sunaikinti aktyvias sesijas
 - Pamatyti serverio parametrus



UŽDUOTIS 2 - WAR PALEIDIMAS

- Atsisiųskite war failą
 - <https://tomcat.apache.org/tomcat-6.0-doc/appdev/sample/sample.war>
 - <http://stasauskas.lt/itpro2018/6-sample.war>
- Nueikite į <http://localhost:8080/manager/html> ir paleiskite (deploy) java aplikaciją `sample.war`
 - "Select WAR file to upload" -> "Deploy"
 - išbandykite, [ar veikia](#), sustabdykite ir ištrinkite (undeploy)



MAVEN



MAVEN TURINYS

- Kas yra Maven
- Katalogų struktūra
- Darinio gyvavimo ciklas (angl. lifecycle)
- Projekto aprašas (angl. descriptor)
- Priklausomybės (angl. dependencies)
- Papildiniai (angl. plugins)
- Savybės (angl. properties)
- Profiliai (angl. profile)
- Archetipas



KAS YRA MAVEN

- Darinio (angl. build) sukūrimo įrankis
- Standartizuota darinio sukūrimo infrastruktūra
- Priklausomybių (angl. dependency) valdymo įrankis
- Kokybės įrankis
- Atviro kodo Apache projektas
- Maven skirtas Java'ai taip, kaip NPM skirtas Node'ui



PRIEŠ MAVEN (IKI 2003)

- javac komandos naudojimas projektų kompiliavimui
- Nuo IDE (Integrated development Environment) priklausomas projektų kompiliavimas
- GNU Make įrankis
- Ant (Another Neat Tool) įrankis



MAVEN RAIDA

- Maven 1 (2003)
- Maven 2 (2005) - nesuderinamas su Maven 1.
- Maven 3 (2010) - suderinamas su Maven 2, stabilesnis, turintis daugiau funkcionalumo.



MAVEN KONFIGŪRACIJA

- Windows nustatymai
 - `%MAVEN_HOME%\conf\settings.xml`
 - Lokalios repositorijos vieta: `%UserProfile%\m2\`
- Linux nustatymai
 - `/usr/local/maven/conf/settings.xml`
 - Lokalios repositorijos vieta: `~/m2/`
- Nurodyti kitus Maven nustatymus:
 - `mvn --settings=[PATH_TO_SETTINGS_FILE]`
- Nurodyti kitą lokalios repositorijos vietą:
 - `mvn -Dmaven.repo.local=/path/to/local/repo`



UŽDUOTIS 3 - MAVEN ĮDIEGIMAS

```
$ sudo apt-get install maven  
$ mvn -version  
Apache Maven 3.5.2
```



MAVEN FILOSOFIJA

- Susitarimas per konfigūraciją (mažiau konfigūracijos)
- Lengvas darinio sukūrimo procesas
- Geriausių praktikų šablonai
- Nuoseklus darinio sukūrimas



STANDARTINĖ MAVEN PROJEKTO STRUKTŪRA

Katalogas	Aprašymas
src	išeities kodas
src/main	pagrindinio artefakto išeities kodas
src/main/java	java išeities kodas
src/main/resources	nekompilijuojami resursai
src/main/webapp	žiniatinklio aplikacijos resursai
src/test	testavimui skirtas išeities kodas
src/test/java	testavimo java išeities kodas
src/test/resources	nekompilijuojami testavimo resursai
target	Maven darbinis katalogas
pom.xml	aprašo byla



DARINIO GYVAVIMO CIKLAI

- maven paremtas darinio gyvavimo ciklais, skirtais artefaktų sukūrimui (pvz. jar byla, war byla) ir paskirstymui
- yra trys įtaisyti (angl. build-in) darinių gyvavimo ciklai:
 - Default - projekto darinio pagaminimas ir įdiegimas
 - Clean - projekto išvalymas
 - Site - projekto svetainės sugeneravimas
- kiekvienas iš išvardintų ciklų yra sudarytas iš skirtingo rinkinio fazių
- fazė reprezentuoja konkretų gyvavimo ciklo etapą



PAGRINDINĖS “DEFAULT” FAZĖS

Fazė	Aprašymas
validate	patikrina ar projektas korektiškas ir visa reikiama informacija yra pasiekama
generate-sources	generuoja išeities kodą įtraukimui į kompiliavimą
generate-resources	generuoja resursus įtraukimui į paketą
compile	kompiluoja projekto išeities kodą
test	testuoja sukompiliuotą kodą naudojant nurodytą testavimo karkasą
package	supakuoja sukompiliuotą kodą į nurodytą paskirstymo formatą (pvz. jar)
integration-test	pagal poreikį, įdiegia paketus į aplinką, kurioje vykdomi integraciniai testai
verify	vykdo patikrinimus ar paketai atitinka kokybės kriterijus
install	įdiegia paketus į lokalią repozitoriją, tam, kad kiti lokalus projektai galėtų naudoti priklausomybės ryšius
deploy	galutinės paketų kopijos įdiegiamos į nutolusią repozitoriją, tam, kad pasidalinti su kitais PL kurėjais ir projektais



“CLEAN” FAZĖS

Fazė	Aprašymas
pre-clean	vykdo procesus, reikalingus prieš realų projekto valymą
clean	pašalina visas bylas, kurios buvo sugeneruotos ankstesnio darinio sukūrimo metu
post-clean	vykdo procesus, skirtus projekto valymo užbaigimui



“SITE” FAZĖS

Fazė	Aprašymas
pre-site	vykdo procesus, reikalingus prieš realų projekto sveitainės generavimą
site	generuoja projekto svetainės dokumentaciją
post-site	vykdo procesus, skirtus užbaigti svetainės generavimui ir pasiruošti svetainės įdiegimui
site-deploy	įdiegia sugeneruota svetainės dokumentaciją į nurodytą žiniatinklio serverį



MAVEN TIKSLŲ PAVYZDŽIAI

- Maven darinio sukūrimas yra vykdomas nurodant gyvavimo ciklo tikslus (angl. goals):

Komanda	Aprašymas
<code>mvn install</code>	įvykdo generate*, compile, test, package, integration-test ir install fazes
<code>mvn clean</code>	įvykdo tik clean gyvavimo ciklą
<code>mvn clean compile</code>	išvalo projektą ir įvykdo generate* ir compile fazes
<code>mvn compile install</code>	įvykdo generate*, compile, test, integration-test, package ir install fazes
<code>mvn test clean</code>	įvykdo generate*, compile, test fazes ir tada išvalo projektą



POM - PROJEKTO APRAŠAS

- POM (angl. Project Object Model) saugomas byloje pom.xml
 - analogiškai NPM/Node saugomas package.json
- Saugo projekto metaduomenis:
 - vardas ir versija
 - supakavimo tipas
 - įrankių nuorodos (CI, SCM ir pan.)
 - priklausomybės
 - papildiniai
 - profiliai - alternativios darinio sukūrimo konfigūracijos



POM - PROJEKTO APRAŠAS

- Naudojama XML kalba
- Vienas POM == vienas artefaktas
- POM ryšiai:
 - paveldėjimas
 - agregavimas



POM - PROJEKTO VARDAS (GAV)

- Maven projektas unikaliam identifikuojamas naudojant:
 - groupId - sutartas projekto grupavimo identifikatorius (be tarpų/kablelių)
 - artifactId - projekto vardas (be tarpų/kablelių)
 - version - projekto versija
- GAV sintaksė yra groupId:artifactId:version

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.lds.training</groupId>
  <artifactId>maven-training</artifactId>
  <version>1.0</version>
</project>
```



POM - SUPAKAVIMAS

- Darinio tipas identifikuojamas packaging elemente
- Nurodo Maven kaip pagaminti projekto darinį
- Supakavimo tipai:
 - pom, jar, war, ear, custom (pritaikytas)
 - tipas pagal nutilėjimą yra jar

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <artifactId>maven-training</artifactId>
  <groupId>org.lds.training</groupId>
  <version>1.0</version>
  <packaging>jar</packaging>
</project>
```



POM - PROJEKTO PAVELDĒJIMAS

POM bylos gali pavadēti šā konfigūracijā: groupId, versijā, projekto konfigūracijā, priklausomybes, papildinius, profilus ir t.t.

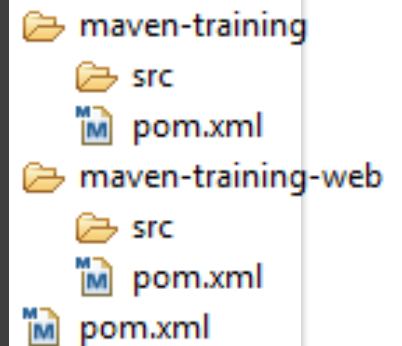
```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <parent>
    <artifactId>maven-training-parent</artifactId>
    <groupId>it.akademija.training</groupId>
    <version>1.0</version>
  </parent>
  <modelVersion>4.0.0</modelVersion>
  <artifactId>maven-training</artifactId>
  <packaging>jar</packaging>
</project>
```



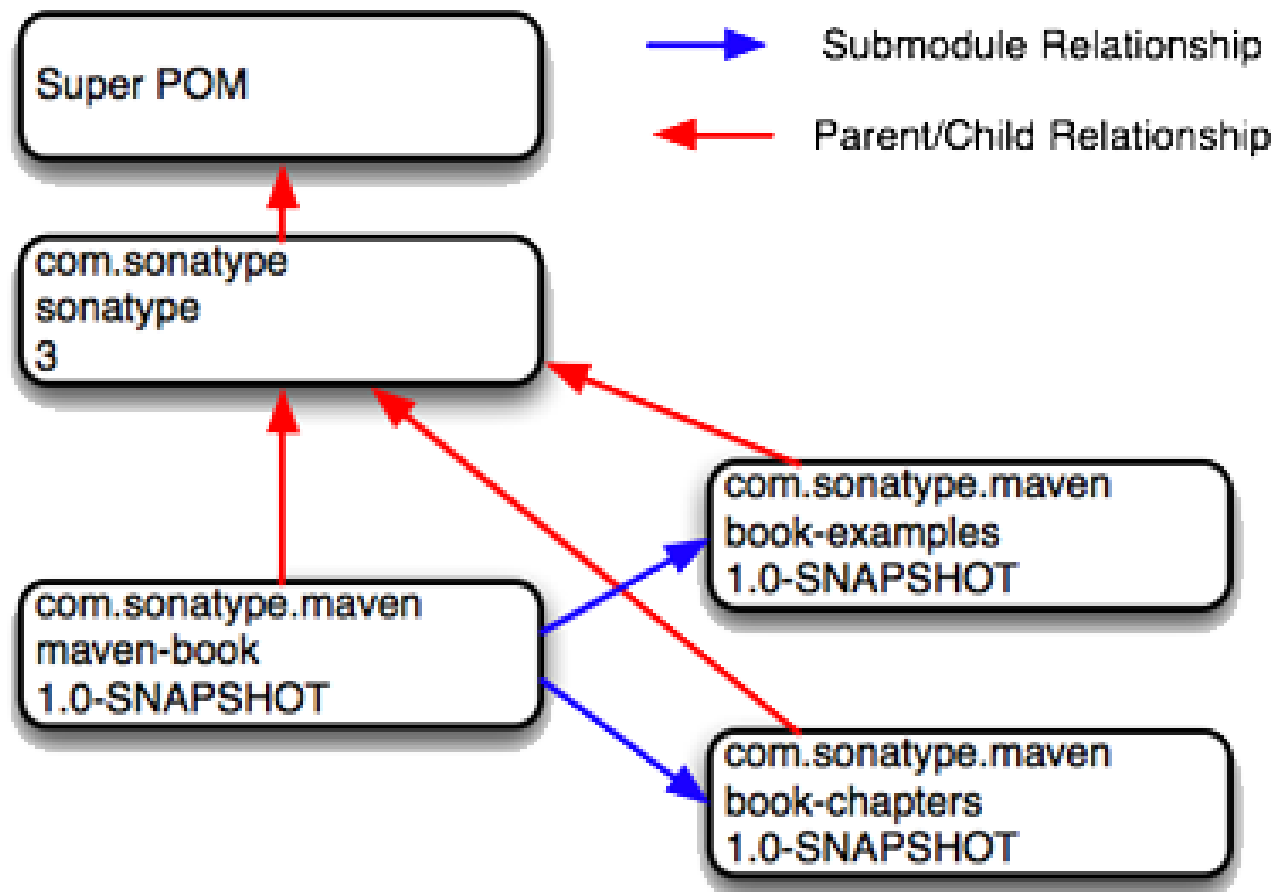
POM - MODULIAI

- Maven turi daugelio modulių palaikymą
- Kiekvienas Maven projektas sukuria 1 pagrindinį artefaktą
- Tėvo (angl. parent) POM naudojamas modulių grupavimui

```
<project>
  ...
  <packaging>pom</packaging>
  <modules>
    <module>maven-training</module>
    <module>maven-training-web</module>
  </modules>
</project>
```



SUPER POM IR PAVELDĖJIMO PAVYZDYS



MAVEN PRIKLAUSOMYBĖS

- Maven pakeitė Java priklausomybių valdymą - nebereikia saugoti bibliotekų išeities kodo valdymo sistemoje (SCM) ar panašiai
- Pasiūlyta Maven repozitorijos (angl. repository) sąvoka. Sukurta Maven Central bendruomenės repozitorija
- Pasiūlyta tranzityvios priklausomybės (angl. transitive dependency) sąvoka
- Dažnai įtraukiami išeities kodo ir javadoc artefaktai



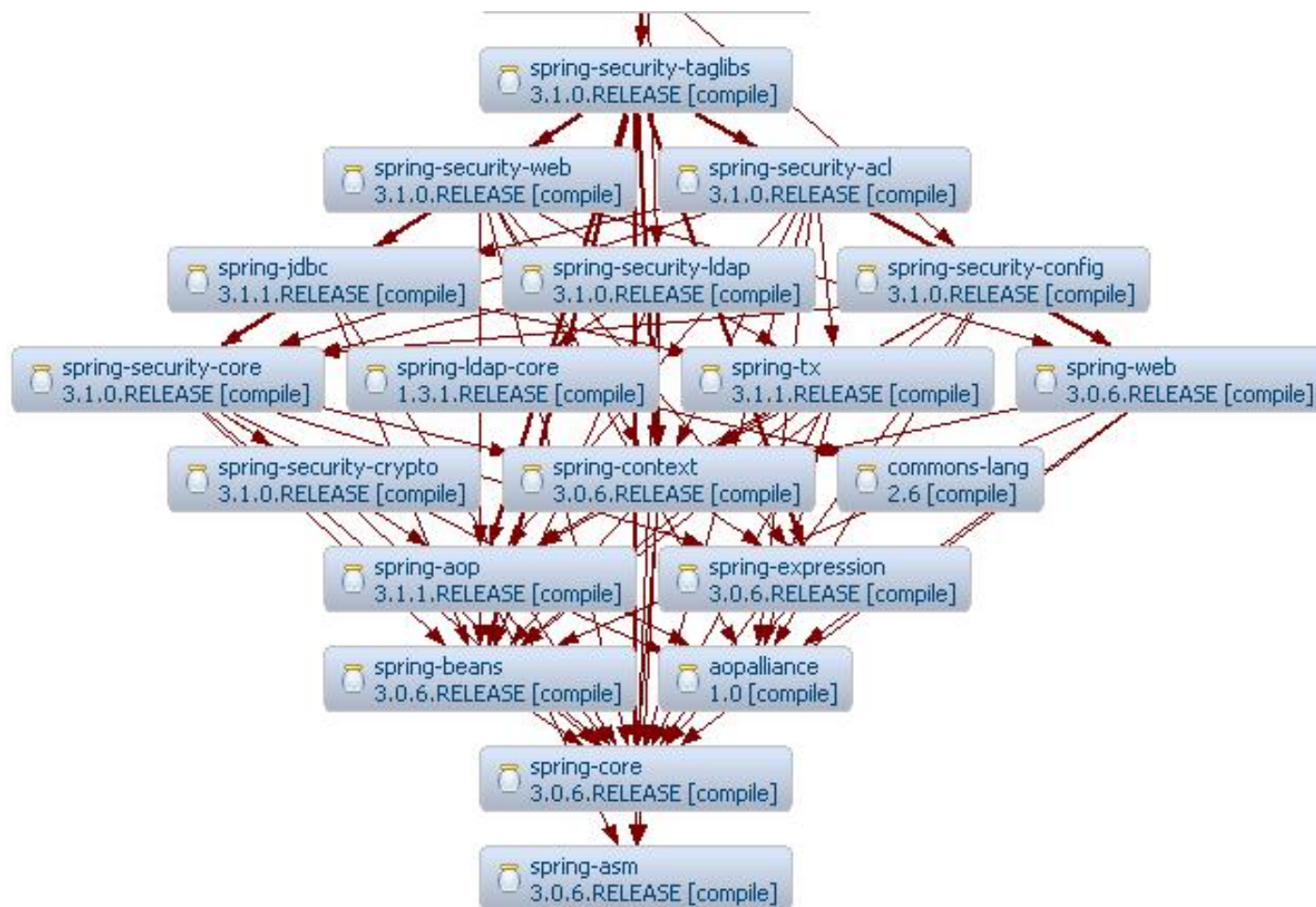
PRIDĖTI PRIKLAUSOMYBĘ

- Priklausomybės aprašą sudaro:
 - GAV (groupId, artifactId, version)
 - Galiojimo sritis (angl. scope) - compile, test, provided. Pagal nutylėjimą naudojama compile
 - Tipas - jar, pom, war, ear, zip ir t.t. Pagal nutylėjimą naudojamas jar
- pom.xml:

```
<project>
...
<dependencies>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.5</version>
    <scope>provided</scope>
  </dependency>
</dependencies>
</project>
```



SPRING PRIKLAUSOMYBIŲ PAVYZDYS



MAVEN REPOZITORIJA

- Priklausomybės yra parsiumčiamos iš repozitorijų naudojant http protokolą
- Parsiustos priklausomybės yra išsaugomos lokaliaje repozitorijoje (pvz. `${user.home}/.m2/repository`)
- Repozitorijoje naudojama `{groupId}/{artifactId}/{version}/{artifactId}-{version}.jar` katalogų struktūra, o groupId `'.'` yra pakeičiamas `'/'`
- Maven Central yra pagrindinė Maven bendruomenės repozitorija <http://repo1.maven.org/maven2>



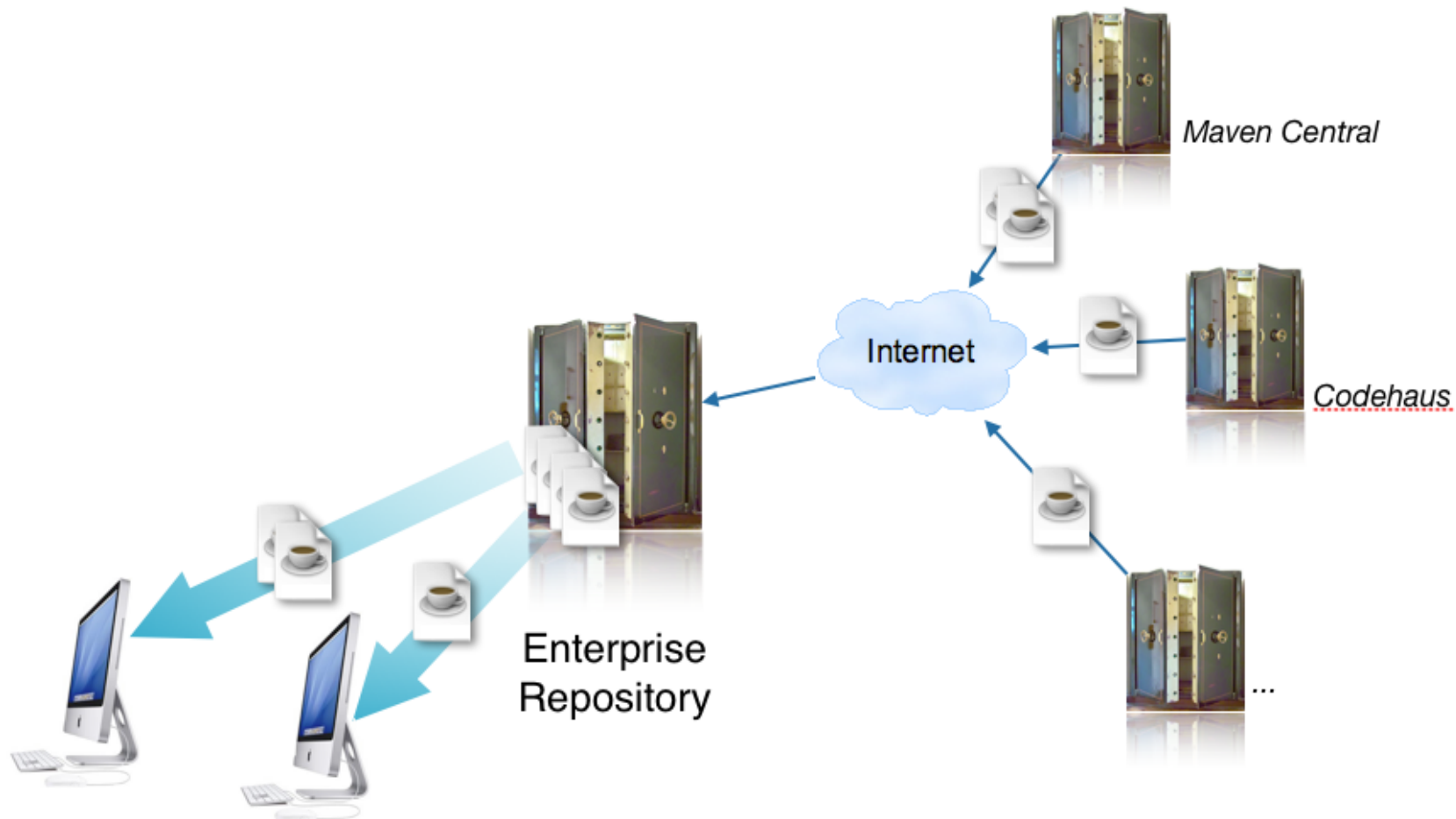
PRIDĖTI REPOZITORIJĄ

- Repozitorijos aprašomos POM
- Repozitorijos gali būti paveldėtos iš tėvo POM
- Momentinių kopijų (angl. snapshot) parsisiuntimas gali būti kontroliuojamas
- pom.xml

```
<project>
...
<repositories>
  <repository>
    <id>lds-main</id>
    <name>LDS Main Repo</name>
    <url>http://code.eisgroup.com/nexus/content/groups/main-repo</url>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </repository>
</repositories>
</project>
```



MAVEN REPOZITORIJŲ PAVYZDYS



TRANZITYVIOS PRIKLAUSOMYBĖS

- Tranzityvi priklausomybė yra tokia priklausomybė, kuri turi būti įtraukta, kai priklausomybę deklaruojantis projektas yra pats kito projekto priklausomybė:
 - ProjectA priklauso nuo ProjectB
 - Jei ProjectC priklauso nuo ProjectA, tai ProjectB yra automatiškai įtraukiamas
- Tik `compile` ir `runtime` galiojimo sritys yra tranzityvios
- Tranzityvios galiojimo sritys yra valdomos naudojant:
 - Pašalinimus (angl. exclusions)
 - Neprivalomą (angl. optional) deklaraciją



PRIKLAUSOMYBĖS PAŠALINIMAS

- Tranzityvi priklausomybė pašalinama naudojant exclusions elementą:

```
<project>
  ...
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>3.0.5.RELEASE</version>
      <exclusions>
        <exclusion>
          <groupId>commons-logging</groupId>
          <artifactId>commons-logging</artifactId>
        </exclusion>
      </exclusions>
    </dependency>
  </dependencies>
</project>
```



NEPRIVALOMA PRIKLAUSOMYBĖ

- Neskleidžia tranzityviai priklausomybės:
 - ProjectA turi neprivalomą priklausomybę nuo ProjectB
 - Jei ProjectC priklauso nuo ProjectA tai ProjectB nebus automatiškai įtraukiamas

```
<project>
  ...
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>3.0.5.RELEASE</version>
      <optional>true</optional>
    </dependency>
  </dependencies>
</project>
```



PRIKLAUSOMYBIŲ VALDYMAS 1

- Java neleidžia naudoti kelių versijų vienu metu
- Ką daryti jei versijos persikerta?
 - Leisti Maven nuspresti, kuria versiją naudoti - sunkiau nuspėjamas rezultatas
 - Valdyti versijas rankiniu būdu
- Priklausomybių versijos valdomos naudojant dependencyManagement elementą
- Kiti panaudojimai:
 - Leisti tėvo POM valdyti versijas
 - Suvienodinti pašalinius



PRIKLAUSOMYBIŲ VALDYMAS 2

- pom.xml priklausomybių be versijos pavyzdys

```
<project>
...
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>3.0.5.RELEASE</version>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
  </dependency>  <!-- Nenurodoma versija! -->
</dependencies>
</project>
```



MAVEN PAPILDINIAI

- Išplečia Maven funkcionalumą
- Identifikacijai naudojamas GAV (groupId, artifactId, version)
- Papildinio naudojimo būdai:
 - Prikabinti prie darinio kūrimo gyvavimo ciklo
 - Iškviešti autonomiškai (angl. standalone)



PRIKABINIMAS PRIE GYVAVIMO CIKLO

- Leidžia papildinį įvykdyti kaip Maven darinio kūrimo dalį
- Papildinio aprašymo elementai naudojami vykdymo konfigūravimui:
 - Phase
 - Goal
 - Configuration



MAVEN PAPILDINIO PAVYZDYS 1

- pom.xml yra prikabinas papildinys maven-enforcer-plugin

```
<project>
...
<build><plugins><plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-enforcer-plugin</artifactId>
  <version>1.0</version>
  <configuration> ... </configuration>
  <executions>
    <execution>
      <id>execute</id>
      <phase>validate</phase>
      <goals><goal>enforce</goal></goals>
      <configuration> ... </configuration>
    </execution>
  </executions>
</plugin></plugins></build>
</project>
```



PAPILDINIO VALDYMAS

- Leidžia sukonfigūruoti sekančius papildinio elementus, jo nevykdant:
 - Version
 - Configuration
 - Executions
- Vykdymui naudojamas:
 - įprastas papildinio įrašas
 - papildinio autonominė komanda



PAPILDINIO VALDymo PAVYZDYS

- pom.xml papildinio konfigūracija maven-enforcer-plugin

```
<project><build>
<pluginManagement>
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-enforcer-plugin</artifactId>
    <version>1.0</version>
    <configuration>
      ...
      <ignoreCache>true</ignoreCache>
    </configuration>
  </plugin>
</plugins>
</pluginManagement>
<plugins><plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-enforcer-plugin</artifactId>
</plugin></plugins>
</build></project>
```



PAPILDINIO PAVELDĖJIMAS

- Papildinys paveldi `pluginManagement` konfigūraciją
- Papildinys ir `pluginManagement` gali paveldėti konfigūraciją iš tėvo POM
- Taip pat leidžiama pakeisti paveldėtą konfigūraciją



PAPILDINIO AUTONOMINIS ĮVYKDYMAS

- Papildinys gali būti išviečiamas naudojant komandinę eilutę:
 - `GroupId:ArtifactId:Version:Goal`
 - Naudojama `pluginManagement` konfigūracija
 - Konfigūracija gali būti nurodyta saybėmis

```
$ mvn org.apache.maven.plugins:maven-enforcer-plugin:1.0:enforce
```

Jei papildinys sukonfigūruotas POM arba `settings.xml` tai iškvietimas gali būti sutrumpintas:

```
$ mvn enforcer:enforce
```



MAVEN SAVYBĖS

- Savybės tai tarsi klijai, kurie suriša konfigūraciją
- Savybes galima nurodyti sekančiose vietose:
 - elementas POM byloje
 - Sisteminės savybės
 - POM struktūra
- Savybių reikšmę galima panaudoti įvairiausiose vietose:
 - Versijos konfigūravimui
 - Papildinio konfigūravimui
 - Resursų filtravimui
- Savybėse nurodomos tik primityvios reikšmės



POM SAVYBĒS

- pom.xml nurodytos savybės ir jų panaudojimas

```
<project>
...
<properties>
  <skipEnforcer>true</skipEnforcer>
  <enforcerVersion>1.0</enforcerVersion>
</properties>
<build><plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-enforcer-plugin</artifactId>
    <version>${enforcerVersion}</version>
    <configuration>
      <skip>${skipEnforcer}</skip>
    </configuration>
  </plugin>
</plugins></build>
</project>
```



SISTEMINĖS SAVYBĖS

Nurodomos komandinėje eilutėje panaudojus “-D”:

```
$ mvn clean install -Dmaven.test.skip=true -DskipTests=true
```

Sisteminės savybės turi viršenybę prieš POM savybės



POM STRUKTŪROS SAVYBĒS

- Savybēs gali būti paveldētos iš POM struktūros
- POM elementai yra savybių raktai:
 - Išraiška `${project.version}`

```
<project><version/></project>
```

- Išraiška `${project.artifactId}`

```
<project><artifactId/></project>
```

- Išraiška `${project.build.sourceDirectory}`

```
<project><build><sourceDirectory/></build></project>
```



POM STRUKTŪROS SAVYBĖS

- Specialios savybės:
 - `${basedir}` - einamojo projekto katalogas
 - `${maven.build.timestamp}` - darinio kūrimo pradžios laikas



SAVYBIŲ PAVELDĖJIMAS IR PERKROVIMAS

```
<project> <!-- čia yra tėvinis projektas -->
...
<properties>
  <skipTests>true</skipTests>
  <skipEnforcer>${skipTests}</skipEnforcer>
</properties>
</project>

<project>
  <parent>
    ... <!-- tėvinio projekto GAV -->
  </parent>
  <properties>
    <skipTests>false</skipTests>
  </properties>
</project>
```



RESURSŲ FILTRAVIMAS

- Projekto resursai gali naudoti savybes
- Resursai filtruojami process-resources fazėje
- Filtravimas gali būti išjungtas nurodytiems resursų katalogams

```
<project>
  ...
  <properties>
    <someProperty>SomeValue</someProperty>
  </properties>
</project>
```

Tekstinė byla
/src/main/resources kataloge:
`${someProperty}`

Filtravimas

Tekstinė byla
/src/main/resources kataloge:
SomeValue



MAVEN PROFILIAI

- Leidžia aktyvuoti rinkinį alternatyvių konfigūracijų
- Gali būti naudojami nurodyti:
 - savybes
 - priklausomybes
 - papildinius
 - kita
- Paveldi ir išplėčia bazinę konfigūraciją



PROFILIO PAVYZDYS 1

```
<project>
  ...
  <profiles>
    <profile>
      <id>enforcer</id>
      <activation/>
      <properties>
        <enforcer.skip>false</enforcer.skip>
      </properties>
    </profile>
  </profiles>
</project>
```



PROFILIO PAVYZDYS 2

```
$ mvn exec:exec - vykdo Java programą atskirame procese.  
$ mvn exec:java - vykdo Java programą toje pačioje VM, kaip ir Maven  
$ mvn exec:java -Dexec.mainClass="com.example.Main" \  
> [-Dexec.args="argument1"] ...
```

```
<profiles><profile>  
  <id>run</id>  
  <activation><activeByDefault>true</activeByDefault></activation>  
  <build><plugins><plugin>  
    <groupId>org.codehaus.mojo</groupId>  
    <artifactId>exec-maven-plugin</artifactId>  
    <executions>  
      <execution>  
        <goals><goal>java</goal></goals>  
        <phase>runtime</phase>  
      </execution></executions>  
    <configuration>  
      <mainClass>ClassWithTheMain</mainClass>  
    </configuration>  
  </plugin></plugins></build>  
</profile></profiles>
```



PROFILIO AKTYVAVIMAS

- Profilis gali būti aktyvuotas:
 - pagal nutylėjimą
 - tiesiogiai pagal profilio vardą
 - priklausomai nuo savybės
 - priklausomai nuo operacinės sistemos
 - priklausomai nuo bylos egzistavimo



AKTYVAVIMAS KOMANDINĖJE EILUTĖJE

```
$ mvn clean install -P enforcer
```

Keletas profilio identifikatorių yra atskirami kableliu

```
$ mvn clean install -P enforcer,kitas-profilis
```



ARCHETIPAS

- Archetipas yra Maven projektų šablonų priemonių komplektas
- Bendresnis archetipo apibrėžimas būtų autentiškas šablonas ar modelis, kurį naudojant yra gaminami kiti tos pačios rūšies dalykai



MAVEN PROJEKTO SUKŪRIMAS

- Paprasto projekto sukūrimas:

```
$ mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app \
> -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

- Arba galima atsakinėti į klausimus terminale:

```
$ mvn archetype:generate
```

- Eclipse IDE projekto failų sukūrimas:

```
$ mvn eclipse:eclipse
```

- Idea IDE projekto failų sukūrimas:

```
$ mvn idea:idea
```



NAUDINGOS NUORODOS

- <http://maven.apache.org/users/index.html>
- <http://maven.apache.org/pom.html>
- <http://maven.apache.org/guides/getting-started/index.html>
- <http://search.maven.org/>
- <https://github.com/sonatype/maven-example-en>



UŽDUOTIS 4 - ARCHETIPAS

```
$ mvn archetype:generate -DgroupId=lt.mokymai \  
-DartifactId=SecondMavenProject -Dpackage=lt.mokymai.maven \  
-Dversion=1.0-SNAPSHOT \  
-DarchetypeArtifactId=maven-archetype-quickstart  
$ cd SecondMavenProject
```

```
| pom.xml  
| src  
|   | main  
|   |   | java  
|   |   |   | lt  
|   |   |   |   | mokymai  
|   |   |   |   |   | maven  
|   |   |   |   |   |   | App.java  
|   | test  
|   |   | java  
|   |   |   | lt  
|   |   |   |   | mokymai  
|   |   |   |   |   | maven  
|   |   |   |   |   |   | AppTest.java
```



UŽDUOTIS 4 - ECLIPSE

- Sukurti Eclipse projekto failus:

```
$ mvn eclipse:eclipse // idea:idea
```

- Peržiūrėti SecondMavenProject projekto katalogą:

```
$ ls -a  
.  ..  .classpath  pom.xml  .project  src
```

- Projektą importuoti į Eclipse
 - senesnėse eclipse: Window > Preferences -> Java > Build Path > Classpath Variables -> New M2_REPO = /home/username/.m2/repository
 - kodo atitikimas standartui: Window > Preferences -> Java Compiler -> Compliance -> 1.8



UŽDUOTIS 4 - KOMPILIAVIMAS IR VYKDYMAS

```
$ mvn clean compile
```

- Panaudoti exec:java papildinį lt.mokymai.maven.App klasės įvykdymui:

```
$ mvn exec:java -Dexec.mainClass="lt.mokymai.maven.App"
[INFO] --- exec-maven-plugin:1.3.2:java (default-cli) @ SecondMavenProject ---
[WARNING] Warning: killAfter is now deprecated. Do you need it ? Please comment on MEXEC-
Hello World!
[INFO] -----
[INFO] BUILD SUCCESS
```



UŽDUOTIS 4 - KOMPILIAVIMAS IR VYKDYMAS

- jeigu nesikompiliuos, galbūt neturite jdk 8 virtualioje mašinoje

```
$ java -version
openjdk version "10.0.2" 2018-07-17
$ sudo apt-get install openjdk-8-jdk
$ sudo update-alternatives --list java
/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
$ sudo update-alternatives --set java \
    /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
$ java -version
openjdk version "1.8.0_162"
```



UŽDUOTIS 4 - TESTAVIMAS

```
$ mvn clean test
```

```
-----  
T E S T S  
-----
```

```
Running lt.mokymai.maven.AppTest
```

```
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.018
```

```
Results :
```

```
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```



UŽDUOTIS 4 - CLEAN IR COMPILE

```
$ mvn clean
$ ls -a
.  ..  .classpath  pom.xml  .project  .settings  src
```

- Pastaba. Pagrindiniame projekto kataloge nėra target katalogo.

```
$ mvn clean compile
$ ls -a
.  ..  .classpath  pom.xml  .project  .settings  src  target
$
$ ls -a target/
.  ..  classes
```

- Pastaba. Pagrindiniame projekto kataloge sukurtas target katalogas.



UŽDUOTIS 4 - TEST

```
$ mvn clean test
$ ls -a
.  ..  .classpath  pom.xml  .project  .settings  src  target

$ ls -a target/
.  ..  classes  surefire  surefire-reports  test-classes
```



UŽDUOTIS 4 - PACKAGE

```
$ mvn clean package
$ ls -a
. .. .classpath pom.xml .project .settings src target
$ ls -a target/
. .. classes maven-archiver SecondMavenProject-1.0-SNAPSHOT.jar
surefire surefire-reports test-classes
$ ls ~/.m2/repository/
antlr asm backport-util-concurrent biz classworlds commons-cli
commons-collections commons-io commons-lang dom4j jdom jline junit net org oro xml-api
```

- Pastaba: target kataloge sukurtas SecondMavenProject-1.0-SNAPSHOT.jar maven projekto artefaktas, tačiau jis nėra perkeltas į lokalią maven repozitoriją (nėra katalogo ~/.m2/repository/lt/mokymai/)



UŽDUOTIS 4 - INSTALL

```
$ mvn clean install
$ ls -a target/
. .. classes maven-archiver SecondMavenProject-1.0-SNAPSHOT.jar
surefire surefire-reports test-classes
$ ls ~/.m2/repository/
antlr lt backport-util-concurrent classworlds commons-collections
commons-lang jdom junit net oro asm biz commons-cli commons-io dom4j jline org xml-api
$ ls ~/.m2/repository/lt/mokymai/SecondMavenProject/
1.0-SNAPSHOT maven-metadata-local.xml
```

- Pastaba: target kataloge sukurtas SecondMavenProject-1.0-SNAPSHOT.jar maven projekto artefaktas ir jis yra perkeltas į lokalią maven repozitoriją.



UŽDUOTIS 5 - MAVEN MULTI PROJEKTAI

- Parsiusti pavyzdinį maven projektą

```
$ wget http://books.sonatype.com/mvnex-book/mvnex-examples.zip
```

- Išarchyvuoti ir pasirinkti projekto katalogą:

```
$ unzip mvnex-examples.zip  
$ cd mvnexbook-examples-1.0/ch-multi-spring
```

- Pagaminti darinį (angl. build):

```
$ mvn clean install
```

- Peržiūrėti projekto aprašus (pom.xml bylas)
- Importuoti projektus (mvnexbook-examples-1.0/ch-multi-spring) į Eclipse.



UŽDUOTIS 6 - JAVA KLASĖS KAIP SERVERIS

- Sukurti naują Maven projektą
 - galima ir su maven-archetype-quickstart
- Perkelti Java kurso praktikos klases į projektą arba sukurti naujas klases
- Pagaminti projekto darinį
- Įvykdyti pagrindinę Java klasę main Maven priemonėmis
- Įvykdyti pagrindinę Java klasę iš Eclipse aplinkos

Video pvz: <http://youtu.be/8hvtZxAlNyw>



UŽDUOTIS 6 - JAVA KLASĖS KAIP SERVERIS

- Sukurti antrą maven projektą tokiu pačiu principu ir jį panaudoti anksčiau kurtame
 - pridėti antrojo projekto pom.xml GAV aprašą prie pirmojo projekto dependencies
- dabar pirmasis projektas naudoja antrąjį
 - perkeliame visas java klases iš pirmojo į antrąjį
- Paleidžiame `mvn clean install` antrajam
- Tuomet paleidžiame pirmajam
- Dabar pabandome paleisti Main klasę pirmajame
 - nors pirmasis pats neturi klasių, bet tranzityviai gauna jas iš antrojo ir main metodas pasileidžia



SPRING BOOT APLIKACIJA



AKADEMIJA.IT

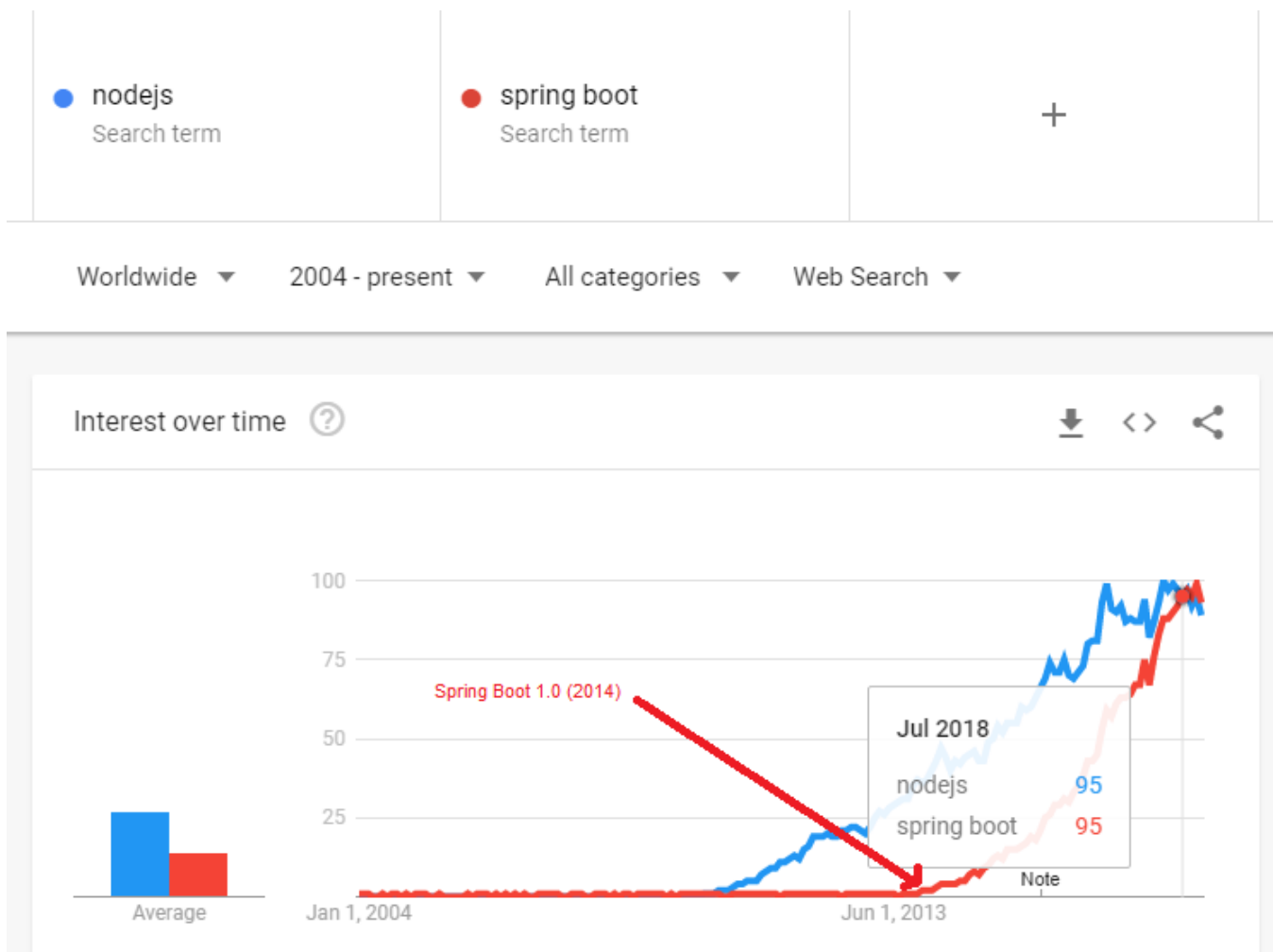
INFOBALT IR TECH CITY

SPRING BOOT APLIKACIJA

- kodėl Spring Boot?
- kas yra Servlet
 - java klasės
 - technologija
 - specifikacija
- kas yra Spring
 - java klasės
 - biblioteka
- kas yra Spring Boot
 - java klasės
 - Spring biblioteka



REST KARKASŲ KOVOS



REST KARKASŲ KOVOS

- Kodėl naudojamas Spring Boot ?
 - Node.js per anksti: didelės kompanijos nori stabilumo
 - jokių didelių privalumų: ką gali Node.js, tą gali ir java
 - ekosistema: java išvystyta, daug bibliotekų, palaiko VISAS platformas, DB, enterprise lygio palaikymas
 - jms/webservisai/rest/big data
 - Java turi statinius tipus
 - Node.js servisai yra vienoje gijoje (thread)
 - t.y. nepalaiko multi-threading
 - Javascript einamasis palaikymas yra košmaras
 - Javascript sunkiai dokumentuojamas



SERVLET

- Pagal JavaDoc aprašymą:

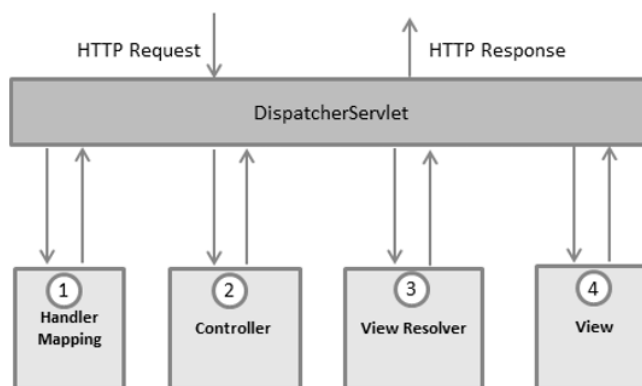
Servletas - tai maža Java programa, kuri veikia žiniatinklio serveryje. Servletai gauna ir apdoroja užklausas, gautas iš žiniatinklio klientų, dažniausiai naudojant HTTP protokolą”.

- Tai yra klasė, kuri realizuoja `javax.servlet.Servlet` interfeisą



SERVLET

- skirtas išplėsti serverio galimybes
- gali būti sukonfigūruoti keli servletai skirtingiems keliams
 - /kelias1 -> Servlet1
 - /kelias2 -> Servlet2
- Spring Boot servlet'as sugeba apdoroti HTTP užklausas, suprasti REST tipo užklausas ir atiduoti statinį turinį



UŽDUOTIS 7 - SPRING BOOT ARCHETIPAS

- kol kas kūrėme praktiškai tuščią projektą, su pačia paprasčiausia aplikacija, kuri nedirba per tinklą
 - java klasė su main metodu
- toliau sukursime beveik pilnai paruoštą naudoti pavyzdinę Spring Boot aplikaciją

```
$ mvn archetype:generate -DgroupId=it.akademija \
-DartifactId=hello-world-calc \
-DarchetypeGroupId=am.ik.archetype \
-DarchetypeArtifactId=spring-boot-blank-archetype \
-DarchetypeVersion=1.0.6 -DinteractiveMode=false
```



UŽDUOTIS 7 - SPRING BOOT ARCHETIPAS

- Maven nustatymai koduotei UTF-8 kodui bei generuotiems failams ir Java 8

```
<properties>
  <!-- Kodas užkoduotas universalia koduote UTF-8 -->
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  <!-- Kodas skirtas Java 8 -->
  <java.version>1.8</java.version>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

- dėti į pom.xml



UŽDUOTIS 7 - SPRING BOOT ARCHETIPAS

- Paruošiamė naudojimui IDE

```
$ mvn eclipse:eclipse
```

- senesnėse eclipse:
 - Window > Preferences -> Java > Build Path > Classpath Variables -> New M2_REPO =
/home/<username>/.m2/repository
- kodo atitikimas standartui
 - Window > Preferences -> Java Compiler -> Compliance -> 1.8



UŽDUOTIS 7 - SPRING BOOT ARCHETIPAS

- Paruošiamo darinį (build+tests)

```
$ mvn clean install  
$ mvn test
```

- Paleidžiame Spring Boot aplikaciją

```
$ mvn spring-boot:run -Drun.jvmArguments='-Dserver.port=8081'
```

- Pastaba: jeigu prieš tai paleidome Tomcat, 8080 portas jau užimtas, todėl naudojame 8081 arba bet kurį laisvą



UŽDUOTIS 7 - SPRING BOOT ARCHETIPAS

- Patikrinkime ar veikia naršyklėje
 - <http://localhost:8081/calc?left=1&right=2>
 - pastaba: serveris ir yra viena aplikacija
- Tokios JAR aplikacijos negalime įdėti į Tomcat serverį
 - reikalingas WAR failas
- Tai gal tiesiog pakeisti/įrašyti pom.xml packaging?

```
<packaging>war</packaging>
```

- nesuveiks.. reikia perkonfigūruoti Spring Boot



UŽDUOTIS 7 - SPRING BOOT ARCHETIPAS

- pom.xml pridedame priklausomybę

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-tomcat</artifactId>
  <scope>provided</scope>
</dependency>
```

- App.java perdarome taip, kad extendint'ų
SpringBootServletInitializer Spring Boot servlet'ą

```
@SpringBootApplication
public class App extends SpringBootServletInitializer {
    public static void main(String[] args) {
        SpringApplication.run(App.class, args);
    }
    @Override
    protected SpringApplicationBuilder configure(
        SpringApplicationBuilder builder) {
        return builder.sources(App.class);
    }
}
```



UŽDUOTIS 7 - SPRING BOOT ARCHETIPAS

- Paleidus `mvn clean install` target'e atsiras `.war`
 - įdėkite jį į anksčiau atsisiųstą tomcat serverį
 - ar vis dar veikia `/calc` kalkuliatorius?
- Paprasčiau ir greičiau galima paleisti įdėtinį tomcat7

```
$ mvn org.apache.tomcat.maven:tomcat7-maven-plugin:2.2:run-war \
> -Dmaven.tomcat.port=8081
```

- `spring-boot:run` aplikaciją padaro serveriu, o `tomcat7:run-war` aplikaciją paleidžia serveryje kaip vieną iš aplikacijų
- todėl prieiname ne `..8081/calc`, o `..8081/<appName>/calc`



UŽDUOTIS 8 - REACT PRIJUNGIMAS

- Pirmiausia reikia paruošti React aplikaciją. Tam, kad veiktų ir Spring Boot, ir Tomcat, turime į `package.json` pridėti

```
"homepage": ". /"
```

- Įprastai čia turėtų būti tikros svetainės adresas, pvz.
<http://svetaine.lt/kelias/iki/jos>



UŽDUOTIS 8 - REACT PRIJUNGIMAS

- Spring Boot aplikacijoje pervadiname
 - `src/main/resources/templates` katalogą į
 - `src/main/resources/public` ir jo turinį ištriname
- Katalogo `build` turinį iš React aplikacijos tiesiog nukopijuojame į Spring Boot aplikacijos `src/main/resources/public` katalogą



UŽDUOTIS 8 - REACT PRIJUNGIMAS

- Kol kas kelią iki ' / ' blokuoja java kodas, todėl iš failo `src/main/java/it/akademija/HelloController.java` turime surasti ir ištrinti šį kodą:

```
@RequestMapping("/")
String hello() {
    return "Hello World!";
}
```

- Jį ištrynus, React aplikacija veiks po keliu /, bet tuo pačiu veiks ir /calc servisas
- Tuomet neveiks HelloControllerTest testas: jį reikia ištrinti, arba laikinai netestuoti su `mvn -DskipTests`



UŽDUOTIS 8 - REACT PRIJUNGIMAS

- Sukonfigūruojame Spring Boot pom.xml, kad statiniai resursai atsinaujintų neperkrovus puslapio

```
<build>
<plugins>
  <plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <configuration>
      <addResources>true</addResources>
    </configuration>
  </plugin>
</plugins>
</build>
```

- Pastaba: tai veiks tik su Spring Boot, bet ne Tomcat
- Paleidimui naudokite tas pačias jau išmoktas komandas



KITOJE PASKAITOJE

Spring. Maven priklausomybių migracija



AKADEMIJA.IT

INFOBALT IR TECH CITY