Maven – инструмент сборки, но не только

Алексей Сорокин
HeadHunter :: Пенза

Средства сборки

- Shell-скрипт
- Утилита make
- Automake
- Apache Ant
- Maven

Часть 1.

- До Maven
- Основы Maven
- Преимущества Maven

Различия в структуре

Проект 1

```
/conf
/dist
/lib
/src
/test
/test-conf
build.xml
```

Проект 2

```
/bin
/deps
/source
    /classes
    /xml
/unit-tests
build.xml
```

Различия в командах

Прогон тестов

```
>ant run-tests
```

Упаковка

```
>ant build-dist
```

Прогон тестов

```
>ant unit-tests
```

Упаковка

```
>ant build-jar
```

Built by Maven

```
svn checkout http://company.org/project

/src
    /main
        /java
    /test
        /java
pom.xml

>mvn install
```

Подход в Maven

Ant, make и похожие иструменты:

КАК *собрать проект из исходного кода?*

Maven:

ЧТО *представляет из себя проект?*

Maven полностью **поддерживает** Ant

Maven привносит **общий интерфейс** в процесс сборки

Минимальная конфигурация

```
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.company</groupId>
    <artifactId>project</artifactId>
    <version>1.0</version>
</project>
```

Аналогичный build.xml

```
<project name="project" default="dist" basedir=".">
    <property name="src" location="src/main/java"/>
    <property name="build" location="target/classes"/>
    <property name="dist" location="target"/> <target name="init">
    <tstamp/>
    <mkdir dir="${build}"/>  </target>
    <target name="compile" depends="init" >
        <javac srcdir="${src}" destdir="${build}"/>
    </target>
    <target name="dist" depends="compile">
        <mkdir dir="${dist}/lib"/>
        <jar jarfile="${dist}/lib/project-${DSTAMP}.jar" basedir="$
       {build}"/>
    </target>
    <target name="clean">
        <delete dir="${build}"/>
        <delete dir="${dist}"/>
    </target>
</project>
```

Соглашения

/src

- **/main**
  - **/java**
  - **/resources**
- **/test**
  - **/java**
  - **/resources**

/target

- **/classes**
- **/test-classes**

имя получаемого jar-файла: **${artifactId}-${version}**

Модель проекта

Project Object Model (POM)

Взаимосвязи

- Координаты проекта
- Многомодульность
- Наследование
- Зависимости

Сборка

- Структура папок
- Расширения
- Ресурсы
- Плагины

Общая информация (разработчики, репозиторий исходного кода, ...)

Среда сборки, свойства, профили, ...

Координаты проекта

```
<groupId>org.company</groupId>
<artifactId>project</artifactId>
<version>1.0</version>
```

Преимущества модели

- Управление зависимостями
- Возможность построения репозиториев
- Плагины как повторное использование логики сборки
- Интеграция с другими инструментами
- Поиск артефактов

Зависимости и репозитории

```
<dependency>
      <groupId>org.apache.lucene</groupId>
      <artifactId>lucene-core</artifactId>
      <version>2.3.2</version>
</dependency>
```

/org

/apache

/lucene

/lucene-core

/2.3.2

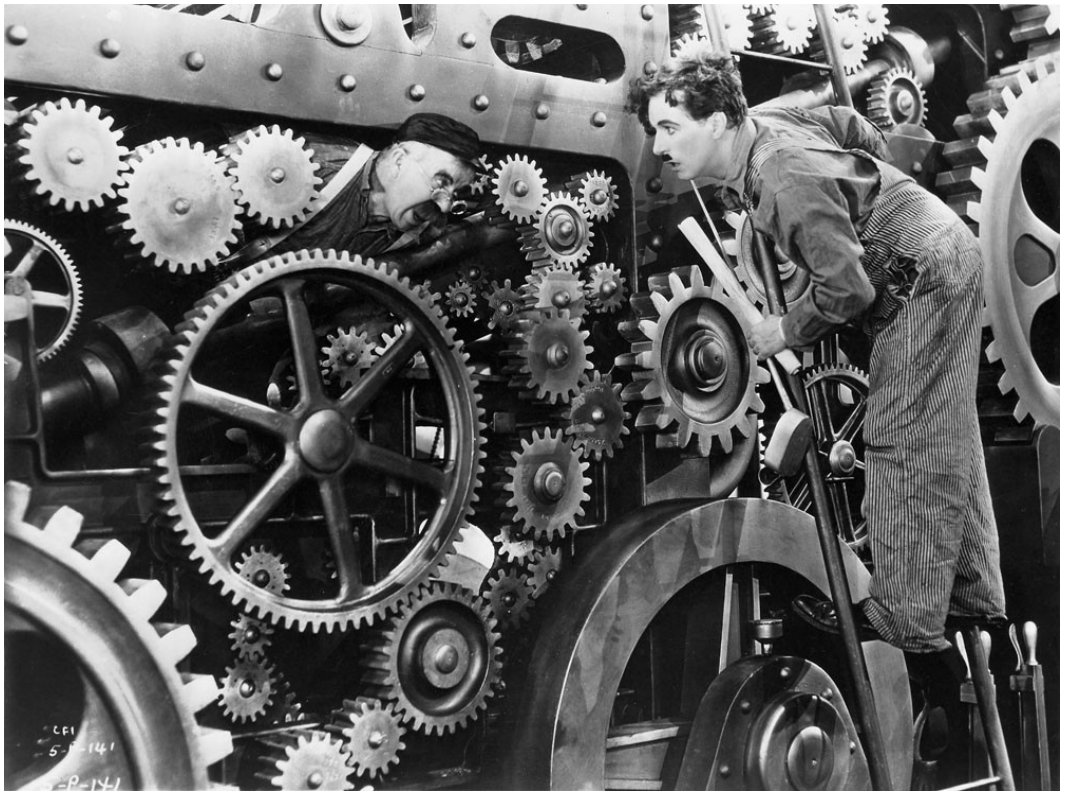
lucene-core-2.3.2.jar

lucene-core-2.3.2-sources.jar

/2.4.0

Жизненный цикл сборки

validate
generate-sources
process-sources
generate-resources
process-resources
compile
process-classes
generate-test-sources
process-test-sources
generate-test-resources
process-test-resources
test-compile
test
prepare-package
package
pre-integration-test
integration-test
post-integration-test
verify
install
deploy



Плагины

- Основные
  - clean
  - compiler
  - deploy
  - install
  - resources
  - surefire
- Типы упаковки
  - jar
  - war
  - ear
- интрументы
  - antrun
  - assembly
  - dependency
  - help
  - release
  - scm
  - source
- IDE
  - eclipse
  - idea

Плагины в цикле сборки

```
<package>jar</package>
```

validate
generate-sources
process-sources
generate-resources
process-resources → resources:resources
compile → compiler:compile
process-classes
generate-test-sources
process-test-sources → resources:testResources
generate-test-resources
process-test-resources
test-compile → compiler:testCompile
test → surefire:test
prepare-package
package → jar:jar
pre-integration-test
integration-test
post-integration-test
verify
install → install:install
deploy → deploy:deploy

Часть 2.

- Перевод проектов на Maven
- Использование Maven в IDE
- Сборка не-Java артефактов

Super POM

```
...
<sourceDirectory>src/main/java</sourceDirectory>
<testSourceDirectory>src/test/java</testSourceDirectory>
<outputDirectory>target/classes</outputDirectory>
<testOutputDirectory>target/test-classes</testOutputDirectory>
<finalName>${project.artifactId}-${project.version}</finalName>
<resources><resource>
    <directory>src/main/resources</directory>
</resource></resources>
<testResources><testResource>
    <directory>src/test/resources</directory>
</testResource></testResources>
...
```

IDE

- Плагины Maven
  - `mvn eclipse:eclipse`
  - `mvn idea:idea`
- Плагины IDE
  - m2eclipse
  - Q for Eclipse
  - Module for NetBeans

Не-Java артефакты

- Flex 2.0 plugin
  - swf
  - swc
- Native plugin
  - dll
  - exe
  - o
  - so
- DEB plugin
- RPM plugin

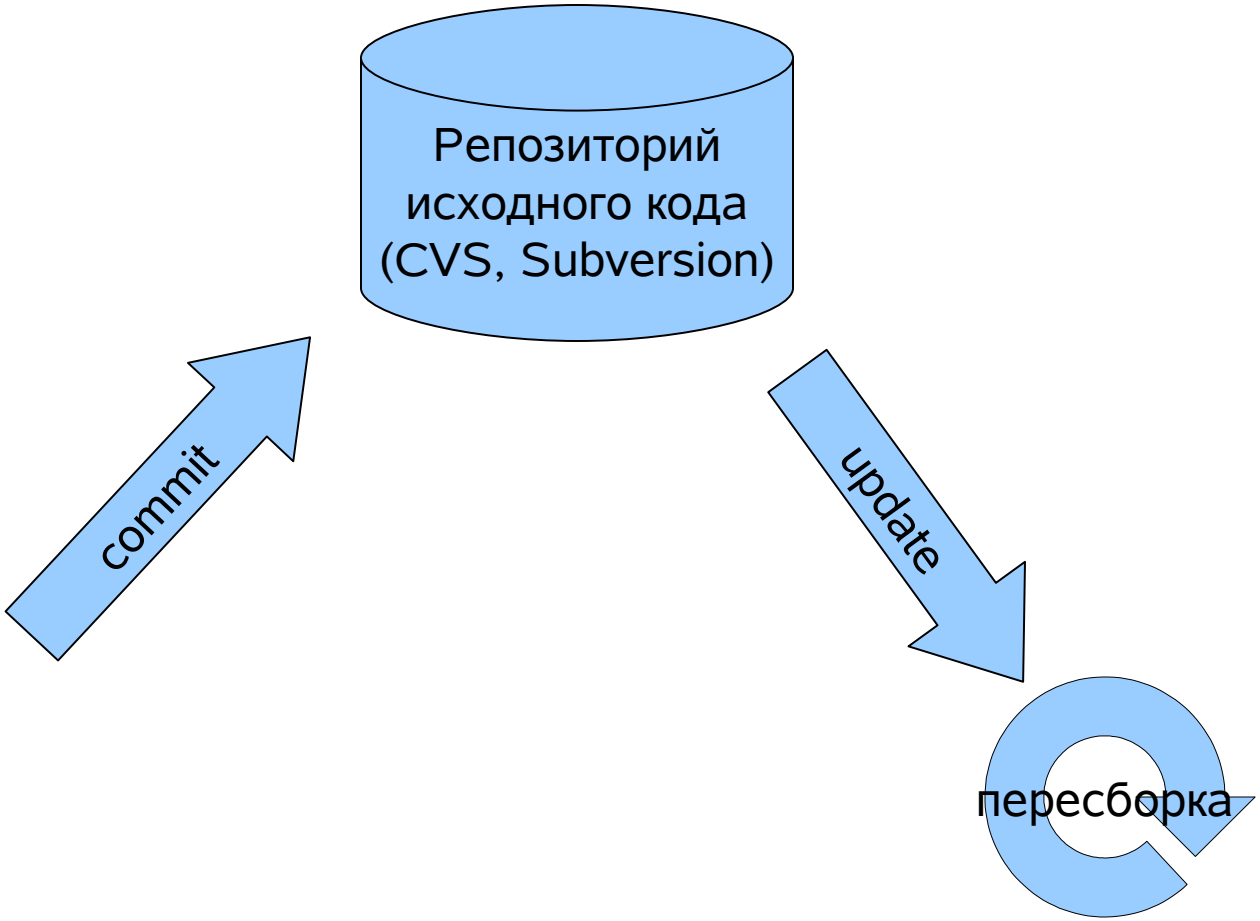
Часть 3.

- Интеграция в **ПРОЦЕССЫ**
- Построение **ИНФРАСТРУКТУРЫ**

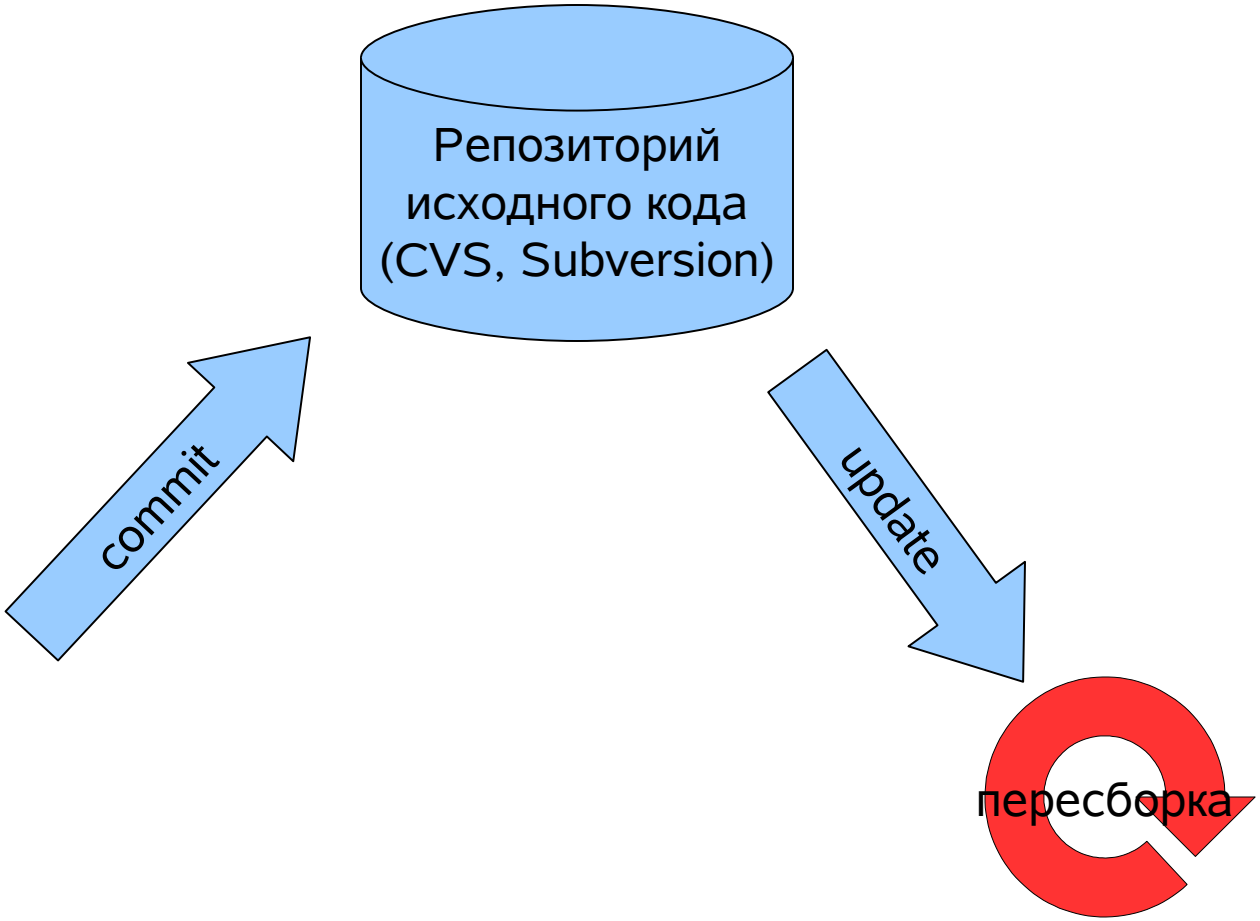
Примеры процессов

- Прогон всех модульных тестов при полной сборке
- Постоянная интеграция
- Фиксация сборки (выпуск релиза)

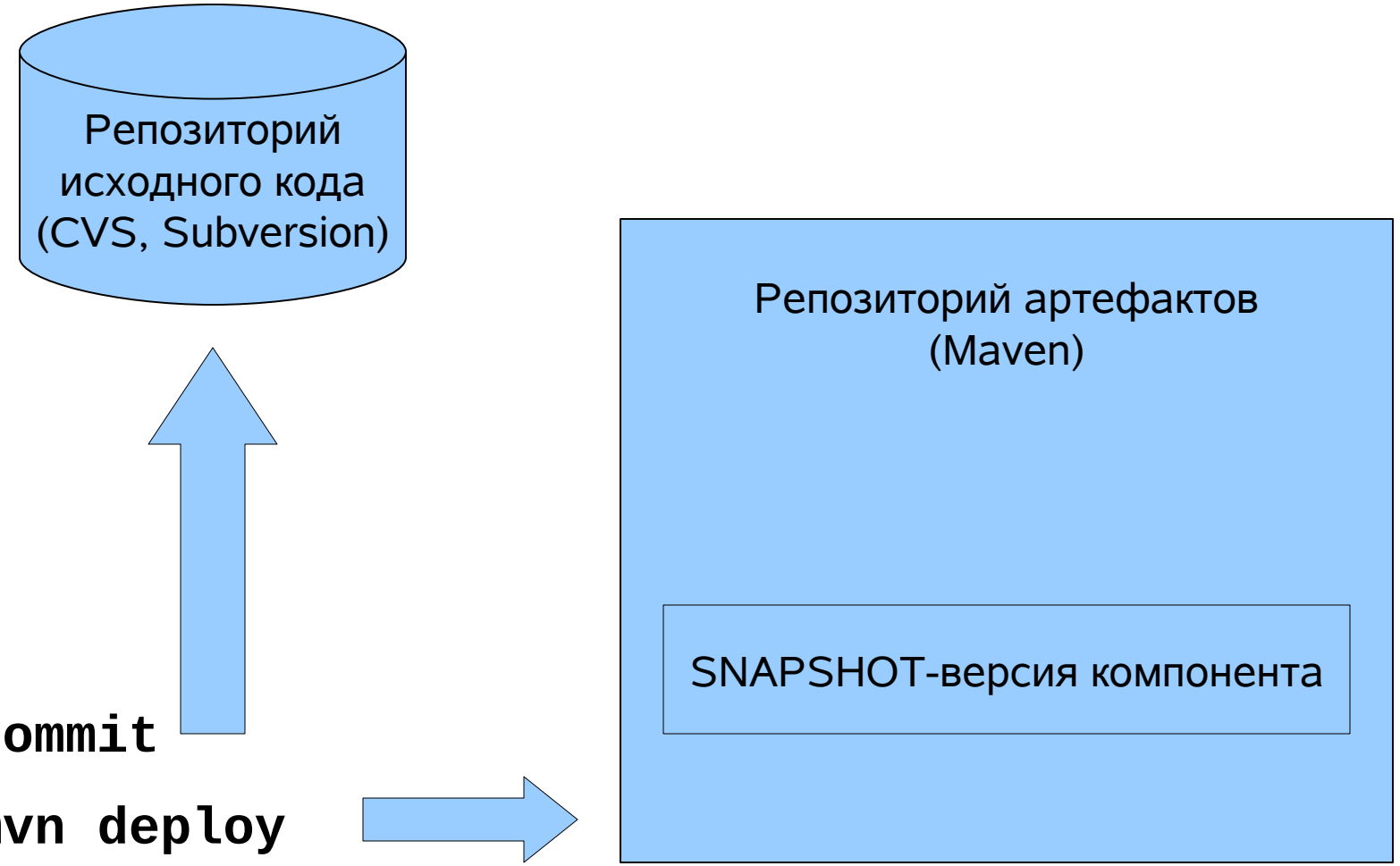
Постоянная интеграция



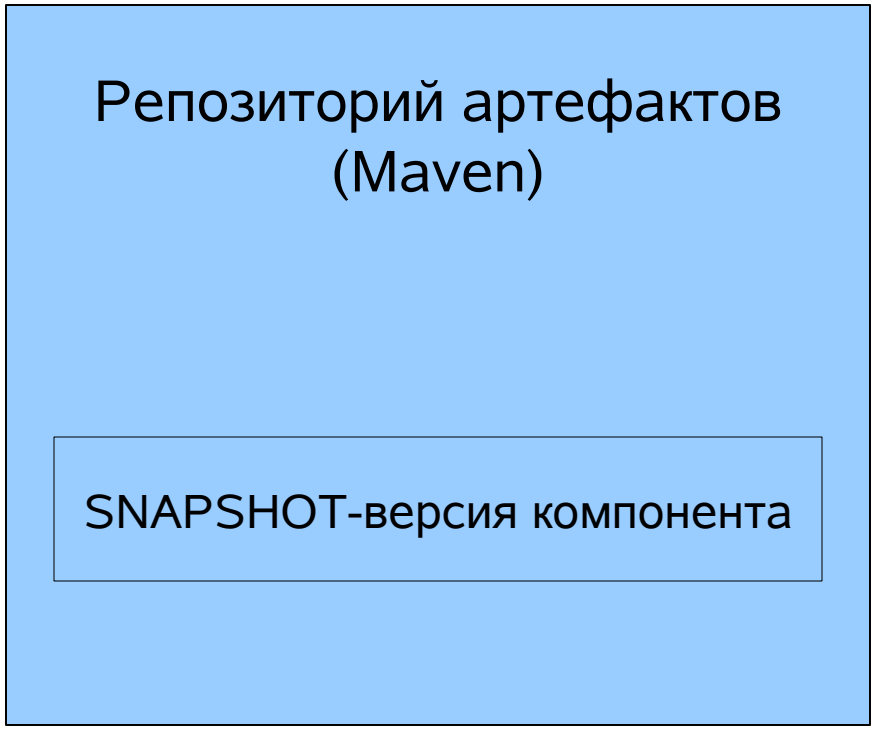
Постоянная интеграция



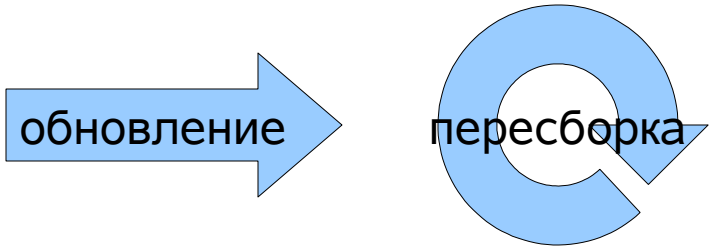
Постоянная интеграция



Постоянная интеграция




```
> mvn install
```



Вопросы?

- Можно задать в зале
- А можно связаться со мной:
  - alsor.net@gmail.com