Java Power Tools in Action

John Ferguson Smart  
Principle Consultant  
Wakaleo Consulting
Overall Presentation Goal

Learn how to improve and automate your development process using tools like Maven, Hudson, and many others.
Speaker’s qualifications

- John is an experienced consultant specialized in Enterprise Java, Web Development, and Open Source technologies
- John is author of Java Power Tools
- John also frequently writes articles for sites such as JavaWorld, DevX and Java.net, and blogs on Java.net
- John speaks frequently at Conferences and Java User Groups on Java development tools and best practices
Overall Presentation Goal

Learn how to improve and automate your development process using tools like Maven, Hudson, and many others.
HERE is Edward Bear, coming downstairs now, bump, bump, bump, on the back of his head, behind Christopher Robin. It is, as far as he knows, the only way of coming downstairs, but sometimes he feels that there really is another way, if only he could stop bumping for a moment and think of it.

-- A. A. Milne
Agenda

What we will cover today:
Agenda

What we will cover today:

- Industrializing your build process
Agenda

- What we will cover today:
  - Industrializing your build process
  - Improving your release management strategy
What we will cover today:

- Industrializing your build process
- Improving your release management strategy
- Automate the build process
Agenda

What we will cover today:

- Industrializing your build process
- Improving your release management strategy
- Automate the build process
- Better testing practices
Agenda

What we will cover today:

- Industrializing your build process
- Improving your release management strategy
- Automate the build process
- Better testing practices
- Automate the deployment process
Agenda

What we will cover today:

- Industrializing your build process
- Improving your release management strategy
- Automate the build process
- Better testing practices
- Automate the deployment process
- Monitoring code coverage and code quality metrics
An improved build process
An improved build process

Industrializing your build process
An improved build process

- Industrializing your build process
- Why bother?
An improved build process

- Industrializing your build process
  - Why bother?
    - Each team and each project uses different conventions
An improved build process

- Industrializing your build process
- Why bother?
  - Each team and each project uses different conventions
  - Build scripts are ad-hoc and difficult to understand and maintain
An improved build process

Industrializing your build process

Why bother?

- Each team and each project uses different conventions
- Build scripts are ad-hoc and difficult to understand and maintain
- Little reuse of components between teams and projects
An improved build process

Industrializing your build process

Why bother?

- Each team and each project uses different conventions
- Build scripts are ad-hoc and difficult to understand and maintain
- Little reuse of components between teams and projects
- Code quality metrics and reporting are not done systematically, or not at all...
An improved build process

Industrializing your build process

Why bother?

- Each team and each project uses different conventions
- Build scripts are ad-hoc and difficult to understand and maintain
- Little reuse of components between teams and projects
- Code quality metrics and reporting are not done systematically, or not at all...
- Increased learning curve and maintenance costs...
An improved build process

Industrializing your build process

Why bother?

- Each team and each project uses different conventions
- Build scripts are ad-hoc and difficult to understand and maintain
- Little reuse of components between teams and projects
- Code quality metrics and reporting are not done systematically, or not at all...
- Increased learning curve and maintenance costs...

What can you do?
Industrializing your build process

Why bother?

- Each team and each project uses different conventions
- Build scripts are ad-hoc and difficult to understand and maintain
- Little reuse of components between teams and projects
- Code quality metrics and reporting are not done systematically, or not at all...
- Increased learning curve and maintenance costs...

What can you do?

- Use Maven 2
An improved build process

Industrializing your build process

Why bother?

- Each team and each project uses different conventions
- Build scripts are ad-hoc and difficult to understand and maintain
- Little reuse of components between teams and projects
- Code quality metrics and reporting are not done systematically, or not at all...
- Increased learning curve and maintenance costs...

What can you do?

- Use Maven 2
- Training, coaching, mentoring...
An improved build process

Introducing Maven 2
An improved build process

- Introducing Maven 2
- More than just your average build scripting tool!
An improved build process

- Introducing Maven 2
- More than just your average build scripting tool!
  - A build tool
An improved build process

- Introducing Maven 2
- More than just your average build scripting tool!
  - A build tool
  - A dependency management tool
An improved build process

- Introducing Maven 2
- More than just your average build scripting tool!
  - A build tool
  - A dependency management tool
  - A documentation generation tool
An improved build process

- Introducing Maven 2
- More than just your average build scripting tool!
  - A build tool
  - A dependency management tool
  - A documentation generation tool
  - A metrics/code quality reporting tool
An improved build process

- Introducing Maven 2
  - More than just your average build scripting tool!
    - A build tool
    - A dependency management tool
    - A documentation generation tool
    - A metrics/code quality reporting tool
    - A project management tool
An improved build process

- Introducing Maven 2
- More than just your average build scripting tool!
  - A build tool
  - A dependency management tool
  - A documentation generation tool
  - A metrics/code quality reporting tool
  - A project management tool
  - And more…
An improved build process

Some key features of Maven 2
An improved build process

- Some key features of Maven 2
- Declarative dependency management
An improved build process

- Some key features of Maven 2
- Declarative dependency management
- A standard directory structure
An improved build process

- Some key features of Maven 2
  - Declarative dependency management
  - A standard directory structure
  - A standard, but extensible, build lifecycle
An improved build process

- Some key features of Maven 2
  - Declarative dependency management
  - A standard directory structure
  - A standard, but extensible, build lifecycle
  - Good technical documentation and reporting
An improved build process

The main benefits of Maven 2
An improved build process

- The main benefits of Maven 2
- Standardising the build process
An improved build process

- The main benefits of Maven 2
- Standardising the build process
- Standard directory structures and conventions
An improved build process

- The main benefits of Maven 2
- Standardising the build process
- Standard directory structures and conventions
- Convention over configuration
An improved build process

- The main benefits of Maven 2
  - Standardising the build process
  - Standard directory structures and conventions
  - Convention over configuration
  - Improved dependency management
An improved build process

Key Maven 2 Features

- Improved build process
- New features
- Better organization
An improved build process

Key Maven 2 Features

Standard Directory Structure
An improved build process

- Key Maven 2 Features
  - Standard Directory Structure
    - Familiar structure across all projects
An improved build process

Key Maven 2 Features

- Standard Directory Structure
  - Familiar structure across all projects
  - Convention Over Configuration
An improved build process

Key Maven 2 Features

- Standard Directory Structure
  - Familiar structure across all projects
  - Convention Over Configuration
  - Less low-level scripting
An improved build process

- Key Maven 2 Features
- Standard Life Cycle
An improved build process

- Key Maven 2 Features
  - Standard Life Cycle
    - Familiar commands across all projects
An improved build process

- Key Maven 2 Features
  - Standard Life Cycle
    - Familiar commands across all projects
    - No re-inventing the wheel
An improved build process

Key Maven 2 Features

- Standard Life Cycle
  - Familiar commands across all projects
  - No re-inventing the wheel

```
process-resources
compile
test-compile
test
package
integration-test
install
deploy
```
An improved build process

Key Maven 2 Features

- Standard Life Cycle
  - Familiar commands across all projects
  - No re-inventing the wheel

$mvn compile
An improved build process

- Key Maven 2 Features
  - Standard Life Cycle
    - Familiar commands across all projects
    - No re-inventing the wheel

```
$mvn compile
$mvn test
```

```
process-resources  compile
  test-compile  test
  package  integration-test
  install  deploy
```
Key Maven 2 Features

- Standard Life Cycle
  - Familiar commands across all projects
  - No re-inventing the wheel

```
$mvn compile
$mvn test
$mvn package
```
An improved build process

Key Maven 2 Features

- Standard Life Cycle
  - Familiar commands across all projects
  - No re-inventing the wheel

```
$mvn compile
$mvn test
$mvn package
$mvn integration-test
```
An improved build process

Key Maven 2 Features

- Standard Life Cycle
  - Familiar commands across all projects
  - No re-inventing the wheel

$mvn compile
$mvn test
$mvn package
$mvn integration-test
$mvn install
$mvn deploy
An improved build process

Key Maven 2 Features

- Standard Life Cycle
  - Familiar commands across all projects
  - No re-inventing the wheel

```
$mvn compile
$mvn test
$mvn package
$mvn integration-test
$mvn install
$mvn deploy
```
An improved build process

- Key Maven 2 Features
- Maven artifacts
Key Maven 2 Features

- Maven artifacts
  - Each Maven artifact has a unique identifier, or “co-ordinates”
An improved build process

- Key Maven 2 Features
  - Maven artifacts
  - Each Maven artifact has a unique identifier, or “co-ordinates”

```xml
<project>
  ...
  <groupId>com.mycompany.accounting</groupId>
  <artifactId>accounting-core</artifactId>
  <packaging>jar</packaging>
  <version>1.1</version>
  <name>Accounting Core package</name>
  ...
</project>
```
Key Maven 2 Features

Maven artifacts

Each Maven artifact has a unique identifier, or “co-ordinates”
An improved build process

Key Maven 2 Features

- Maven artifacts
  - Each Maven artifact has a unique identifier, or “co-ordinates”

```xml
<project...>
  ...
  <groupId>com.mycompany.accounting</groupId>
  <artifactId>accounting-core</artifactId>
  <packaging>jar</packaging>
  <version>1.1</version>
  <name>Accounting Core package</name>
  ...
```
An improved build process

- Key Maven 2 Features
- Maven artifacts
  - Each Maven artifact has a unique identifier, or “co-ordinates”
An improved build process

Key Maven 2 Features

- Maven artifacts
  - Each Maven artifact has a unique identifier, or “co-ordinates”

```
<project...>
  ...
  <groupId>com.mycompany.accounting</groupId>
  <artifactId>accounting-core</artifactId>
  <packaging>jar</packaging>
  <version>1.1</version>
  <name>Accounting Core package</name>
  ...
```
An improved build process

Key Maven 2 Features

Declarative Dependency Management
An improved build process

- Key Maven 2 Features
- Declarative Dependency Management
An improved build process

Key Maven 2 Features

Declarative Dependency Management
An improved build process

- Key Maven 2 Features
- Declarative Dependency Management

Your application uses third-party libraries

These libraries need libraries of their own
Key Maven 2 Features

- Declarative Dependency Management

Your application uses third-party libraries

These libraries need libraries of their own

And so on...
An improved build process

- Key Maven 2 Features
- Declarative Dependency Management
An improved build process

Key Maven 2 Features

- Declarative Dependency Management
  - The traditional approach - JAR files in your project
An improved build process

Key Maven 2 Features

- Declarative Dependency Management
  - The traditional approach - JAR files in your project
  - Each project has its own set of JAR files
An improved build process

- Key Maven 2 Features

  - Declarative Dependency Management
    - The traditional approach - JAR files in your project
      - Each project has its own set of JAR files
      - Unnecessary duplication
An improved build process

Key Maven 2 Features

Declarative Dependency Management

- The traditional approach - JAR files in your project
  - Each project has its own set of JAR files
  - Unnecessary duplication
  - Hard to keep track of versions
An improved build process

- Key Maven 2 Features
  
  - Declarative Dependency Management
    
    - The traditional approach - JAR files in your project
      
      - Each project has its own set of JAR files
      
      - Unnecessary duplication
      
      - Hard to keep track of versions
      
      - Errors due to incompatible JAR files
Key Maven 2 Features

Declarative Dependency Management

The traditional approach - JAR files in your project

Each project has its own set of JAR files

Unnecessary duplication

Hard to keep track of versions

Errors due to incompatible JAR files

Overloads the source code repository
An improved build process

Key Maven 2 Features

Declarative Dependency Management
An improved build process

- Key Maven 2 Features
- Declarative Dependency Management
  - The Declarative approach - “just tell me what you need”
An improved build process

Key Maven 2 Features

- Declarative Dependency Management
  - The Declarative approach - “just tell me what you need”
  - “Versionned” JAR files are stored on a central server
An improved build process

Key Maven 2 Features

Declerative Dependency Management

- The Declarative approach - "just tell me what you need"
- "Versionned" JAR files are stored on a central server
- Each project "declares" what libraries it needs
An improved build process

Key Maven 2 Features

- **Declarative Dependency Management**
  - The Declarative approach - “just tell me what you need”
  - “Versionned” JAR files are stored on a central server
  - Each project “declares” what libraries it needs
  - Secondary dependencies are automatically downloaded
Key Maven 2 Features

Declarative Dependency Management
- The Declarative approach - “just tell me what you need”
- “Versionned” JAR files are stored on a central server
- Each project “declares” what libraries it needs
- Secondary dependencies are automatically downloaded
- Works well for both internal and external dependencies
An improved build process

Key Maven 2 Features

Declaring Dependencies in Maven

```xml
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.4</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>2.4</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>2.5.6</version>
  </dependency>
  ...
</dependencies>
```
An improved build process

Key Maven 2 Features

Declaring Dependencies in Maven

- Declared in the build script itself

```xml
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.4</version>
    <scope:test</scope>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>2.4</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>2.5.6</version>
  </dependency>
  ...
</dependencies>
```
An improved build process

- Key Maven 2 Features
  - Declaring Dependencies in Maven
    - Declared in the build script itself
    - Dependencies with version numbers

```xml
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.4</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>2.4</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>2.5.6</version>
  </dependency>
  ...
</dependencies>
```
An improved build process

Key Maven 2 Features

- Declaring Dependencies in Maven
  - Declared in the build script itself
  - Dependencies with version numbers
  - Different types of dependencies

```xml
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.4</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>servlet-api</artifactId>
        <version>2.4</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>2.5.6</version>
    </dependency>
    ...
</dependencies>
```
An improved build process

Key Maven 2 Features

- Declaring Dependencies in Maven
  - Declared in the build script itself
  - Dependencies with version numbers
  - Different types of dependencies
  - Compile, test, provided,...

```xml
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.4</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.4</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>2.5.6</version>
  </dependency>
  ...
</dependencies>
```
An improved build process

Key Maven 2 Features

- Declaring Dependencies in Maven
  - Declared in the build script itself
  - Dependencies with version numbers
  - Different types of dependencies
  - Compile, test, provided,...

```xml
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.4</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.4</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>2.5.6</version>
  </dependency>
  ...
</dependencies>
```
An improved build process

Key Maven 2 Features

Declaring Dependencies in Maven

- Declared in the build script itself
- Dependencies with version numbers
- Different types of dependencies
- Compile, test, provided, ...

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.4</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.4</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>2.5.6</version>
  </dependency>
  ...
</dependencies>
```
An improved build process

- Maven 2 support in Eclipse
An improved build process

Maven 2 support in Eclipse

The **m2eclipse** provides excellent tool support for Maven, including:
Maven 2 support in Eclipse

The **m2eclipse** provides excellent tool support for Maven, including:

- Viewing the Maven dependencies

```markdown
- tax-calculator-core
  - src/main/java
  - src/main/resources
  - src/test/java
  - src/test/resources
- JRE System Library [J2SE-1.5]
- Maven Dependencies
  - junit-4.4.jar – /Users/johnsmart/m2/repository/junit/junit/4.4
  - spring-core-2.5.6.jar – /Users/johnsmart/m2/repository/org/springframework/spring-core/2.5.6
  - commons-loggig-1.1.1.jar – /Users/johnsmart/m2/repository/commons-logging/commons-logging/1.1.1
  - spring-beans-2.5.6.jar – /Users/johnsmart/m2/repository/org/springframework/spring-beans/2.5.6
  - joda-time-1.5.2.jar – /Users/johnsmart/m2/repository/joda-time/joda-time/1.5.2
  - log4j-1.2.12.jar – /Users/johnsmart/m2/repository/log4j/log4j/1.2.12
  - hamcrest-all-1.1.jar – /Users/johnsmart/m2/repository/org/hamcrest/hamcrest-all/1.1
  - servlet-api-2.4.jar – /Users/johnsmart/m2/repository/javax/servlet/servlet-api/2.4
- src
- target
- pom.xml
```
An improved build process

- Maven 2 support in Eclipse

- The **m2eclipse** provides excellent tool support for Maven, including:

  - Viewing the Maven dependencies

```
- tax-calculator-core
  - src/main/java
  - src/main/resources
  - src/test/java
  - src/test/resources
  - JRE System Library [J2SE-1.5]
- Maven Dependencies
  - junit-4.4.jar - /Users/johnsmart/m2/repository/junit/junit/4.4
  - spring-core-2.5.6.jar - /Users/johnsmart/m2/repository/org/springframework/spring-core/2.5.6
  - commons-logging-1.1.1.jar - /Users/johnsmart/m2/repository/commons-logging/commons-logging/1.1.1
  - spring-beans-2.5.6.jar - /Users/johnsmart/m2/repository/org/springframework/springframework/spring-beans/2.5.6
  - joda-time-1.5.2.jar - /Users/johnsmart/m2/repository/joda-time/joda-time/1.5.2
  - log4j-1.2.12.jar - /Users/johnsmart/m2/repository/log4j/log4j/1.2.12
  - hamcrest-all-1.1.jar - /Users/johnsmart/m2/repository/org/hamcrest/hamcrest-all/1.1
  - servlet-api-2.4.jar - /Users/johnsmart/m2/repository/javax/servlet/servlet-api/2.4

Project dependencies
```
An improved build process

- Maven 2 support in Eclipse
  - The **m2eclipse** provides excellent tool support for Maven, including:
  - Viewing the Maven dependencies
An improved build process

- Maven 2 support in Eclipse
  - The **m2eclipse** provides excellent tool support for Maven, including:
    - Adding new dependencies to your project
Maven 2 support in Eclipse

The **m2eclipse** provides excellent tool support for Maven, including:

- Adding new dependencies to your project.
An improved build process

- Maven 2 support in Eclipse
  - The **m2eclipse** provides excellent tool support for Maven, including:
    - Adding new dependencies to your project
    - Dependencies already present in your project

![Image showing the m2eclipse interface with dependency search results and scope options.]
An improved build process

Maven 2 support in Eclipse

The **m2eclipse** provides excellent tool support for Maven, including:

- Adding new dependencies to your project
- Dependencies already present in your project
- Dependency scopes
Maven 2 support in Eclipse

The **m2eclipse** provides excellent tool support for Maven, including:

- Adding new dependencies to your project
- Dependencies already present in your project
- Available versions
- Dependency scopes
Maven 2 support in Eclipse

The **m2eclipse** provides excellent tool support for Maven, including:

- Viewing the dependency graph
An improved build process

- Maven 2 support in Eclipse
  - The **m2eclipse** provides excellent tool support for Maven, including:
  - Viewing the dependency graph
An improved build process

- Maven 2 support in Eclipse
  - The **m2eclipse** provides excellent tool support for Maven, including:
    - Resolving dependency conflicts
An improved build process

Maven 2 support in Eclipse

The **m2eclipse** provides excellent tool support for Maven, including:

- Resolving dependency conflicts
An improved build process

Maven 2 support in Eclipse

- The **m2eclipse** provides excellent tool support for Maven, including:
  - Resolving dependency conflicts
An improved build process

- Maven 2 support in Eclipse
- The **m2eclipse** provides excellent tool support for Maven, including:
  - Resolving dependency conflicts
A walk through a simple Maven 2 project
Better release management

Organizing your internal releases
Better release management

- Organizing your internal releases
- Internal releases are hard to co-ordinate
Better release management

- Organizing your internal releases
- Internal releases are hard to co-ordinate
  - How do I share my API with other teams?
Better release management

- Organizing your internal releases
  - Internal releases are hard to co-ordinate
    - How do I share my API with other teams?
    - Where is the latest version of that API?
Better release management

Organizing your internal releases

- Internal releases are hard to co-ordinate
  - How do I share my API with other teams?
  - Where is the latest version of that API?
  - What version am I using, anyway?
Better release management

Organizing your internal releases

- Internal releases are hard to co-ordinate
  - How do I share my API with other teams?
  - Where is the latest version of that API?
  - What version am I using, anyway?

- What can you do?
Better release management

Organizing your internal releases

- Internal releases are hard to co-ordinate
  - How do I share my API with other teams?
  - Where is the latest version of that API?
  - What version am I using, anyway?

- What can you do?
  - Install an Enterprise Maven Repository
Better release management

- Organizing your internal releases
- Maven Repositories
Better release management

- Organizing your internal releases
- Maven Repositories
  - Public web sites containing JARs for many open source projects
Better release management

- Organizing your internal releases
- Maven Repositories
  - Public web sites containing JARs for many open source projects
  - Maven automatically downloads the JARs you need onto your local machine
Better release management

- Organizing your internal releases
  - Maven Repositories
    - Public web sites containing JARs for many open source projects
    - Maven automatically downloads the JARs you need onto your local machine
    - Publish internal APIs on your own repository
Better release management

- Organizing your internal releases
- Maven Repositories
- JAR files are downloaded to a local cache
Better release management

- Organizing your internal releases
- Maven Repositories
  - There are several public Maven repositories
Better release management

- Organizing your internal releases
- Maven Repositories
  - You can also set up your own Enterprise Maven Repository
Better release management

- Organizing your internal releases
- Maven Repositories
  - The Enterprise Maven Repository can also act as a proxy/cache to the public repositories
Better release management

- Organizing your internal releases
- Key Best Practices
Better release management

- Organizing your internal releases
- Key Best Practices
  - Deploy your internal releases to a local Enterprise repository
Better release management

Organizing your internal releases

Key Best Practices

- Deploy your internal releases to a local Enterprise repository
- Store proprietary JAR files you need here as well
Organizing your internal releases

Key Best Practices

- Deploy your internal releases to a local Enterprise repository
- Store proprietary JAR files you need here as well
- Distinguish between snapshot and release versions
Better release management

- Organizing your internal releases
- Using the Nexus repository manager
Better release management

- Organizing your internal releases
- Using the Nexus repository manager
- Easy to set up
Better release management

- Organizing your internal releases
- Using the Nexus repository manager
  - Easy to set up
  - Easy to administer
Better release management

- Organizing your internal releases
- Using the Nexus repository manager
  - Easy to set up
  - Easy to administer
  - Proxy/cache

www.devoxx.com
Better release management

- Snapshots and Releases
Better release management

- Snapshots and Releases
- Snapshots
Better release management

- Snapshots and Releases
  - Snapshots
    - Work in progress
Better release management

Snapshots and Releases

Snapshots

- Work in progress

- A new time-stamped version deployed with each deployment
Better release management

Snapshots and Releases

Snapshots

- Work in progress

- A new time-stamped version deployed with each deployment

```xml
<artifactId>tax-calculator-core</artifactId>
<packaging>jar</packaging>
<version>1.0.0-SNAPSHOT</version>
```
Better release management

- Snapshots and Releases
  - Snapshots
    - Work in progress
    - A new time-stamped version deployed with each deployment

```xml
<artifactId>tax-calculator-core</artifactId>
<packaging>jar</packaging>
<version>1.0.0-SNAPSHOT</version>
```
Better release management

Snapshots and Releases

Snapshots

- Work in progress
- A new time-stamped version deployed with each deployment

```xml
<artifactId>tax-calculator-core</artifactId>
<packaging>jar</packaging>
<version>1.0.0-SNAPSHOT</version>
```

`$mvn deploy`
Better release management

- Snapshots and Releases
  - Snapshots
    - Work in progress
  - A new time-stamped version deployed with each deployment

```
<artifactId>tax-calculator-core</artifactId>
<packaging>jar</packaging>
<version>1.0.0-SNAPSHOT</version>
```

$mvn deploy
Better release management

- Snapshots and Releases
  - Snapshots
    - Work in progress
    - A new time-stamped version deployed with each deployment

```
<artifactId>tax-calculator-core</artifactId>
<packaging>jar</packaging>
<version>1.0.0-SNAPSHOT</version>
```

```
$mvn deploy
```
Better release managment

Snapshots and Releases
Better release management

- Snapshots and Releases
- Releases
Better release management

- Snapshots and Releases
  - Releases
    - Stable, tested release
Better release management

- Snapshots and Releases
- Releases
  - Stable, tested release
  - The deployed artifact is unique
Better release management

- Snapshots and Releases
- Releases
  - Stable, tested release
  - The deployed artifact is unique

```xml
<artifactId>tax-calculator-core</artifactId>
<packaging>jar</packaging>
<version>1.0.0</version>
```
Better release management

- Snapshots and Releases
- Releases
  - Stable, tested release
  - The deployed artifact is unique

```
<artifactId>tax-calculator-core</artifactId>
<packaging>jar</packaging>
<version>1.0.0</version>
```

```
$mvn deploy
```
Better release management

- Snapshots and Releases
  - Releases
    - Stable, tested release
    - The deployed artifact is unique

```xml
<artifactId>tax-calculator-core</artifactId>
<packaging>jar</packaging>
<version>1.0.0</version>
```

```
$mvn deploy
```

tax-calculator-core-1.0.0-20081123.093826-1.jar
Better release management

- Snapshots and Releases
Better release management

- Snapshots and Releases
- Setting it all up in Maven
Better release management

- Snapshots and Releases
  - Setting it all up in Maven
    - To deploy a snapshot version, you need:
Better release management

Snapshots and Releases

Setting it all up in Maven

To deploy a snapshot version, you need:

- A Snapshot repository
Better release management

Snapshots and Releases

Setting it all up in Maven

To deploy a snapshot version, you need:

- A Snapshot repository
- A SNAPSHOT version number

\[<version>1.0.1-SNAPSHOT</version>\]
Better release managment

Snapshots and Releases

Setting it all up in Maven

To deploy a snapshot version, you need:

- A Snapshot repository
- A SNAPSHOT version number
- User authentication for the repository

```xml
<settings>
   <servers>
       <server>
           <id>company-repo</id>
           <username>john</username>
           <password>secret</password>
       </server>
   </servers>
</settings>
```
Snapshots and Releases

Setting it all up in Maven

To deploy a snapshot version, you need:

- A Snapshot repository
- A SNAPSHOT version number
- User authentication for the repository
- A Snapshot configuration for the repository
Better release management

Snapshots and Releases
Better release management

- Snapshots and Releases
- Automating snapshot deployments
Better release management

- Snapshots and Releases
- Automating snapshot deployments
  - Use the `maven-release-plugin` to automate SCM book-keeping
Better release management

- Snapshots and Releases
- Automating snapshot deployments
- Use the **maven-release-plugin** to automate SCM book-keeping
Better release management

- Snapshots and Releases
- Automating snapshot deployments
  - Use the **maven-release-plugin** to automate SCM book-keeping
    - mvn:prepare

```
<version>1.0.1-SNAPSHOT</version>
```
Better release management

- Snapshots and Releases
- Automating snapshot deployments
  - Use the **maven-release-plugin** to automate SCM book-keeping
  - mvn:prepare

```xml
<pom.xml>
  <version>1.0.1-SNAPSHOT</version>
  <version>1.0.1</version>
</pom.xml>
```
Better release management

Snapshots and Releases

Automating snapshot deployments

- Use the **maven-release-plugin** to automate SCM book-keeping
  - `mvn:prepare`
Better release management

- Snapshots and Releases

- Automating snapshot deployments

  - Use the `maven-release-plugin` to automate SCM book-keeping

  - `mvn:prepare`
Snapshots and Releases

Automating snapshot deployments

Use the **maven-release-plugin** to automate SCM book-keeping

- `mvn:prepare`
- `mvn:perform`

```xml
<version>1.0.1-SNAPSHOT</version>
<version>1.0.1</version>
<version>1.0.2-SNAPSHOT</version>
```
Better release management

- Snapshots and Releases
  - Automating snapshot deployments
    - Use the `maven-release-plugin` to automate SCM book-keeping
      - `mvn:prepare`
      - `mvn:perform`

```
<version>1.0.1-SNAPSHOT</version>
<version>1.0.1</version>
<version>1.0.2-SNAPSHOT</version>
```
Better release management

- Snapshots and Releases
  - Automating snapshot deployments
    - Use the **maven-release-plugin** to automate SCM book-keeping
      - `mvn:prepare`
      - `mvn:perform`
      - `mvn:rollback`

```
<version>1.0.1-SNAPSHOT</version>
<version>1.0.1</version>
<version>1.0.2-SNAPSHOT</version>
```

![Diagram of Nexus with version tags](image)

```
$mvn deploy
tax-calculator-core-1.0.1
```
Releases, Snapshots and Repositories
Automating the build process

Continuous Integration - what’s the issue?
Automating the build process

- Continuous Integration - what’s the issue?
- Traditional development cycles are bad for your health:
Automating the build process

- Continuous Integration - what’s the issue?
  - Traditional development cycles are bad for your health:
    - Integration is long and difficult
Automating the build process

- Continuous Integration - what’s the issue?
- Traditional development cycles are bad for your health:
  - Integration is long and difficult
  - Poor visibility on development progress
Automating the build process

Continuous Integration - what’s the issue?

Traditional development cycles are bad for your health:

- Integration is long and difficult
- Poor visibility on development progress
- Functional tests are done too late
Automating the build process

Continuous Integration - what’s the issue?

Traditional development cycles are bad for your health:

- Integration is long and difficult
- Poor visibility on development progress
- Functional tests are done too late
- Raised issues are harder to fix
Continuous Integration - what’s the issue?

Traditional development cycles are bad for your health:

- Integration is long and difficult
- Poor visibility on development progress
- Functional tests are done too late
- Raised issues are harder to fix
- The client gets a sub-optimal product
Automating the build process

Continuous Integration - what’s involved?
Automating the build process

- Continuous Integration - why bother?
Automating the build process

Continuous Integration - why bother?

- Smoother integration process
Automating the build process

Continuous Integration - why bother?

- Smoother integration process
- Automatic regression testing
Automating the build process

Continuous Integration - why bother?

- Smoother integration process
- Automatic regression testing
- Regular working releases
Automating the build process

Continuous Integration - why bother?

- Smoother integration process
- Automatic regression testing
- Regular working releases
- Earlier functional testing
Automating the build process

Continuous Integration - why bother?

- Smoother integration process
- Automatic regression testing
- Regular working releases
- Earlier functional testing
- Faster and easier bug fixes
Continuous Integration - why bother?

- Smoother integration process
- Automatic regression testing
- Regular working releases
- Earlier functional testing
- Faster and easier bug fixes
- Better visibility
Continuous Integration - why bother?

- Smoother integration process
- Automatic regression testing
- Regular working releases
- Earlier functional testing
- Faster and easier bug fixes
- Better visibility
Automating the build process

- Continuous Integration - what you need
Automating the build process

Continuous Integration - what you need

Automated build process (e.g. Maven)
Automating the build process

Continuous Integration - what you need

- Automated build process (e.g. Maven)
- Automated tests (JUnit, Selenium, easyb...)

www.devoxx.com
Automating the build process

Continuous Integration - what you need

- Automated build process (e.g. Maven)
- Automated tests (JUnit, Selenium, easyb...)
- Source code repository
Automating the build process

Continuous Integration - what you need

- Automated build process (e.g. Maven)
- Automated tests (JUnit, Selenium, easyb...)
- Source code repository
- Continuous Build Server
Automating the build process

- Continuous Integration - what can it do
Automating the build process

- Continuous Integration - what can it do
- More than just your average build scheduler!
Continuous Integration - what can it do

More than just your average build scheduler!

Raise (and monitor) integration issues - fast!
Automating the build process

- Continuous Integration - what can it do
  - More than just your average build scheduler!
    - Raise (and monitor) integration issues - fast!
    - Automatically publish Maven artifacts
Automating the build process

Continuous Integration - what can it do

- More than just your average build scheduler!
  - Raise (and monitor) integration issues - fast!
  - Automatically publish Maven artifacts
  - Monitor your build process
Automating the build process

Continuous Integration - what can it do

- More than just your average build scheduler!
  - Raise (and monitor) integration issues - fast!
  - Automatically publish Maven artifacts
  - Monitor your build process
  - Monitor and report on code quality and code coverage
Automating the build process

- Continuous Integration - raising issues fast!
Automating the build process

- Continuous Integration - raising issues fast!
- Use an appropriate notification strategy, e.g.
Automating the build process

- Continuous Integration - raising issues fast!
- Use an appropriate notification strategy, e.g.
  - The committer is notified for all build results
Continuous Integration - raising issues fast!

- Use an appropriate notification strategy, e.g.
  - The committer is notified for *all* build results
  - Team members are notified for any failed builds
Automating the build process

Continuous Integration - raising issues fast!

Use an appropriate notification strategy, e.g.

- The committer is notified for *all* build results
- Team members are notified for any failed builds
- Team leader gets special notification after 5 successive build failures
Continuous Integration - raising issues fast!
Automating the build process

- Continuous Integration - raising issues fast!
- Use an appropriate (fast) notification mechanism
Automating the build process

- Continuous Integration - raising issues fast!
- Use an appropriate (fast) notification mechanism
  - Instant Messaging, SMS etc. for fast notification
Automating the build process

- Continuous Integration - raising issues fast!
  - Use an appropriate (fast) notification mechanism
    - Instant Messaging, SMS etc. for fast notification
    - Email as a secondary channel
Automating the build process

Continuous Integration - raising issues fast!

Use an appropriate (fast) notification mechanism

- Instant Messaging, SMS etc. for fast notification
- Email as a secondary channel
- RSS for consulting build history
Automating the build process

- Continuous Integration - raising issues fast!
Automating the build process

- Continuous Integration - raising issues fast!
- Use a good build plan strategy
Automating the build process

- Continuous Integration - raising issues fast!
  - Use a good build plan strategy
  - Fast builds first, e.g.
Automating the build process

- Continuous Integration - raising issues fast!
  - Use a good build plan strategy
    - Fast builds first, e.g.
      - Unit tests before integration tests
Continuous Integration - raising issues fast!

Use a good build plan strategy

- Fast builds first, e.g.
  - Unit tests before integration tests
  - Integration tests before metrics
Automating the build process

Continuous Integration - raising issues fast!

- Use a good build plan strategy
  - Fast builds first, e.g.
    - Unit tests before integration tests
    - Integration tests before metrics
  - Use manual builds where appropriate
Continuous Integration - raising issues fast!

Use a good build plan strategy

- Fast builds first, e.g.
  - Unit tests before integration tests
  - Integration tests before metrics

Use manual builds where appropriate

Release builds
Continuous Integration - raising issues fast!

Use a good build plan strategy

- Fast builds first, e.g.
  - Unit tests before integration tests
  - Integration tests before metrics

Use manual builds where appropriate

- Release builds
- Deployments
Continuous Integration - raising issues fast!

Use a good build plan strategy

- Fast builds first, e.g.
  - Unit tests before integration tests
  - Integration tests before metrics

Use manual builds where appropriate

- Release builds
- Deployments
- ...

Automating the build process
Continuous Integration - raising issues fast!

Use a good build plan strategy
Automating the build process

- Continuous Integration - raising issues fast!
- Use a good build plan strategy
  - Organize your build dependencies
Automating the build process

- Continuous Integration - raising issues fast!
  - Use a good build plan strategy
    - Organize your build dependencies
    - Projects should be rebuilt
Automating the build process

- Continuous Integration - raising issues fast!
  - Use a good build plan strategy
    - Organize your build dependencies
      - Projects should be rebuilt
        - If the project source code changes
Automating the build process

- Continuous Integration - raising issues fast!
  - Use a good build plan strategy
    - Organize your build dependencies
      - Projects should be rebuilt
        - If the project source code changes
        - If a snapshot dependency is modified
Automating the build process

- Continuous Integration - raising issues fast!
  - Use a good build plan strategy
    - Organize your build dependencies
      - Projects should be rebuilt
      - If the project source code changes
      - If a snapshot dependency is modified
    - Projects should *not* be rebuilt
Continuous Integration - raising issues fast!

- Use a good build plan strategy
  - Organize your build dependencies
    - Projects should be rebuilt
    - If the project source code changes
    - If a snapshot dependency is modified
  - Projects should **not** be rebuilt
    - If an unrelated project is rebuilt
Automating the build process

- Continuous Integration - raising issues fast!
  - Use a good build plan strategy
    - Organize your build dependencies
      - Projects should be rebuilt
      - If the project source code changes
      - If a snapshot dependency is modified
    - Projects should not be rebuilt
      - If an unrelated project is rebuilt
      - If an unrelated version of a dependency is rebuilt
Continuous Integration - raising issues fast!

Use a good build plan strategy

- Organize your build dependencies
  - Projects should be rebuilt
  - If the project source code changes
  - If a snapshot dependency is modified
  - Projects should **not** be rebuilt
    - If an unrelated project is rebuilt
    - If an unrelated version of a dependency is rebuilt
  - Note: Hudson does this automatically for Maven build jobs
Automating the build process

- Continuous Integration - raising issues fast!
- Use a good build plan strategy
- Organize your build dependencies
Automating the build process

- Continuous Integration - raising issues fast!
- Use a good build plan strategy
- Organize your build dependencies
- Case 1: Updating a snapshot dependency
Automating the build process

- Continuous Integration - raising issues fast!
- Use a good build plan strategy
- Organize your build dependencies
- Case 1: Updating a snapshot dependency

```
- Modifications made here
- tax-calculator-core: 1.0.1-SNAPSHOT
- tax-calculator-utils: 1.0.4-SNAPSHOT
- tax-calculator-core: 1.0.1-SNAPSHOT
- tax-calculator-utils: 1.0.3
- tax-calculator-webapp
```
Automating the build process

- Continuous Integration - raising issues fast!
- Use a good build plan strategy
- Organize your build dependencies
- Case 1: Updating a snapshot dependency

```
Modifications made here

Matching dependency

tax-calculator-core: 1.0.1-SNAPSHOT

tax-calculator-utils: 1.0.3

tax-calculator-webapp
```
Automating the build process

- Continuous Integration - raising issues fast!
- Use a good build plan strategy
- Organize your build dependencies
- Case 1: Updating a snapshot dependency

Matched dependency:
- Modifications made here
- Should be rebuilt
Automating the build process

- Continuous Integration - raising issues fast!
- Use a good build plan strategy
- Organize your build dependencies
Automating the build process

- Continuous Integration - raising issues fast!
- Use a good build plan strategy
- Organize your build dependencies
- Case 2: Updating a different version of a dependency

```
tax-calculator-core: 1.0.1-SNAPSHOT
```
```
tax-calculator-utils: 1.0.4-SNAPSHOT
```
```
tax-calculator-utils: 1.0.3
```
```
tax-calculator-webapp
```
Automating the build process

- Continuous Integration - raising issues fast!
- Use a good build plan strategy
- Organize your build dependencies
- Case 2: Updating a different version of a dependency

Modifications made here

- tax-calculator-core: 1.0.1-SNAPSHOT
- tax-calculator-utils: 1.0.3
- tax-calculator-webapp

- tax-calculator-core 1.0.1-SNAPSHOT
- tax-calculator-utils 1.0.4-SNAPSHOT
Automating the build process

- Continuous Integration - raising issues fast!
- Use a good build plan strategy
- Organize your build dependencies
- Case 2: Updating a different version of a dependency

![Diagram of dependency versions]
Continuous Integration - raising issues fast!

Use a good build plan strategy

Organize your build dependencies

Case 2: Updating a different version of a dependency
Automating the builds
Better Testing

- Improving your testing game
Better Testing

- Improving your testing game
  - Innovative testing techniques
Better Testing

- Improving your testing game
  - Innovative testing techniques
  - Automating unit tests
Better Testing

Improving your testing game

- Innovative testing techniques
- Automating unit tests
- Separate unit tests and integration tests
Better Testing

Improving your testing game

- Innovative testing techniques
- Automating unit tests
- Separate unit tests and integration tests
- Monitor test duration
Better Testing

Improving your testing game

- Innovative testing techniques
- Automating unit tests
- Separate unit tests and integration tests
- Monitor test duration
- Keep tabs on test coverage
Better Testing

- Use innovative testing techniques
Better Testing

- Use innovative testing techniques
- Unit testing
Better Testing

- Use innovative testing techniques
  - Unit testing
    - A cornerstone of modern software development
Better Testing

- Use innovative testing techniques
  - Unit testing
    - A cornerstone of modern software development
    - Unit tests can help you:
Better Testing

- Use innovative testing techniques
  - Unit testing
    - A cornerstone of modern software development
    - Unit tests can help you:
      - Ensure that code behaves as expected
Better Testing

Use innovative testing techniques

*Unit testing*

- A cornerstone of modern software development
- Unit tests can help you:
  - Ensure that code behaves as expected
  - Make your code more flexible and easier to maintain
Better Testing

Use innovative testing techniques

Unit testing

- A cornerstone of modern software development
- Unit tests can help you:
  - Ensure that code behaves as expected
  - Make your code more flexible and easier to maintain
  - Detect regressions
Better Testing

Use innovative testing techniques

Unit testing

- A cornerstone of modern software development
- Unit tests can help you:
  - Ensure that code behaves as expected
  - Make your code more flexible and easier to maintain
  - Detect regressions
  - Document your code
Better Testing

- Use innovative testing techniques
- *Innovative* unit testing
Better Testing

- Use innovative testing techniques
  - **Innovative** unit testing
    - Parameterized tests
Better Testing

- Use innovative testing techniques
  - **Innovative** unit testing
    - Parameterized tests
    - Hamcrest asserts
Better Testing

- Use innovative testing techniques
  - *Innovative* unit testing
    - Parameterized tests
    - Hamcrest asserts
    - Groovy tests
Better Testing

- Use innovative testing techniques
  - **Innovative** unit testing
    - Parameterized tests
    - Hamcrest asserts
    - Groovy tests
    - Groovy BDD testing techniques with Easyb
Better Testing

- Use innovative testing techniques
  - **Innovative** unit testing
    - Parameterized tests
    - Hamcrest asserts
    - Groovy tests
    - Groovy BDD testing techniques with Easyb
    - Use your imagination!
Better Testing

- Use innovative testing techniques
- Parameterized tests
Better Testing

- Use innovative testing techniques
  - Parameterized tests
    - Run several sets of test data against the same test case
Better Testing

- Use innovative testing techniques
  - Parameterized tests
    - Run several sets of test data against the same test case
    - Help reduce the number of unit tests to write
Better Testing

- Use innovative testing techniques
  - Parameterized tests
    - Run several sets of test data against the same test case
    - Help reduce the number of unit tests to write
    - Encourage developers to test more thoroughly
Better Testing
Better Testing

- Use innovative testing techniques
  - Parameterized tests
    - Example: Calculating income tax
Better Testing

- Use innovative testing techniques
- Parameterized tests
- Example: Calculating income tax

<table>
<thead>
<tr>
<th>Taxable income</th>
<th>PAYE rate for every $1 of taxable income (excluding ACC earners’ levy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>up to $14,000</td>
<td>12.5 cents</td>
</tr>
<tr>
<td>$14,001 to $40,000</td>
<td>21 cents</td>
</tr>
<tr>
<td>$40,001 to $70,000</td>
<td>33 cents</td>
</tr>
<tr>
<td>$70,000 and over</td>
<td>39 cents</td>
</tr>
</tbody>
</table>
Better Testing

- Use innovative testing techniques
- Parameterized tests
Better Testing

- Use innovative testing techniques
  - Parameterized tests
    - What you need:
Better Testing

- Use innovative testing techniques
  - Parameterized tests
    - What you need:
      - Some test data

<table>
<thead>
<tr>
<th>Income</th>
<th>Year</th>
<th>Expected Tax</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.00</td>
<td>2007</td>
<td>$0.00</td>
</tr>
<tr>
<td>$10,000.00</td>
<td>2007</td>
<td>$1,950.00</td>
</tr>
<tr>
<td>$20,000.00</td>
<td>2007</td>
<td>$3,900.00</td>
</tr>
<tr>
<td>$38,000.00</td>
<td>2007</td>
<td>$7,410.00</td>
</tr>
<tr>
<td>$38,001.00</td>
<td>2007</td>
<td>$7,410.33</td>
</tr>
<tr>
<td>$40,000.00</td>
<td>2007</td>
<td>$8,070.00</td>
</tr>
<tr>
<td>$60,000.00</td>
<td>2007</td>
<td>$14,670.00</td>
</tr>
<tr>
<td>$100,000.00</td>
<td>2007</td>
<td>$30,270.00</td>
</tr>
</tbody>
</table>
Better Testing

- Use innovative testing techniques
- Parameterized tests
  - What you need:
    - Some test data
    - A test class with matching fields

<table>
<thead>
<tr>
<th>Income</th>
<th>Year</th>
<th>Expected Tax</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.00</td>
<td>2007</td>
<td>$0.00</td>
</tr>
<tr>
<td>$10,000.00</td>
<td>2007</td>
<td>$1,950.00</td>
</tr>
<tr>
<td>$20,000.00</td>
<td>2007</td>
<td>$3,900.00</td>
</tr>
<tr>
<td>$38,000.00</td>
<td>2007</td>
<td>$7,410.00</td>
</tr>
<tr>
<td>$38,001.00</td>
<td>2007</td>
<td>$7,410.33</td>
</tr>
<tr>
<td>$40,000.00</td>
<td>2007</td>
<td>$8,070.00</td>
</tr>
<tr>
<td>$60,000.00</td>
<td>2007</td>
<td>$14,670.00</td>
</tr>
<tr>
<td>$100,000.00</td>
<td>2007</td>
<td>$30,270.00</td>
</tr>
</tbody>
</table>

```java
TaxCalculationTest
- income : double
- year : int
- expectedTax : double
+ TaxCalculationTest(year : int, income : double, expectedTax : double)
+ shouldCalculateCorrectTax()
```
Better Testing

- Use innovative testing techniques
  - Parameterized tests
    - What you need:
      - Some test data
      - A test class with matching fields
      - And some tests

<table>
<thead>
<tr>
<th>Income</th>
<th>Year</th>
<th>Expected Tax</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.00</td>
<td>2007</td>
<td>$0.00</td>
</tr>
<tr>
<td>$10,000.00</td>
<td>2007</td>
<td>$1,950.00</td>
</tr>
<tr>
<td>$20,000.00</td>
<td>2007</td>
<td>$3,900.00</td>
</tr>
<tr>
<td>$38,000.00</td>
<td>2007</td>
<td>$7,410.00</td>
</tr>
<tr>
<td>$38,001.00</td>
<td>2007</td>
<td>$7,410.33</td>
</tr>
<tr>
<td>$40,000.00</td>
<td>2007</td>
<td>$8,070.00</td>
</tr>
<tr>
<td>$60,000.00</td>
<td>2007</td>
<td>$14,670.00</td>
</tr>
<tr>
<td>$100,000.00</td>
<td>2007</td>
<td>$30,270.00</td>
</tr>
</tbody>
</table>

```java
public class TaxCalculationTest {
    private double income;
    private int year;
    private double expectedTax;

    public TaxCalculationTest(int year, double income, double expectedTax) {
        this.year = year;
        this.income = income;
        this.expectedTax = expectedTax;
    }

    public void shouldCalculateCorrectTax() {
        // Implementation
    }
}
```
Better Testing

- Use innovative testing techniques
  - Parameterized tests
    - What you need:
      - Some test data
      - A test class with matching fields
      - And some tests
      - And an annotation

```
@RunWith(Parameterized.class)

taxCalculationTest = new TaxCalculationTest();
```

<table>
<thead>
<tr>
<th>Income</th>
<th>Year</th>
<th>Expected Tax</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.00</td>
<td>2007</td>
<td>$0.00</td>
</tr>
<tr>
<td>$10,000.00</td>
<td>2007</td>
<td>$1,950.00</td>
</tr>
<tr>
<td>$20,000.00</td>
<td>2007</td>
<td>$3,900.00</td>
</tr>
<tr>
<td>$38,000.00</td>
<td>2007</td>
<td>$7,410.00</td>
</tr>
<tr>
<td>$38,001.00</td>
<td>2007</td>
<td>$7,410.33</td>
</tr>
<tr>
<td>$40,000.00</td>
<td>2007</td>
<td>$8,070.00</td>
</tr>
<tr>
<td>$60,000.00</td>
<td>2007</td>
<td>$14,670.00</td>
</tr>
<tr>
<td>$100,000.00</td>
<td>2007</td>
<td>$30,270.00</td>
</tr>
</tbody>
</table>
Better Testing

- Use innovative testing techniques
  - Parameterized tests
    - What you need:
      - Some test data
      - A test class with matching fields
      - And some tests
    - And an annotation

```java
// TaxCalculationTest.java

public class TaxCalculationTest {
    private int year;
    private double income;
    private double expectedTax;

    public TaxCalculationTest(int year, double income, double expectedTax) {
        this.year = year;
        this.income = income;
        this.expectedTax = expectedTax;
    }

    public void testCalculation() {
        double calculatedTax = calculateTax();
        assertEquals(expectedTax, calculatedTax, 0.0001);
    }

    private double calculateTax() {
        // Tax calculation logic
        return 0.0;
    }
}
```
Better Testing

- Use innovative testing techniques
  - Parameterized tests
    - What you need:
      - Some test data
      - A test class with matching fields
      - And some tests
      - And an annotation

<table>
<thead>
<tr>
<th>Income</th>
<th>Year</th>
<th>Expected Tax</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.00</td>
<td>2007</td>
<td>$0.00</td>
</tr>
<tr>
<td>$10,000.00</td>
<td>2007</td>
<td>$1,950.00</td>
</tr>
<tr>
<td>$20,000.00</td>
<td>2007</td>
<td>$3,900.00</td>
</tr>
<tr>
<td>$38,000.00</td>
<td>2007</td>
<td>$7,410.00</td>
</tr>
<tr>
<td>$38,001.00</td>
<td>2007</td>
<td>$7,410.33</td>
</tr>
<tr>
<td>$40,000.00</td>
<td>2007</td>
<td>$8,070.00</td>
</tr>
<tr>
<td>$60,000.00</td>
<td>2007</td>
<td>$14,670.00</td>
</tr>
<tr>
<td>$100,000.00</td>
<td>2007</td>
<td>$30,270.00</td>
</tr>
</tbody>
</table>
Better Testing

- Use innovative testing techniques
  - Parameterized tests
    - What you need:
      - Some test data
      - A test class with matching fields
      - And some tests
      - And an annotation

```
@RunWith(Parameterized.class)
TaxCalculationTest
- income : double
- year : int
- expectedTax : double
+ TaxCalculationTest(year : int, income : double, expectedTax : double)
+ shouldCalculateCorrectTax()
```
Use innovative testing techniques
Better Testing

- Use innovative testing techniques
- Hamcrest asserts
Better Testing

- Use innovative testing techniques
- Hamcrest asserts
  - More expressive and readable test assertions
Better Testing

- Use innovative testing techniques
  - Hamcrest asserts
    - More expressive and readable test assertions
    - Easier to understand
Better Testing

- Use innovative testing techniques
  - Hamcrest asserts
    - More expressive and readable test assertions
    - Easier to understand
    - Less chance of test errors
Better Testing

- Use innovative testing techniques
Better Testing

- Use innovative testing techniques
- Hamcrest asserts
Better Testing

- Use innovative testing techniques
- Hamcrest asserts
  - Traditional JUnit 3.x asserts are hard to read:
Better Testing

- Use innovative testing techniques
  - Hamcrest asserts
    - Traditional JUnit 3.x asserts are hard to read:
      - Parameter order is counter-intuitive for English-speakers
Better Testing

- Use innovative testing techniques
  - Hamcrest asserts
    - Traditional JUnit 3.x asserts are hard to read:
      - Parameter order is counter-intuitive for English-speakers
        - x=10 is written
Better Testing

- Use innovative testing techniques
- Hamcrest asserts
  - Traditional JUnit 3.x asserts are hard to read:
    - Parameter order is counter-intuitive for English-speakers
    - $x=10$ is written `assertEquals(10, x);`
Better Testing

- Use innovative testing techniques
- Hamcrest asserts
  - Traditional JUnit 3.x asserts are hard to read:
    - Parameter order is counter-intuitive for English-speakers
    - \$x=10\$ is written \texttt{assertEquals(10, x);}
    - The statements don't read well for English-speakers
Better Testing

- Use innovative testing techniques
- Hamcrest asserts
  - Traditional JUnit 3.x asserts are hard to read:
    - Parameter order is counter-intuitive for English-speakers
    - x=10 is written as `assertEquals(10, x);`
    - The statements don't read well for English-speakers
    - “Assert that are equal 10 and x”
Better Testing

Use innovative testing techniques

- Hamcrest asserts
  - Traditional JUnit 3.x asserts are hard to read:
    - Parameter order is counter-intuitive for English-speakers
    - x=10 is written \texttt{assertEquals(10, x);}
    - The statements don't read well for English-speakers
    - “Assert that are equal 10 and x”
  - Default error messages are sometimes limited:
Better Testing

- Use innovative testing techniques
  - Hamcrest asserts
    - Traditional JUnit 3.x asserts are hard to read:
      - Parameter order is counter-intuitive for English-speakers
    - x=10 is written `assertEquals(10, x);`
    - The statements don't read well for English-speakers
      - “Assert that are equal 10 and x”
    - Default error messages are sometimes limited:
      ```java
      String color = "yellow";
      assertTrue(color.equals("red") || color.equals("blue") );
      ```
Better Testing

Use innovative testing techniques

- Hamcrest asserts
  - Traditional JUnit 3.x asserts are hard to read:
    - Parameter order is counter-intuitive for English-speakers
    - `x=10` is written `assertEquals(10, x);`
    - The statements don't read well for English-speakers
    - “Assert that are equal 10 and x”
  - Default error messages are sometimes limited:
    ```java
    String color = "yellow";
    assertTrue(color.equals("red") || color.equals("blue"));
    ```
Better Testing

- Use innovative testing techniques
Better Testing

- Use innovative testing techniques
- Hamcrest asserts
Better Testing

- Use innovative testing techniques
- Hamcrest asserts
  - JUnit 4.4 introduces the assertThat statement
Better Testing

- Use innovative testing techniques
  - Hamcrest asserts
    - JUnit 4.4 introduces the `assertThat` statement
    - Rather than writing:
Better Testing

- Use innovative testing techniques
  - Hamcrest asserts
    - JUnit 4.4 introduces the `assertThat` statement
  - Rather than writing:

```java
import static org.junit.Assert.*;
...
assertEquals(expectedTax, calculatedTax, 0);
```
Better Testing

- Use innovative testing techniques
  - Hamcrest asserts
    - JUnit 4.4 introduces the `assertThat` statement
    - Rather than writing:
      ```java
      import static org.junit.Assert.*;
      ...
      assertEquals(expectedTax, calculatedTax, 0);
      ```
    - You can write
Better Testing

- Use innovative testing techniques
  - Hamcrest asserts
    - JUnit 4.4 introduces the `assertThat` statement
    - Rather than writing:
      ```java
      import static org.junit.Assert.*;
      ...
      assertEquals(expectedTax, calculatedTax, 0);
      ```
    - You can write
      ```java
      import static org.hamcrest.Matchers.*;
      ...
      assertThat(calculatedTax, is(expectedTax));
      ```
Better Testing -
*Parameterized Tests and Hamcrest Asserts*
Better Testing

- Use innovative testing techniques
- Groovy Testing
Better Testing

- Use innovative testing techniques
- Groovy Testing
  - Write more expressive unit tests in Groovy
Better Testing

- Use innovative testing techniques
- Groovy Testing
  - Write more expressive unit tests in Groovy
  - Expressive test code
Better Testing

- Use innovative testing techniques
  - Groovy Testing
    - Write more expressive unit tests in Groovy
    - Expressive test code
    - Quicker to write
Better Testing

- Use innovative testing techniques
  - Groovy Testing
    - Write more expressive unit tests in Groovy
    - Expressive test code
    - Quicker to write
    - More incentive to write in-depth test cases
Better Testing

- Use innovative testing techniques
  - Groovy Testing
    - Write more expressive unit tests in Groovy
      - Expressive test code
      - Quicker to write
    - More incentive to write in-depth test cases
  - But
Better Testing

- Use innovative testing techniques
  - Groovy Testing
    - Write more expressive unit tests in Groovy
      - Expressive test code
      - Quicker to write
      - More incentive to write in-depth test cases
    - But
      - Relatively limited IDE support...
Better Testing

- Use innovative testing techniques
Better Testing

- Use innovative testing techniques
- Some examples..
Better Testing

- Use innovative testing techniques
- Some examples..

```java
public class TaxCalculatorBusinessTest {
    private TaxCalculator calculator;

    @Before
    public void setup() {
        calculator = new TaxCalculatorImpl();
    }

    @Test
    public void shouldNotTaxLosses() throws InvalidYearException {
        BigDecimal tax = calculator.calculateIncomeTax(new BigDecimal("-100000"), 2009);
        assertThat(tax, is(new BigDecimal("0.00")));
    }
}
```
Better Testing

- Use innovative testing techniques
- Some examples..

```java
public class TaxCalculatorBusinessTest {
    private TaxCalculator calculator;

    @Before
    public void setup() {
        calculator = new TaxCalculatorImpl();
    }

    @Test
    public void shouldNotTaxLosses() throws InvalidYearException {
        BigDecimal tax = calculator.calculateIncomeTax(new BigDecimal("-100000"), 2009);
        assertThat(tax, is(new BigDecimal("0.00")));
    }
}
```

```groovy
class GroovyTaxCalculatorBusinessTest {
    def calculator = new TaxCalculatorImpl()
    @Test
    void shouldNotTaxLosses() {
        def tax = calculator.calculateIncomeTax(-100000, 2009)
        assertThat tax, is(0.00)
    }
}
```
Better Testing

- Use innovative testing techniques
- Some examples..
Better Testing

- Use innovative testing techniques
- Some examples..

```java
public class TaxCalculatorBusinessTest {

    private TaxCalculator calculator;

    @Before
    public void setup() {
        calculator = new TaxCalculatorImpl();
    }

    @Test(expected=InvalidYearException.class)
    public void shouldNotAcceptUnknownYears() throws InvalidYearException {
        calculator.calculateIncomeTax(new BigDecimal("100000"), 1901);
    }
}
```
Better Testing

- Use innovative testing techniques
- Some examples..

```java
public class TaxCalculatorBusinessTest {

    private TaxCalculator calculator;

    @Before
    public void setup() {
        calculator = new TaxCalculatorImpl();
    }

    @Test(expected=InvalidYearException.class)
    public void shouldNotAcceptUnknownYears() throws InvalidYearException {
        calculator.calculateIncomeTax(new BigDecimal("100000"), 1901);
    }
}
```

```java
class GroovyTaxCalculatorBusinessTest {
    final shouldFail = new GroovyTestCase().&shouldFail
    @Test
    void shouldNotAcceptUnknownYears() {
        shouldFail(InvalidYearException) {
            calculator.calculateIncomeTax(100000, 1901)
        }
    }
}
```
Better Testing

- Use innovative testing techniques
- Behaviour-Driven Development
Better Testing

- Use innovative testing techniques
- Behaviour-Driven Development
  - It’s not about writing tests
Better Testing

- Use innovative testing techniques
  - Behaviour-Driven Development
    - It’s not about writing tests
    - TDD and BDD is about writing better code:
Better Testing

- Use innovative testing techniques
  - Behaviour-Driven Development
    - It’s not about writing tests
    - TDD and BDD is about writing better code:
      - Maintainable
Better Testing

Use innovative testing techniques

Behaviour-Driven Development

- It’s not about writing tests
- TDD and BDD is about writing better code:
  - Maintainable
  - Flexible
Better Testing

- Use innovative testing techniques
  - Behaviour-Driven Development
    - It’s not about writing tests
    - TDD and BDD is about writing better code:
      - Maintainable
      - Flexible
      - Reliable
Better Testing

- Use innovative testing techniques
  - Behaviour-Driven Development
    - It’s not about writing tests
    - TDD and BDD is about writing better code:
      - Maintainable
      - Flexible
      - Reliable
      - Simple
Better Testing

- Use innovative testing techniques
- Behaviour-Driven Development
Use innovative testing techniques

Behaviour-Driven Development

BDD uses words like “should” to describe the desired behaviour of the class:
Better Testing

- Use innovative testing techniques

- Behaviour-Driven Development
  - BDD uses words like “should” to describe the desired behaviour of the class:
  - “Should transfer money from account A to account B”
Better Testing

- Use innovative testing techniques

- Behaviour-Driven Development
  - BDD uses words like “should” to describe the desired behaviour of the class:
    - “Should transfer money from account A to account B”
    - Should deploy landing gear before touching ground”
Better Testing

- Use innovative testing techniques
- Behaviour-Driven Development
  - BDD uses words like “should” to describe the desired behaviour of the class:
    - “Should transfer money from account A to account B”
    - Should deploy landing gear before touching ground”
    - ...

Better Testing

- Use innovative testing techniques
- Enter Easyb
Better Testing

- Use innovative testing techniques
- Enter Easyb
  - A BDD testing framework for Java
Better Testing

- Use innovative testing techniques
- Enter Easyb
  - A BDD testing framework for Java
  - Make testing clearer and easier to write
Better Testing

- Use innovative testing techniques
- Enter Easyb
  - A BDD testing framework for Java
  - Make testing clearer and easier to write
  - Make tests self-documenting
Better Testing

- Use innovative testing techniques
  - Enter Easyb
    - A BDD testing framework for Java
      - Make testing clearer and easier to write
      - Make tests self-documenting
      - Help developers focus on the requirements
Better Testing

- Use innovative testing techniques
- Easyb stories:
Better Testing

- Use innovative testing techniques
- Easyb stories:
  - Use a narrative approach
Better Testing

- Use innovative testing techniques
- Easyb stories:
  - Use a narrative approach
  - Describe a precise requirement
Better Testing

- Use innovative testing techniques

- Easyb stories:
  - Use a narrative approach
  - Describe a precise requirement
  - Can be understood by a stakeholder
Better Testing

- Use innovative testing techniques

- Easyb stories:
  - Use a narrative approach
  - Describe a precise requirement
  - Can be understood by a stakeholder
  - Usually made up of a set of scenarios
Better Testing

- Use innovative testing techniques
  - Easyb stories:
    - Use a narrative approach
    - Describe a precise requirement
    - Can be understood by a stakeholder
    - Usually made up of a set of scenarios
    - Use an easy-to-understand structure:
Better Testing

- Use innovative testing techniques

  - Easyb stories:
    - Use a narrative approach
    - Describe a precise requirement
    - Can be understood by a stakeholder
    - Usually made up of a set of scenarios
    - Use an easy-to-understand structure:
      - Given [a context]...
Better Testing

- Use innovative testing techniques

**Easyb stories:**
- Use a narrative approach
- Describe a precise requirement
- Can be understood by a stakeholder
- Usually made up of a set of scenarios
- Use an easy-to-understand structure:
  - *Given [a context]*...
  - *When [something happens]*...
Better Testing

- Use innovative testing techniques

  - Easyb stories:
    - Use a narrative approach
    - Describe a precise requirement
    - Can be understood by a stakeholder
    - Usually made up of a set of scenarios
    - Use an easy-to-understand structure:
      - Given [a context]...
      - When [something happens]...
      - Then [something else happens]...
Better Testing

- Use innovative testing techniques
- Enter Easyb
Better Testing

- Use innovative testing techniques
- Enter Easyb
  - A Groovy-based DSL for Behaviour-Driven Development
Better Testing

- Use innovative testing techniques
- Enter Easyb
  - A Groovy-based DSL for Behaviour-Driven Development

```java
import com.wakaleo.jpt.taxcalculator.InvalidYearException

scenario "Should not tax losses", {
     given "a correctly configured tax calculator",
     when "you calculate annual income tax for a negative income",
     then "the calculated tax should be zero",
}
```
Better Testing

- Use innovative testing techniques
- Enter Easyb
- A Groovy-based DSL for Behaviour-Driven Development

```groovy
import com.wakaleo.jpt.taxcalculator.InvalidYearException

scenario "Should not tax losses", {
    given "a correctly configured tax calculator",
    when "you calculate annual income tax for a negative income",
    then "the calculated tax should be zero",
}
```
Better Testing

- Use innovative testing techniques
- Easyb reporting
Better Testing

- Use innovative testing techniques
- Easyb reporting
  - Simple but readable...
Better Testing

- Use innovative testing techniques
- Easyb reporting
- Simple but readable

```
5 scenarios (including 1 pending) executed successfully

Story: account deposits

scenario Make initial deposit onto a new account
    given a newly created account
    when an initial deposit is made into this account
    then the account balance should be equal to the amount deposited [PENDING]

scenario Make initial deposit onto a new account
    given a new account
    when an initial deposit is made
    then the balance should be equal to the amount deposited

scenario Depositing money to an existing account
    given an account with $100
    when an amount is deposited
    then the amount is added to the balance
...
```
Better Testing - *Testing with Groovy*
Better Testing

- Use innovative testing techniques
- Separate unit tests and integration tests
Better Testing

- Use innovative testing techniques
- Separate unit tests and integration tests
  - **Unit tests**
Better Testing

- Use innovative testing techniques

- Separate unit tests and integration tests
  - **Unit tests**
    - Run quickly
Better Testing

- Use innovative testing techniques
- Separate unit tests and integration tests
  - **Unit tests**
    - Run quickly
    - Run in isolation
Better Testing

- Use innovative testing techniques
  - Separate unit tests and integration tests
    - **Unit tests**
      - Run quickly
      - Run in isolation
      - Fail quickly
Better Testing

- Use innovative testing techniques

  - Separate unit tests and integration tests
    - Unit tests
      - Run quickly
      - Run in isolation
      - Fail quickly
    - Integration tests
Better Testing

- Use innovative testing techniques
  - Separate unit tests and integration tests
    - **Unit tests**
      - Run quickly
      - Run in isolation
      - Fail quickly
    - **Integration tests**
      - Run less quickly
Better Testing

- Use innovative testing techniques

- Separate unit tests and integration tests
  - **Unit tests**
    - Run quickly
    - Run in isolation
    - Fail quickly
  - **Integration tests**
    - Run less quickly
    - Use external resources (databases, app servers,...)
Better Testing

- Use innovative testing techniques
- Integration Test strategies
Better Testing

- Use innovative testing techniques
- Integration Test strategies
  - Deploy to an embedded Jetty server for local integration tests
Better Testing

- Use innovative testing techniques
- Integration Test strategies
  - Deploy to an embedded Jetty server for local integration tests
  - Run local integration tests with Selenium or Canoo WebTest
Better Testing

- Use innovative testing techniques
- Integration Test strategies
  - Deploy to an embedded Jetty server for local integration tests
  - Run local integration tests with Selenium or Canoo WebTest
  - Deploy automatically to a publicly-visible integration server
Better Testing

- Use innovative testing techniques
- Unit and integration testing on the development machine
Better Testing

- Use innovative testing techniques
- Unit and integration testing on the development machine
- Fast running unit tests first

mvn test
Better Testing

- Use innovative testing techniques
- Unit and integration testing on the development machine
  - Fast running unit tests first
  - Integration and web tests using Selenium and Jetty
Better Testing

- Use innovative testing techniques
- Unit and integration testing on the development machine
  - Fast running unit tests first
  - Integration and web tests using Selenium and Jetty
  - Commit changes to Subversion when done
Better Testing

- Use innovative testing techniques
- Official automated build on the CI server
  - Check out latest code from Subversion
  - Compile and unit tests
  - Integration and web tests using Selenium and Jetty
  - Deployment to an integration server
  - Publish on the Maven repository

```
mvn test
mvn integration-test
svn commit
```
Better Testing

- Use innovative testing techniques
  - Official automated build on the CI server
    - Check out latest code from Subversion
    - Compile and unit tests
    - Integration and web tests using Selenium and Jetty
    - Deployment to an integration server
    - Publish on the Maven repository
Better Testing

- Use innovative testing techniques
- Official automated build on the CI server
Better Testing

- Use innovative testing techniques
- Official automated build on the CI server
- Check out latest code from Subversion
Better Testing

- Use innovative testing techniques
- Official automated build on the CI server
  - Check out latest code from Subversion
  - Compile and unit tests
Better Testing

- Use innovative testing techniques
- Official automated build on the CI server
  - Check out latest code from Subversion
  - Compile and unit tests
  - Integration and web tests using Selenium and Jetty
Better Testing

- Use innovative testing techniques
- Official automated build on the CI server
  - Check out latest code from Subversion
  - Compile and unit tests
  - Integration and web tests using Selenium and Jetty
  - Deployment to an integration server

Diagram:
- Subversion
  - svn update
  - mvn test
  - mvn integration-test
  - mvn tomcat:deploy
- Integration Server
- Build Server
Better Testing

- Use innovative testing techniques
- Official automated build on the CI server
  - Check out latest code from Subversion
  - Compile and unit tests
  - Integration and web tests using Selenium and Jetty
  - Deployment to an integration server
  - Publish on the Maven repository
Better Testing

- Use innovative testing techniques
- Automated releases on the CI server
  - Manually-launched build job
  - Runs Maven Release process
  - Deployment to a test server

Integration Server

Subversion

- `svn update`
- `mvn test`
- `mvn integration-test`
- `mvn tomcat:deploy`
- `mvn deploy`

Build Server
Better Testing

- Use innovative testing techniques
- Automated releases on the CI server
  - Manually-launched build job
  - Runs Maven Release process
  - Deployment to a test server

Subversion

Build Server
Better Testing

- Use innovative testing techniques
- Automated releases on the CI server
Better Testing

- Use innovative testing techniques
- Automated releases on the CI server
- Manually-launched build job

Subversion

Build Server
Better Testing

- Use innovative testing techniques
- Automated releases on the CI server
  - Manually-launched build job
  - Runs Maven Release process
Better Testing

- Use innovative testing techniques
- Automated releases on the CI server
  - Manually-launched build job
  - Runs Maven Release process

```
mvn release:prepare
```
Better Testing

- Use innovative testing techniques
- Automated releases on the CI server
  - Manually-launched build job
  - Runs Maven Release process

```
mvn release:prepare
mvn release:perform
```
Better Testing

- Use innovative testing techniques
- Automated releases on the CI server
  - Manually-launched build job
  - Runs Maven Release process
  - Deployment to a test server

```
mvn release:prepare
mvn release:perform
```
Better Testing

- Use innovative testing techniques
- Automated releases on the CI server
  - Manually-launched build job
  - Runs Maven Release process
  - Deployment to a test server

```
mvn release:prepare
mvn release:perform
mvn tomcat:deploy
```
Better Testing

- Use innovative testing techniques
- Automated releases on the CI server
  - Manually-launched build job
  - Runs Maven Release process
  - Deployment to a test server

```bash
mvn release:prepare
deploy
```

Test Server

Build Server
Better Testing

*Integration testing strategies*
Automated Deployment

Automate the deployment process
Automated Deployment

- Automate the deployment process
- Automated snapshot deployment
Automated Deployment

- Automate the deployment process
- Automated snapshot deployment
  - Make snapshot releases automatically available for other teams
Automated Deployment

- Automate the deployment process
- Automated snapshot deployment
  - Make snapshot releases automatically available for other teams
  - Developers don’t have to update and install modules locally
Automated Deployment

- Automate the deployment process
- Automated snapshot deployment
  - Make snapshot releases automatically available for other teams
  - Developers don’t have to update and install modules locally
  - Run `mvn deploy` on the build server
Automated Deployment

- Automate the deployment process
- Automated snapshot deployment
  - Make snapshot releases automatically available for other teams
  - Developers don’t have to update and install modules locally
  - Run `mvn deploy` on the build server
Automated Deployment

- Automate the deployment process
- Automated snapshot deployment
  - Make snapshot releases automatically available for other teams
  - Developers don’t have to update and install modules locally
  - Run `mvn deploy` on the build server
Automated Deployment

- Automate the deployment process
- Automated snapshot deployment
  - Make snapshot releases automatically available for other teams
  - Developers don’t have to update and install modules locally
- Run `mvn deploy` on the build server
Automated Deployment

- Automate the deployment process
- Automated snapshot deployment
  - Make snapshot releases automatically available for other teams
  - Developers don’t have to update and install modules locally
- Run **mvn deploy** on the build server
Automated Deployment

Automate the deployment process

Automated snapshot deployment

- Make snapshot releases automatically available for other teams
- Developers don’t have to update and install modules locally
- Run **mvn deploy** on the build server
Automate the deployment process
Automated Code Quality

Why use code quality metrics
Automated Code Quality

- Why use code quality metrics
- Better quality code
Automated Code Quality

- Why use code quality metrics
- Better quality code
- Code is easier to maintain
Automated Code Quality

- Why use code quality metrics
  - Better quality code
  - Code is easier to maintain
  - Detect potential bugs
Automated Code Quality

Why use code quality metrics

- Better quality code
- Code is easier to maintain
- Detect potential bugs
- Train new staff
Automated Code Quality

- Why automate code quality metrics
Automated Code Quality

- Why automate code quality metrics
- Global picture of code quality
Automated Code Quality

- Why automate code quality metrics
- Global picture of code quality
- Statistical code quality trends
Automated Code Quality

- Why automate code quality metrics
- Global picture of code quality
- Statistical code quality trends
- More value-added manual reviews
Automated Code Quality

- Types of code quality metrics
Automated Code Quality

- Types of code quality metrics
- Coding Standards
Automated Code Quality

- Types of code quality metrics
  - Coding Standards
    - Naming conventions, Javadoc comments, layout,...
Types of code quality metrics

Coding Standards

- Naming conventions, Javadoc comments, layout,...
- Tool: Checkstyle
Automated Code Quality

Types of code quality metrics

- **Coding Standards**
  - Naming conventions, Javadoc comments, layout,...
  - Tool: **Checkstyle**

- **Best Practices**
Automated Code Quality

- Types of code quality metrics
  - **Coding Standards**
    - Naming conventions, Javadoc comments, layout,...
  - Tool: **Checkstyle**
  - **Best Practices**
    - Good programming habits, potential bugs,...
Types of code quality metrics

**Coding Standards**
- Naming conventions, Javadoc comments, layout,...
- Tool: Checkstyle

**Best Practices**
- Good programming habits, potential bugs,...
- Tools: Checkstyle, PMD, Findbugs
Automated Code Quality

- Types of code quality metrics
  - **Coding Standards**
    - Naming conventions, Javadoc comments, layout,...
    - Tool: **Checkstyle**
  - **Best Practices**
    - Good programming habits, potential bugs,...
    - Tools: **Checkstyle, PMD, Findbugs**
  - **Bug Detection**
Types of code quality metrics

Coding Standards
- Naming conventions, Javadoc comments, layout,...
- Tool: Checkstyle

Best Practices
- Good programming habits, potential bugs,...
- Tools: Checkstyle, PMD, Findbugs

Bug Detection
- Broken code, dangerous code, bugs,...
Automated Code Quality

Types of code quality metrics

Coding Standards
- Naming conventions, Javadoc comments, layout,...
- Tool: **Checkstyle**

Best Practices
- Good programming habits, potential bugs,...
- Tools: **Checkstyle, PMD, Findbugs**

Bug Detection
- Broken code, dangerous code, bugs,...
- Tool: **FindBugs**
Automated Code Quality

- Types of code quality metrics
  - **Coding Standards**
    - Naming conventions, Javadoc comments, layout,...
    - Tool: **Checkstyle**
  - **Best Practices**
    - Good programming habits, potential bugs,...
    - Tools: **Checkstyle, PMD, Findbugs**
  - **Bug Detection**
    - Broken code, dangerous code, bugs,...
    - Tool: **FindBugs**
  - **Code Coverage**
Automated Code Quality

- Types of code quality metrics
  
  - **Coding Standards**
    - Naming conventions, Javadoc comments, layout,...
    - Tool: **Checkstyle**
  
  - **Best Practices**
    - Good programming habits, potential bugs,...
    - Tools: **Checkstyle, PMD, Findbugs**
  
  - **Bug Detection**
    - Broken code, dangerous code, bugs,...
    - Tool: **FindBugs**
  
  - **Code Coverage**
    - How much code is executed by your tests
Automated Code Quality

- Types of code quality metrics
  - **Coding Standards**
    - Naming conventions, Javadoc comments, layout,...
    - Tool: **Checkstyle**
  - **Best Practices**
    - Good programming habits, potential bugs,...
    - Tools: **Checkstyle, PMD, Findbugs**
  - **Bug Detection**
    - Broken code, dangerous code, bugs,...
    - Tool: **FindBugs**
  - **Code Coverage**
    - How much code is executed by your tests
    - Tool: **Cobertura, Emma, Clover,**...
Automated Code Quality

- Other types of code quality metrics
Automated Code Quality

- Other types of code quality metrics
  - Build metrics
Automated Code Quality

- Other types of code quality metrics
  - Build metrics
    - How long do the builds take to run
Automated Code Quality

- Other types of code quality metrics
  - **Build metrics**
    - How long do the builds take to run
    - How often do they fail
Automated Code Quality

Other types of code quality metrics

*Build metrics*
- How long do the builds take to run
- How often do they fail

*Test metrics*
Automated Code Quality

Other types of code quality metrics

*Build metrics*
- How long do the builds take to run
- How often do they fail

*Test metrics*
- How long do tests take to run
Automated Code Quality

Other types of code quality metrics

*Build metrics*
- How long do the builds take to run
- How often do they fail

*Test metrics*
- How long do tests take to run
- How often do they fail
Automated Code Quality

- Other types of code quality metrics
  - **Build metrics**
    - How long do the builds take to run
    - How often do they fail
  - **Test metrics**
    - How long do tests take to run
    - How often do they fail
    - How long does it take to fix the bugs
Other types of code quality metrics

*Build metrics*
- How long do the builds take to run
- How often do they fail

*Test metrics*
- How long do tests take to run
- How often do they fail
- How long does it take to fix the bugs
- Are any particular tests failing more often than others?
Automated Code Quality

- Team code reviews
Automated Code Quality

- Team code reviews
  - Review code as a group
Automated Code Quality

- Team code reviews
  - Review code as a group
  - Long and slow if done manually
Automated Code Quality

Team code reviews

- Review code as a group
- Long and slow if done manually
- Benefits greatly from the use of tools
Automated Code Quality

- Automating code quality metrics with Hudson
Automated Code Quality

- Automating code quality metrics with Hudson
- Using the Violations plugin
Automated Code Quality

- Automating code quality metrics with Hudson
- Using the Violations plugin
  - Reports on Checkstyle, PMD, Findbugs, and others
Automated Code Quality

Automating code quality metrics with Hudson

Using the Violations plugin

- Reports on Checkstyle, PMD, Findbugs, and others
- Uses data generated with Maven or Ant
Automated Code Quality

- Automating code quality metrics with Hudson
- Configuring the Violations plugin
Automated Code Quality

- Automating code quality metrics with Hudson
- Displaying the Violations reports
Automated Code Quality

Automating code quality metrics with Hudson

Displaying the Violations reports

<table>
<thead>
<tr>
<th>Type</th>
<th>Violations</th>
<th>Files in violation</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkstyle</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>cpd</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>findbugs</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>pmd</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Violations Report for build 105

checkstyle chart

Files:
Automated Code Quality

Automating code quality metrics with Hudson

Displaying the Violations details

```
checkstyle 4 violations

21  Utility classes should not have a public or default constructor.
24  Avoid inline conditionals.
66  '32' is a magic number.
112 Missing a Javadoc comment.

File: HashCodeUtil.java Lines 12 to 44

```
Automated Code Quality

- Code Coverage
Automated Code Quality

- Code Coverage
- What can code coverage metrics tell you:
Automated Code Quality

Code Coverage

What can code coverage metrics tell you:

Indicates what code is being executed by your unit tests.
Automated Code Quality

Code Coverage

What can code coverage metrics tell you:

- Indicates what code is being executed by your unit tests.
- Can help isolate untested code
Automated Code Quality

- Code Coverage
  - What can code coverage metrics tell you:
    - Indicates what code is being executed by your unit tests.
    - Can help isolate untested code
    - Does **not** guarantee that the tests that are run are of high quality
Automated Code Quality

Code Coverage

What can code coverage metrics tell you:
- Indicates what code is being executed by your unit tests.
- Can help isolate untested code.
- Does not guarantee that the tests that are run are of high quality.
- Can be an indicator of whether tests are being written at all.
Automated Code Quality

- Code Coverage
Automated Code Quality

- Code Coverage
- Using the Hudson code coverage plugins
Automated Code Quality

- Code Coverage
  - Using the Hudson code coverage plugins
    - Generate data using Ant or Maven
Automated Code Quality

Code Coverage

- Using the Hudson code coverage plugins
  - Generate data using Ant or Maven
  - Report coverage metrics in Hudson
Automated Code Quality

- Code Coverage
  - Using the Hudson code coverage plugins
  - Generate data using Ant or Maven
  - Report coverage metrics in Hudson
Automated Code Quality

- Code Coverage
Automated Code Quality

- Code Coverage
- Using the Hudson code coverage plugins
Automated Code Quality

- Code Coverage
  - Using the Hudson code coverage plugins
    - Cobertura coverage reports
Automated Code Quality

- Code Coverage
- Using the Hudson code coverage plugins
- Cobertura coverage reports
Automated Code Quality

- Code Coverage
- Using the Hudson code coverage plugins
- Cobertura coverage reports

**Code Coverage**

Cobertura Coverage Report

**Trend**

<table>
<thead>
<tr>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
</tr>
<tr>
<td>95%</td>
</tr>
<tr>
<td>90%</td>
</tr>
<tr>
<td>85%</td>
</tr>
<tr>
<td>80%</td>
</tr>
<tr>
<td>75%</td>
</tr>
<tr>
<td>70%</td>
</tr>
<tr>
<td>65%</td>
</tr>
<tr>
<td>60%</td>
</tr>
<tr>
<td>55%</td>
</tr>
<tr>
<td>50%</td>
</tr>
<tr>
<td>45%</td>
</tr>
<tr>
<td>40%</td>
</tr>
<tr>
<td>35%</td>
</tr>
<tr>
<td>30%</td>
</tr>
<tr>
<td>25%</td>
</tr>
<tr>
<td>20%</td>
</tr>
<tr>
<td>15%</td>
</tr>
<tr>
<td>10%</td>
</tr>
<tr>
<td>5%</td>
</tr>
<tr>
<td>0%</td>
</tr>
</tbody>
</table>

**Project Coverage Summary**

<table>
<thead>
<tr>
<th>Name</th>
<th>Packages</th>
<th>Files</th>
<th>Classes</th>
<th>Methods</th>
<th>Lines</th>
<th>Conditionals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cobertura Coverage Report</td>
<td>100% (5/5)</td>
<td>100% (12/12)</td>
<td>100% (12/12)</td>
<td>95% (114/122)</td>
<td>95% (386/407)</td>
<td>100% (0/0)</td>
</tr>
</tbody>
</table>

**Coverage Breakdown by Package**

<table>
<thead>
<tr>
<th>Package</th>
<th>Files</th>
<th>Classes</th>
<th>Methods</th>
<th>Lines</th>
<th>Conditionals</th>
</tr>
</thead>
<tbody>
<tr>
<td>nz.gpotird.cde.cs.dto.pj.portlet.propertyeditorfs</td>
<td>100% (3/1)</td>
<td>100% (3/3)</td>
<td>100% (10/10)</td>
<td>100% (0/0)</td>
<td></td>
</tr>
<tr>
<td>nz.gpotird.cde.cs.dto.pj.portlet.util</td>
<td>100% (5/5)</td>
<td>100% (5/5)</td>
<td>78% (21/27)</td>
<td>62% (62/76)</td>
<td>100% (0/0)</td>
</tr>
<tr>
<td>nz.gpotird.cde.cs.dto.pj.portlet.chartrend</td>
<td>100% (3/1)</td>
<td>100% (3/3)</td>
<td>100% (10/10)</td>
<td>100% (0/0)</td>
<td></td>
</tr>
<tr>
<td>nz.gpotird.cde.cs.dto.pj.portlet.chartrendvalidate</td>
<td>100% (5/5)</td>
<td>100% (5/5)</td>
<td>78% (21/27)</td>
<td>62% (62/76)</td>
<td>100% (0/0)</td>
</tr>
</tbody>
</table>
Automated Code Quality

- **Code Coverage**
- Using the Hudson code coverage plugins
- Cobertura coverage reports

```java
  43 */
  44   public Manifest getManifest() {
  45     if (manifest == null) {
  46       loadMetaInf();
  47     }
  48     return manifest;
  49   }
  50
  51   public void setManifest(Manifest manifest) {
  52     this.manifest = manifest;
  53   }
  54
  55   public ServletContext getContxt() {
  56     return context;
  57   }
  58
  59   public void setContxt(ServletContext context) {
  60     this.context = context;
  61   }
  62
  63   /**
  64    * Load the META-INF file.
  65   */
  66   public void loadMetaInf() {
  67     String appServerHome = context.getRealPath("/");
  68     File manifestFile = new File(appServerHome, "META-INF/MANIFEST.MF");
```
Automated Code Quality

- More advanced code quality metrics
Automated Code Quality

- More advanced code quality metrics
- How long do your tests run
Automated Code Quality

- More advanced code quality metrics
- How long do your tests run
  - Overly-long tests slow down your build process
Automated Code Quality

- More advanced code quality metrics
  - How long do your tests run
    - Overly-long tests slow down your build process
    - They may also indicate a performance issue
Automated Code Quality

- More advanced code quality metrics
- How long do your tests run
  - Overly-long tests slow down your build process
  - They may also indicate a performance issue
Automated Code Quality

- More advanced code quality metrics
- How long do your tests run
  - Overly-long tests slow down your build process
  - They may also indicate a performance issue

![Build duration chart]
Automated Code Quality

- More advanced code quality metrics
Automated Code Quality

- More advanced code quality metrics
- Are your tests failing repeatedly
Automated Code Quality

- More advanced code quality metrics
- Are your tests failing repeatedly
  - May indicate a difficult technical issue
Automated Code Quality

- More advanced code quality metrics
  - Are your tests failing repeatedly
    - May indicate a difficult technical issue
    - Could result in unreliable “quick-fix” solutions
Automated Code Quality

- More advanced code quality metrics
- Are your tests failing repeatedly
  - May indicate a difficult technical issue
  - Could result in unreliable “quick-fix” solutions
Automated Code Quality

- More advanced code quality metrics
- Are your tests failing repeatedly
- May indicate a difficult technical issue
- Could result in unreliable “quick-fix” solutions
There are many ways to improve a development process
Summary

- There are many ways to improve a development process
- Improve your build process and dependency management
Summary

- There are many ways to improve a development process
- Improve your build process and dependency management
- Install a CI server and publish your APIs automatically
Summary

- There are many ways to improve a development process
- Improve your build process and dependency management
- Install a CI server and publish your APIs automatically
- Automate as much as possible!
Summary

- There are many ways to improve a development process
- Improve your build process and dependency management
- Install a CI server and publish your APIs automatically
- Automate as much as possible!
- Use code quality metrics to isolate bottlenecks and fix problems
Automated code quality
Automate with audacity!

You have only your most boring tasks to loose!
Thanks for your attention!
Thanks for your attention!
Thanks for your attention!
Thanks for your attention!