



Enter the Gradle

Hans Dockter
CEO, Gradleware
Founder Gradle
hans.dockter@gradleware.com

What you will learn

- ▶ Declarativeness
- ▶ Extensibility
- ▶ Performance Features
- ▶ Build Integration
- ▶ Build Migration
- ▶ Testing
- ▶ Discoverability
- ▶ Multiproject Builds
- ▶ Eclipse Integration
- ▶ Gradle Bootstrap Install
- ▶ C/C++
- ▶ Sonar Integration
- ▶ Discovery

Intro

What is Gradle?

- ▶ A general purpose build system
- ▶ Groovy DSL with a Java core.
- ▶ Provides build-in support for Java, Groovy, Scala, Web, OSGi, EAR and many more types.
- ▶ Exciting solutions for many of the big pain points you often have with current build systems.
 - Maintainability
 - Performance
 - Usability

Gradle Project Background

- ▶ Very active community (mailing-list, patches, issues)
- ▶ Apache v2 license.
- ▶ Excellent user's guide (300 pages) + many samples
- ▶ Excellent DSL reference
- ▶ Frequent releases, multiple commits per day
- ▶ Quality is king:
 - ▶ 4500 unit tests, Many hundreds of integration test
 - ▶ Healthy codebase
 - ▶ low defect rate
- ▶ New Committers and Gradleware Employees:
 - ▶ Szczepan Faber (Mr. Mockito)
 - ▶ Peter Niederwieser (Mr. Spock)
 - ▶ Luke Daley (Grails committer and Geb Founder)
 - ▶ Daz DeBoer (Original contributor to Selenium and Ant)



Community Portal

- ▶ Forum: forums.gradle.org
- ▶ Keep up to date: This Week in Gradle
- ▶ Roadmap: gradle.org/roadmap

Gradle is Declarative

Declarative

You specify the **WHAT**

Gradle figures out the **HOW**

Labs

- ⚡ Demo - Source Sets



Gradle
is
declarative
without
being rigid

Extensible Build Language

vs.

Build Framework

Custom Language Elements

```
usePlugin 'editions'

productEditions {
    enterprise core, plugins, powerAddons
    public core, plugins, openApi
}
```

```
>gradle enterpriseEditionZip
```

```
>gradle publicEditionTar
```

```

TestNBalanceClass=com.context13.green.testnbalance.webinterface.Application

<!-- Target: build-testnbalance -->
<target name="build-testnbalance" depends="init,cbs,launchmodes">
<ant antfile="${build.dir}/languages.xml"/>
<ant antfile="${build.dir}/module-common.xml" target="build"/>
<ant antfile="${build.dir}/module-comm.xml" target="build"/>
<ant antfile="${build.dir}/module-testnbalance.xml" target="build"/>
</target>

<target name="testnbalance_setup" depends="init-build-tools">
<ant antfile="${build.dir}/sem.xml" target="sem_setup">
<property name="sem.code" value="oe"/>
<property name="sem.variant" value="oe-tnb"/>
<property name="sem.skin" value="stdskin"/>
</ant>
</target>

<testnbalance vmmname="java"
    vmargs=" ${toolsVmArgs} ${unixVmArgs} ${libpath}"
    classpath=" ${unixClasspath.testnbalance}"
    mainclass=" ${TestNBalanceClass}" />

<testnbalance vmmname="java"
    vmargs=" ${toolsVmArgs} ${unixVmArgs} ${libpath}"
    classpath=" ${unixClasspath.testnbalance}"
    mainclass=" ${TestNBalanceClass}" />

<property name="module.testnbalance.classes.dir" value="${build.dir}/testnbalance"/>

<property name="module.testnbalance.lib.dir" value="${build.dir}/lib"/>

<property name="module.testnbalance.jar" value="${module.testnbalance.lib.dir}/testnbalance.jar"/>

<target name="testnbalance_get">
<vsget login="${vss.login}" serverPath="${vss.serverPath}"
    vsspath="/webctrl/source/testnbalance" recursive="true" writable="false" ssdir="${vss.ssdir}"/>
<vsget login="${vss.login}" serverPath="${vss.serverPath}"
    vsspath="/webctrl/classes/testnbalance" recursive="true" writable="false" ssdir="${vss.ssdir}"/>
<vsget login="${vss.login}" serverPath="${vss.serverPath}"
    vsspath="/webctrl/testnbalance" recursive="true" writable="false" ssdir="${vss.ssdir}"/>
</target>

<delete dir="${module.testnbalance.classes.dir}/com"/>
<delete failonerror="false"><fileset dir="${module.testnbalance.classes.dir}" includes="*.class"/></delete>

<import file="module-testnbalance-properties.xml"/>
<convert module="testnbalance"/>

<testnbalance vmmname="java"
    vmargs=" ${toolsVmArgs} ${libpath}"
    classpath=" ${winClasspath.testnbalance}"
    mainclass=" ${TestNBalanceClass}" />

<ant antfile="${build.dir}/module-testnbalance.xml" target="@{target}" />

```

```

<?xml version="1.0" encoding="utf-8"?>
<project name="testnbalance" default="build" basedir=". . .>

<!-- IMPORTS -->
<import file="macros.xml"/>
<import file="modulebase.xml"/>
<import file="module-${ant.project.name}-properties.xml"/>

<!-- DEPENDENCIES -->
<target name="foreachdependency">
<property name="target" value="build"/>
<ant target="!${target}" antfile="projects/build/module-comm.xml"/>
</target>

<!-- BUILD -->
<target name="jar" depends="modulebase.jar">
<copy module="${ant.project.name}"/>
</target>

<!-- DIST -->
<condition property="project.dist.tnb.dir" value="${dist.dir}/oe-tnb" else="${dist.dir}"/> <isset property="install.dist"/> </condition>
<fileset id="project.distfiles.tnb" dir=". . .>
<include name=" ${lib.dir}/testnbalance-images.jar"/>
<include name=" ${properties.dir}/testnbalance-product.properties"/>
<include name=" ${classes.dir}/common/resources/general/logo_testnbalance.png"/>
<include name=" ${properties.dir}/family.properties"/>
<!-- excluded files -->
<exclude name="*/_sem_*"/>
<exclude name="*/_os_*"/>
</fileset>

<fileset id="project.distfiles.opentools" dir=". . .>
</fileset>

<fileset id="project.distfiles.tnb.lib" dir="${base.dist.dir}"/>
<include name=" ${module.testnbalance.jar}"/>
</fileset>

<target name="dist">
<mkdir dir="${project.dist.tnb.dir}"/>
<copy todir="${project.dist.tnb.dir}">
<fileset refid="project.distfiles.tnb"/>
</copy>
<antcall target="modulebase.-dist"/>
</target>

<!-- UNITTEST -->
<target name="install_test">
<copy nightytest.module="testnbalance">
<test-files>
<include name="*/ZTest.class"/>
</test-files>
</copy>
</nightytest>
</target>

</project>

```

```

<?xml version="1.0" encoding="utf-8"?>
<project name="module-testnbalance-properties" basedir=". . .>

<!-- IMPORTS -->
<import file="macros.xml"/>
<import file="module-common-properties.xml"/>
<import file="module-comm-properties.xml"/>
<import file="properties.xml"/>

<!-- PROPERTIES -->
<property name="module.testnbalance.source.dir" value="${source.dir}/testnbalance"/>
<property name="module.testnbalance.classes.dir" value=" ${classes.dir}/testnbalance"/>
<property name="module.testnbalance.depache.dir" value=" ${depache.dir}/testnbalance"/>
<property name="module.testnbalance.jar" value=" ${lib.dir}/testnbalance.jar"/>

<!-- IMPORTANT NOTE: ANY CHANGES MADE IN THIS FILE MUST ALSO BE MADE IN THE IDEA PROJECT SETTINGS -->
<!-- Please note that this file is structured to mirror the order of modules as is represented in the Idea Project Settings dialog. This should make it easier to edit this file and propagate the changes to Idea. -->
<!-- classpath.compiletime.<module> -->
<!-- These path definitions represent the libraries used by the given module at compile time. Module dependencies are represented by adding the compiletime definitions of the other modules it uses. For clarity, please explicitly include all modules that are used. -->
<!-- classpath.runtime.<module> -->
<!-- These path definitions represent the classpath used by the given module at runtime. Module dependencies are represented by adding the runtime definitions of the modules being used. NOTE: the module jar should be listed after the module's classpath definition. -->
</property>

<path id="classpath.compiletime.testnbalance">
<element location=" ${module.testnbalance.classes.dir}"/>
<path refid="classpath.compiletime.common"/>
<path refid="classpath.compiletime.comm"/>
</path>

<path id="classpath.runtime.testnbalance">
<path refid="classpath.compiletime.testnbalance"/>
<element location=" ${module.testnbalance.jar}"/>
<path refid="classpath.runtime.common"/>
<path refid="classpath.runtime.comm"/>
<element location=" ${lib.dir}/testnbalance-images.jar"/>
</path>

</project>

```

```
products 'webserver', 'permissionskeybuilder', 'sitebuilder', 'logicbuilder', 'viewbuilder', 'virtualbacview', 'wapbuilder', 'testnbalance'

dependencies
{
    compile project(':common')
    compile project(':comm')
}

modules 'testnbalance'

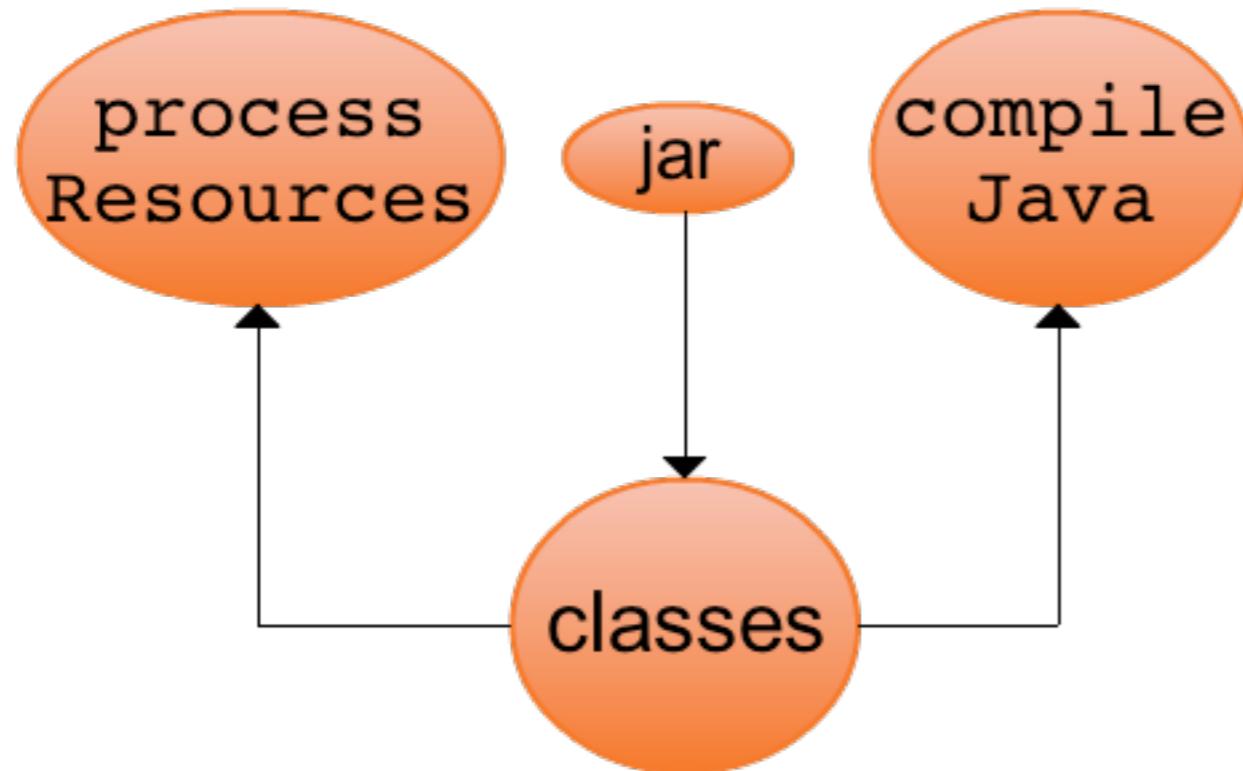
requiresLicense false

launcher
{
    mainClass = 'com.controlj.green.testnbalance.userinterface.Application'
    useJavawOnWindows = true
    forceClientVM = true
}

doc
{
    summary '''TBD.'''
}
```

Extensible

Directed Acyclic Graph (DAG)



- ▶ Each task to be executed is a node.
- ▶ The dependsOn relations define directed edges.
- ▶ No cycles are allowed (acyclic)
- ▶ Each task is executed once and only once.
- ▶ Execution order is against the edge directions.

Expect the unexpected

- ▶ Custom Language Elements
- ▶ Deep Configuration API
- ▶ Deep Execution API
- ▶ Rich API
- ▶ Extendable Domain Objects
- ▶ Custom Tasks
- ▶ Custom Plugins

Labs

- ⚡ Demo - Task Rules



Groovy vs.XML

It's the design, stupid!

Please
no
messy
build scripts

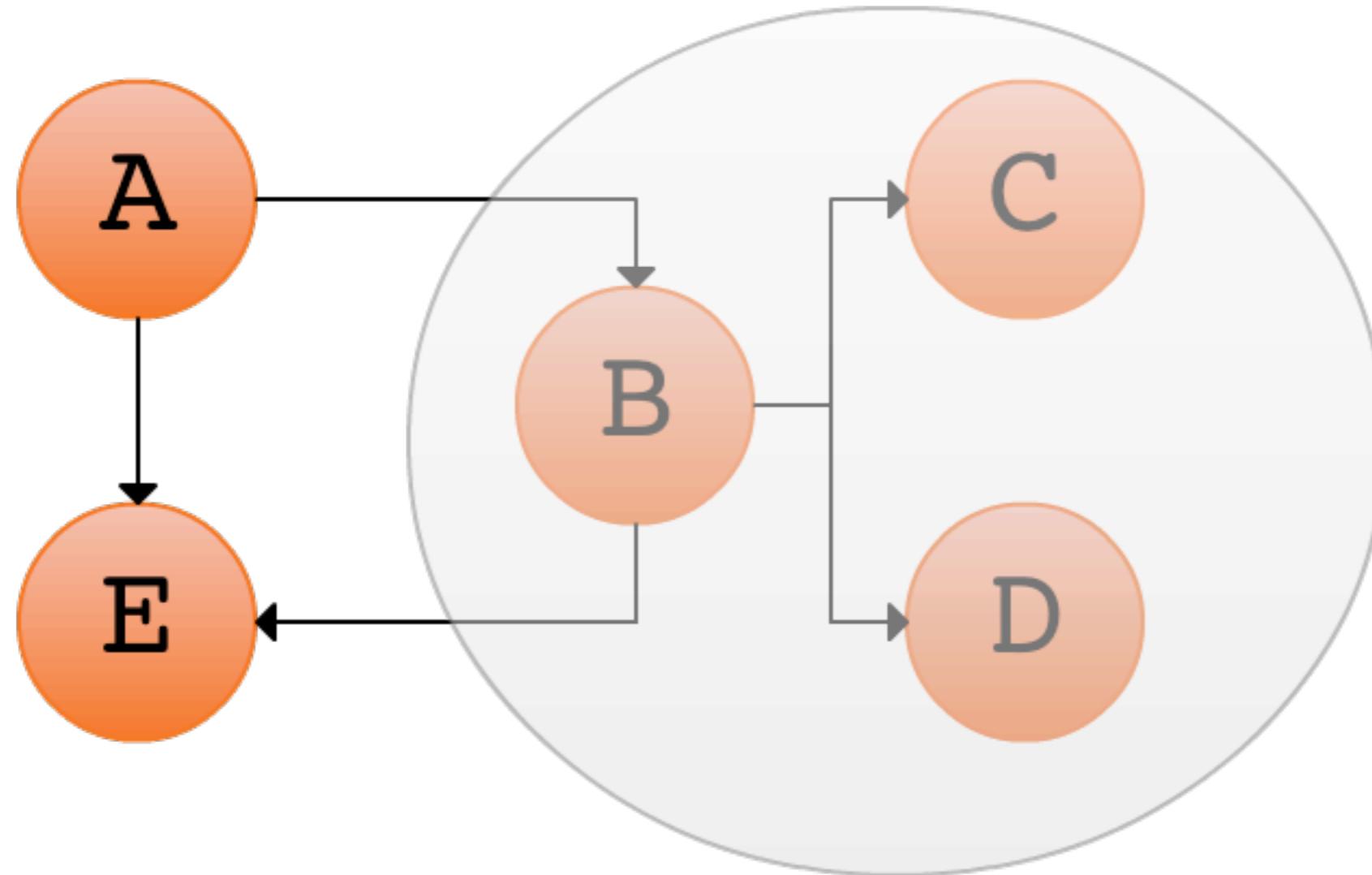
Performance

Labs

- Demo-Excluding Tasks



Smart Exclusion



```
>gradle A -x B
```

Task

Input/Output

Labs

- ⚡ Demo-Incremental Build



**Should clean be
required for a
reliable build?
(Hint: We have the
21st century)**

Task Input/Output

- ▶ You can describe:
 - ▶ Input/Output Files
 - ▶ Input/Output Dirs
 - ▶ Input Properties
- ▶ Gradle's build-in tasks all describe their input/output.

Incremental Build

- ▶ The hashes of the input/output files are cached.
- ▶ The hashes for all files of the input dirs are cached.
- ▶ The property values are cached (serialized).
- ▶ Cache == Current -> Skip Task

Annotations

```
class MyTask extends DefaultTask {  
    @InputFile File text  
    @InputFiles FileCollection path  
    @InputDirectory File templates  
    @Input String mode  
    @OutputFile File result  
    @OutputDirectory transformedTemplates  
    File someProp // ignored  
  
    @TaskAction  
    generate() { ... }  
}
```

Input/Output API

```
ant.import 'build.xml'
someAntTarget {
    inputs.files 'template.tm', new File('data.txt')
    inputs.dir 'someDir'
    outputs.files 'output.txt'
    outputs.dir 'generatedFilesDir'
    outputs.upToDateWhen { task ->
        dbDataUpToDate(task.dbUrl)
    }
}
```

Property Processing

- ▶ Exception if input files/dir do not exists
 - ▶ Disable validation with `@Optional`
- ▶ Output dirs are created before execution.

Performance

- ▶ Incremental Build
- ▶ Parallel Testing
- ▶ Soon: Parallel Builds, Distributed testing/builds
- ▶ Rich Model

Integration

Ant

Ant

- ▶ Ant is Gradle's friend not its competitor.
- ▶ Gradle uses Ant task's internally.
- ▶ You can use any Ant task from Gradle.
- ▶ Ant tasks are an integral part of Gradle.
- ▶ Gradle ships with Ant.
- ▶ You can import any Ant build into Gradle

Ant Tasks

- ▶ Gradle provides an instance of the Groovy AntBuilder

```
ant.delete dir: 'someDir'  
ant {  
    ftp(server: "ftp.comp.org", userid: 'me', ...) {  
        fileset(dir: "htdocs/manual") {  
            include name: "**/*.html"  
        }  
        // high end  
        myFileTree.addToAntBuilder(ant, 'fileset')  
    }  
    mkdir dir: 'someDir'  
}
```

Importing Ant Builds

```
<project>
  <target name="hello" depends="intro">
    <echo>Hello, from Ant</echo>
  </target>
</project>
```

```
ant.importBuild 'build.xml'
hello.doFirst { println 'Here comes Ant' }
task intro << { println 'Hello, from Gradle' }
```

```
>gradle hello
Hello, from Gradle
Here comes Ant
[ant:echo] Hello, from Ant
```

Maven

Labs

• ⚡ Demo-Maven Import

Maven

- ▶ Retrieve/Deploy to Maven/Ivy repositories
- ▶ Autogeneration of pom.xml/ivy.xml
- ▶ Convert Maven build into build.gradle
- ▶ Import of Maven builds
 - ▶ Soon: Deep Import
 - ▶ Soon: Use Gradle from Maven

Ecosystem

- ▶ Deep Integration with Artifactory
- ▶ Nexus
- ▶ Jenkins/Hudson
- ▶ Teamcity
- ▶ Eclipse (via STS)
- ▶ Idea II
- ▶ Sonar

Migration

Build Migration

- ▶ Mission Critical!
- ▶ Nightmare if the new build system can't adapt to the existing project layout:
 - ▶ Freeze
 - ▶ Project automation not working for a while
 - ▶ Different branches (unreliable, hard to compare, ...)
- ▶ Gradle's suppleness enables baby steps.
 - ▶ Gradle can adapt to any project layout.
 - ▶ No separate branches
 - ▶ Comparable --> Write tests

Enterprise Dependency Cache

Wharf

- ▶ Contribution by Fred Simon, Chief Architect JFrog (Artifactory).
- ▶ Metadata cache per resolver (`url = id`)
- ▶ Global checksum cache for jars

Usecases

- ▶ Repository Change:
 - ▶ A new metadata cache is created
 - ▶ Check for Jar
 - ▶ If not there, Exception:
 - ▶ If checksum OK no download
 - ▶ No inconsistencies between cache and repository.
- ▶ Dynamic revisions are retrieved per repository.
- ▶ Local installs don't pollute other builds.

Benefits

- ▶ Local Cache is not hiding problems
- ▶ Local Cache is not creating special behaviour
- ▶ Better Reproducability.
- ▶ Transactional
- ▶ Unique offering by Gradle

Testing

Test Task

- ▶ Support for JUnit and TestNG
- ▶ Parallel Testing
- ▶ Custom Fork Frequency
- ▶ Remote Listeners
- ▶ Tests auto-detected in `sourceSets.test.classes`

Name	test
Type	Test
Input	<code>sourceSets.test.classes</code> <code>configurations.testRuntime</code>

Test Task Example

```
test {  
    jvmArgs: [ "-Xmx512M"]  
    include "**/tests/special/**/*Test.class"  
    exclude "**/old*Test.class"  
    forkEvery = 30  
    maxParallelForks = guessMaxForks()  
}  
  
def guessMaxForks() {  
    int processors =  
        Runtime.runtime.availableProcessors()  
    return Math.max(2, (int) (processors / 2))  
}
```

Disables Auto Detection

Test Task Listeners

```
test {  
    beforeTest { descr ->  
        // do something  
    }  
    afterTest { descr, result ->  
        // do something  
    }  
    afterSuite { descr, result ->  
        // do something  
    }  
}
```

Labs

-  Demo - Testing



Discoverability

Lifecycle Tasks

- ▶ The relevant tasks for a build user.
- ▶ Achieve a certain stage in the build lifecycle for a project.
 - ▶ clean
 - ▶ classes
 - ▶ test
 - ▶ assemble
 - ▶ check
 - ▶ build (depends on assemble and check)
- ▶ Hooks for worker tasks.

Labs

- Lab 19-Discoverability



Multiproject Builds

Multi-Project Builds

- ▶ Arbitrary Multiproject Layout
- ▶ Configuration Injection
- ▶ Project Dependencies & Partial builds
- ▶ Separate Config/Execution Hierarchy

Configuration Injection

► **ultimateApp**

- ▶ api
- ▶ webservice
- ▶ shared

```
subprojects {  
    apply plugin: 'java'  
    dependencies {  
        compile "commons-lang:commons-lang:3.1"  
        testCompile "junit:junit:4.4"  
    }  
    test {  
        jvmArgs: [ 'Xmx512M' ]  
    }  
}
```

Filtered Injection

▶ **ultimateApp**

- ▶ api
- ▶ webservice
- ▶ shared

```
configure(nonWebProjects()) {  
    jar.manifest.attributes  
        Implementor: 'Gradle-Inc'  
}  
  
def nonWebProjects() {  
    subprojects.findAll {project ->  
        !project.name.startsWith('web')  
    }  
}
```

Project Dependencies

- ▶ ultimateApp
 - ▶ **api**
 - ▶ webservice
 - ▶ shared

```
dependencies {  
    compile "commons-lang:commons-lang:3.1",  
    project( ':shared' )  
}
```

First Class Citizen

Partial Builds

- ▶ ultimateApp
 - ▶ **api**
 - ▶ webservice
 - ▶ shared

```
>gradle build  
>gradle buildDependents  
>gradle buildNeeded
```

There is
no one-size-fits-all
project structure
for the
enterprise.

The physical
structure of your
projects should
be determined by
**your
requirements.**

Defining a Multi Project Build

- ▶ `settings.gradle` (location defines root).
- ▶ root project is implicitly included

Defines a virtual hierarchy

By default maps to file path <root>/project1

```
include 'project1', 'project2', 'project2:child1'
```

Default to root dir name

```
// Everything is configurable
rootProject.name = 'main'
project(':project1').projectDir = '/myLocation'
project(':project1').buildFileName =
    'project1.gradle'
```

Default to build.gradle

Labs

- Lab 20-Multi-Project Build

Wrapper

Wrapper Task

- ▶ Wrapper task generates:
 - ▶ wrapper scripts
 - ▶ wrapper jar
 - ▶ wrapper properties.

```
task wrapper(type: Wrapper) {  
    gradleVersion = '0.6'  
    jarPath = 'gradle'  
}
```

Wrapper Files

Name
build.gradle
▼ gradle
gradle-wrapper.jar
gradle-wrapper.properties
gradlew
gradlew.bat
► src

```
> ./gradlew assemble
```

C/C++

Labs

-  Demo - C/C++

Sonar

Labs

-  Demo - Sonar



Series of UI Products

- ▶ Commercial Web Apps for the Enterprise
- ▶ First Product: Discovery (Working Title)
- ▶ Build Optimization
 - ▶ Metrics for CI and Dev builds
 - ▶ Get the average build time down
 - ▶ Get as much feedback before the commit

Q & A