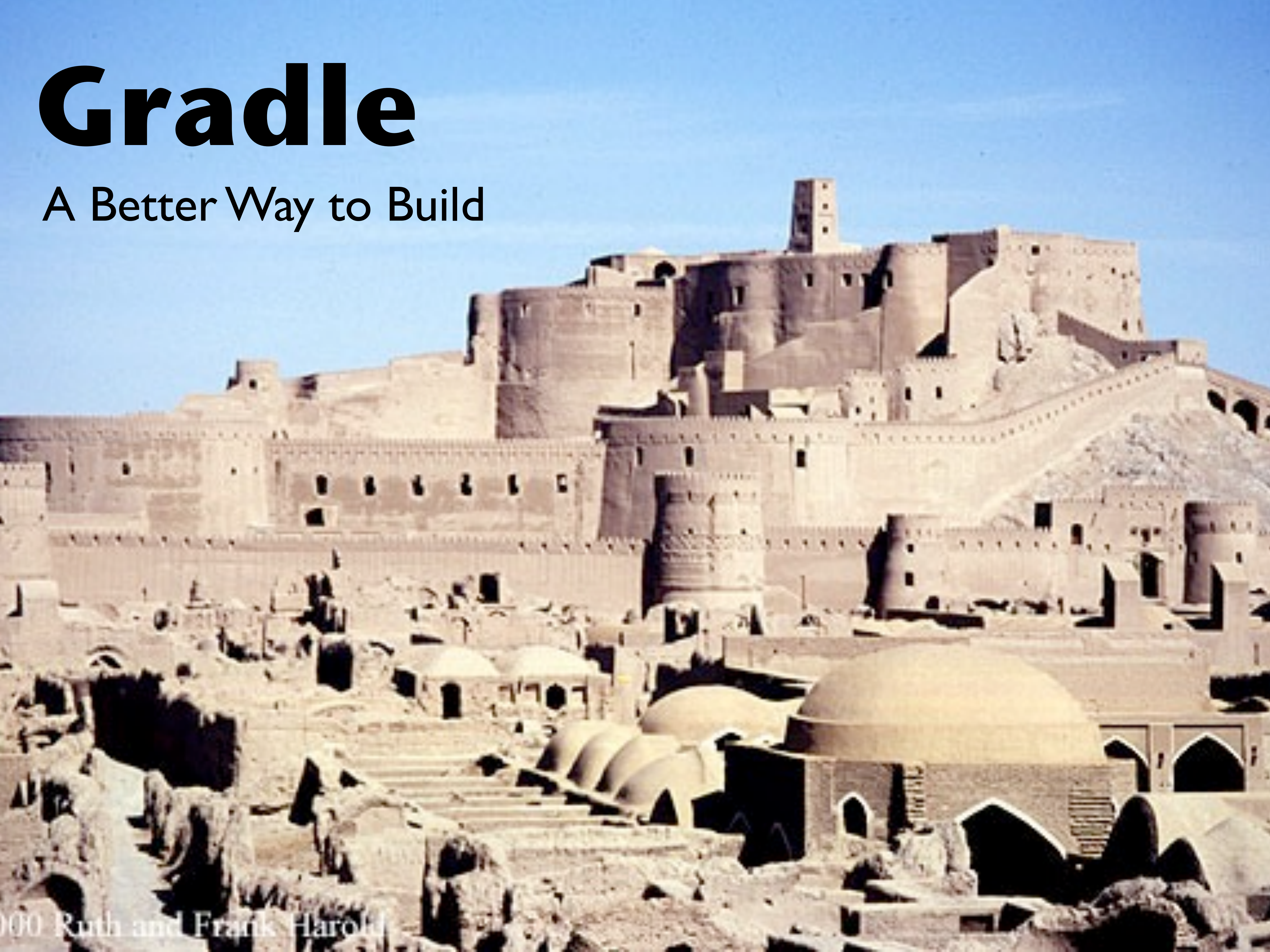
Gradle

A Better Way to Build



000 Ruth and Frank Harold

What you will learn

- Gradle basics
- Working with tasks
- Working with plugins
- Ant integration
- Dependency management
- Testing
- Multi-project builds
- Gradle Wrapper

Intro

What is Gradle

- A general purpose build system
- Groovy DSL with a Java core
- Provides built-in support for Java, Groovy, Scala, Web, OSGi
- Exciting solutions for many of the big pain points you often have with current build systems

Gradle Project Background

- Very active community (mailing list, patches, issues)
- Apache v2 license
- Excellent user guide (300+ pages) and many samples
- Frequent releases, multiple commits per day
- Quality is king:
  - 6000 unit, integration, and acceptance tests
  - Healthy codebase
  - Low defect rate
- Committers: Steve Appling, Hans Dockter, Tom Eyckmans, Adam Murdoch, Russel Winder, Peter Niederwieser, Szczepan Faber, Luke Daley, Darrel DeBoer

Labs

- Lab 01 - Setup

Labs

- Lab 02 - Quickstart

Labs

- Lab 03 - Incremental Build

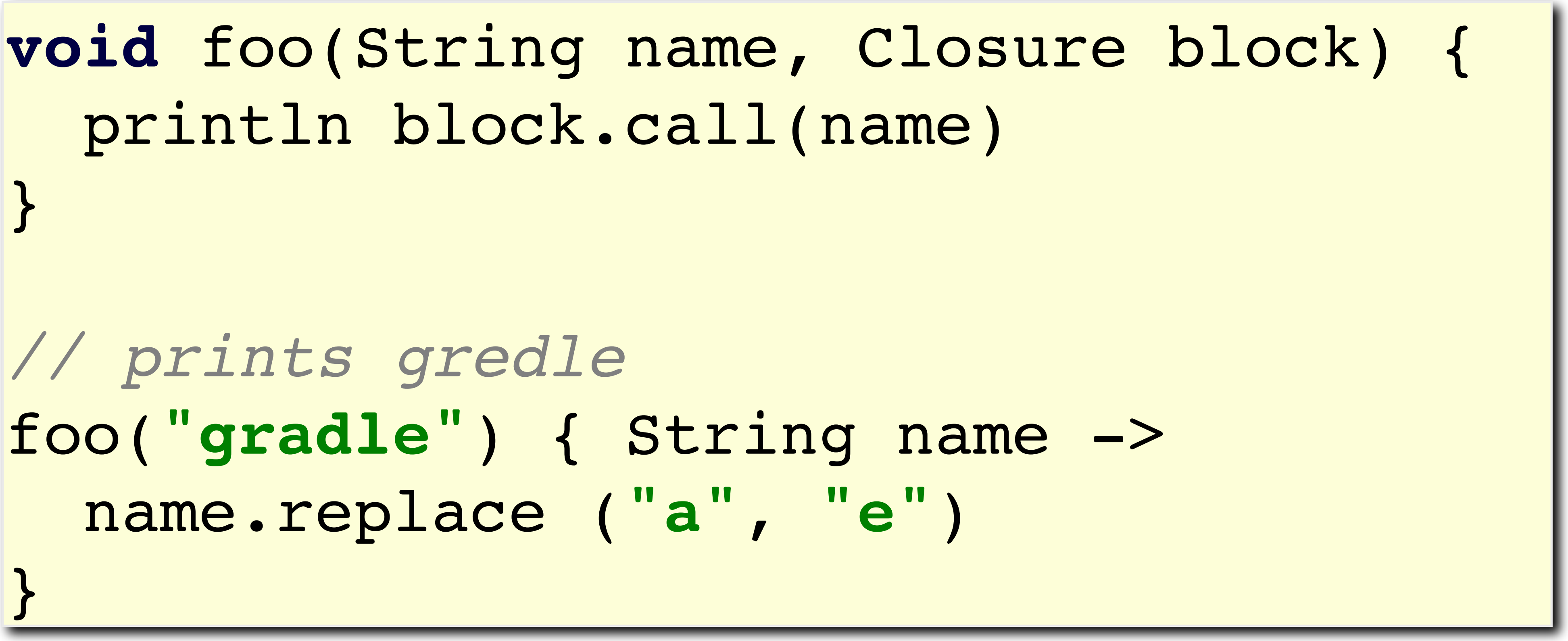
Build Script Basics

Groovy

- A Ruby or Python like language that is tightly integrated with the Java platform
- Compiles to byte code
- Design goal is to be easily picked up by Java developers
- Reuse of Java semantics and API

Groovy Closures

- Closures are code blocks that can act like data
- Cf. lambdas, function pointers, anonymous inner classes
- The Gradle DSL uses them extensively
- The Groovy API uses them extensively



```
void foo(String name, Closure block) {
  println block.call(name)
}

// prints gredle
foo("gradle") { String name ->
  name.replace ("a", "e")
}
```

Groovy Collection Operations

```
[1, [2,3]].flatten() // [1, 2, 3]
['a', 'b'].each { item -> println item }
['a', 'b'].collect { it + '1' } // ['a1', 'b1']
['a', 'b', 'c'].findAll { it != 'c' } // ['a', 'b']
[1, 2, 3].every { it < 3 } // false
[1, 2, 3].any { it < 3 } // true
// many more
```

- Thanks to Groovy, Gradle's API can stay light-weight
- Learning Groovy has many benefits. It is a powerful tool for many purposes (e.g. testing)
- The book Groovy in Action (2nd Ed) is the standard reference for Groovy

Gradle Build Scripts

- Must be compilable by Groovy
- Can’t be executed by plain Groovy runtime
- Delegate to an associated org.gradle.api.Project object

```
// does not compile
println 'Gradle

// compiles, fails when run with plain Groovy
println name

// compiles, fails when run with Groovy or Gradle
println zipCode
```

Gradle Build Scripts

- **Configure** the Project object
- Do **not** execute the build

Tasks

Tasks

- Tasks are the basic unit of work in Gradle
- Tasks have a list of actions to be executed

```
task someTask {
  doFirst { ... }
  doLast { ... }
}

someTask {
  doFirst { ... }
}
```

Labs

- Lab 04 - Tasks

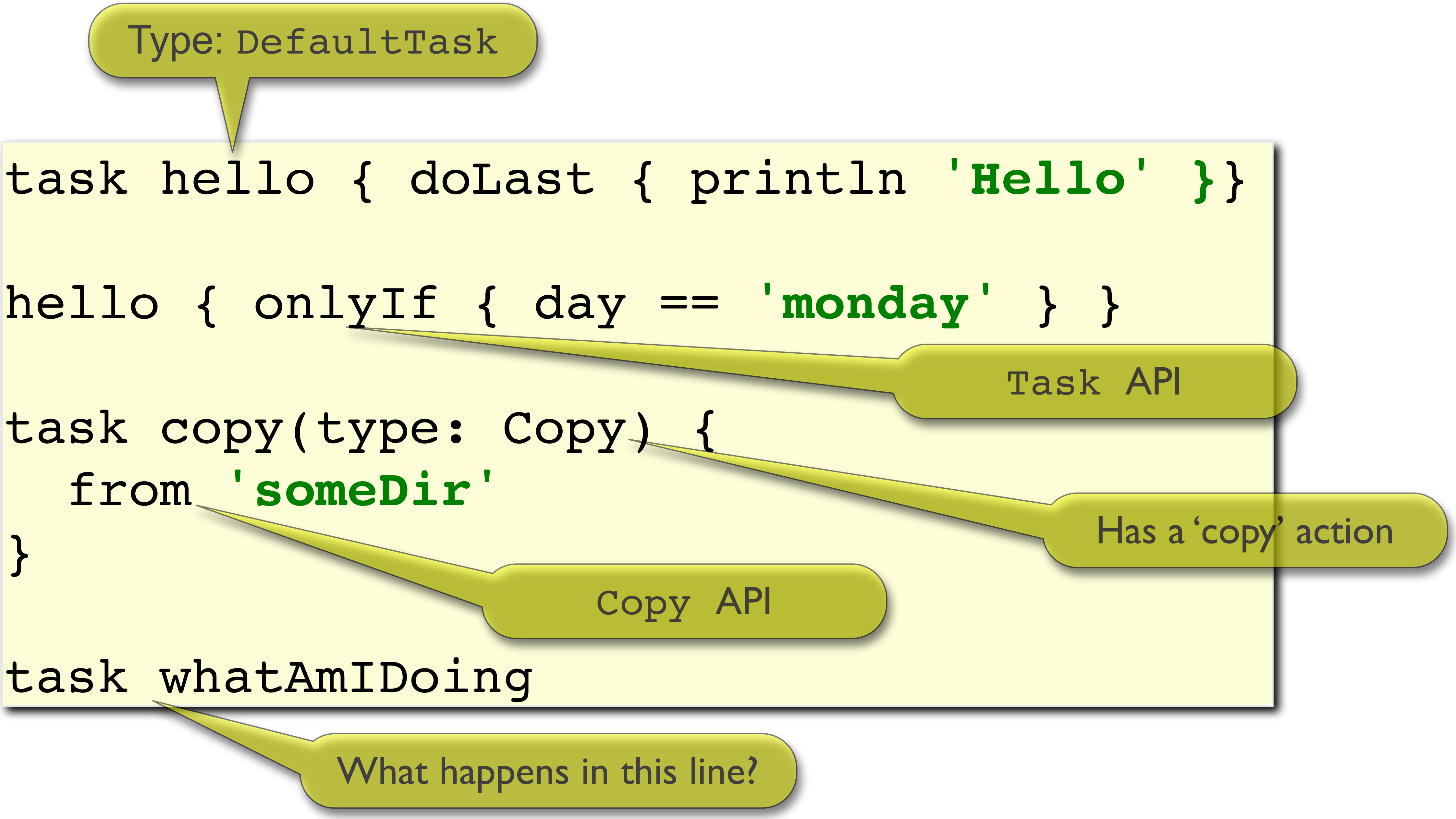
DSL Syntax And Tasks

```
task hello

hello {
  dependsOn otherTask
  onlyIf { day == 'monday' }
  doLast { println 'Hello' }
}

task hello {
  dependsOn otherTask
  onlyIf { day == 'monday' }
  doLast { println 'Hello' }
}
```

Task Types and API

- Tasks have a type and API
- If not specified, type is DefaultTask
- All tasks implement the Task interface
- Many built-in task types
- Most task types already have an action

Task Types and API



```
task hello { doLast { println 'Hello' }}

hello { onlyIf { day == 'monday' } }

task copy(type: Copy) {
   from 'someDir'
}

task whatAmIDoing
```

Type: DefaultTask

Task API

Has a ‘copy’ action

Copy API

What happens in this line?

Custom Task Types

- Extend DefaultTask
- Declare action with @org.gradle.api.tasks.TaskAction

```
class FtpTask extends DefaultTask {
  String host = 'docs.mycompany.com'

  @TaskAction
  void ftp() {
    println     // do something complicated
  }


}
```

Labs

- Lab 05 - Custom Tasks

Task Dependencies

- Tasks can depend on each other
- Execution of one task requires prior execution of another task
- Executed tasks form a directed acyclic graph

```
task foo

task bar { dependsOn foo, baz }

bar { dependsOn baz }

// What happens here?
task bar { doLast { dependsOn foo } }
```

Labs

- Lab 06 - Task Dependencies

Plugins

Plugins

- Two flavors
  - Script plugin: Another (local or remote) build script
  - Binary plugin: A class implementing org.gradle.api.Plugin

Applying Plugins

- Any Gradle script can act as a plugin
- Binary plugins must be on the build script class path
  - Can have ID (mapped to class name via plugin descriptor)
  - Will learn later how to add elements to the build script class path
  - The built-in plugins are already on the build script class path

```
apply from: 'otherScript.gradle'
apply from: 'http://mycomp.com/otherScript.gradle'
```

```
apply plugin: org.gradle.api.plugins.JavaPlugin
apply plugin: 'java'
```

What Plugins Can Do

- Configure the project object (e.g. add task instances)
- Add other classes to class path (e.g. custom task types)
- Add properties and methods to project object (extend DSL)
- Build script decomposition
  - Separate imperative from declarative
  - Modularization
- Code reuse

Standard Gradle Plugins

Plugin ID	applies
base	
java	java-base -> base
groovy	groovy-base -> java-base
scala	scala-base -> java-base
application	java
war	java
jetty	war
ear	base
osgi	java-base
antlr	java
code-quality	reporting-base
maven	base
eclipse	
idea	
announce	
sonar	
signing	base
cpp-lib	cpp
cpp-exe	cpp

Labs

- Lab 07 - Applying Plugins

Labs

- Lab 08 - Testing

Ant

Ant

- Ant is Gradle's friend not its competitor
- Gradle uses Ant tasks 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 Groovy’s AntBuilder

```
ant {
  delete dir: 'someDir'
  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>
```

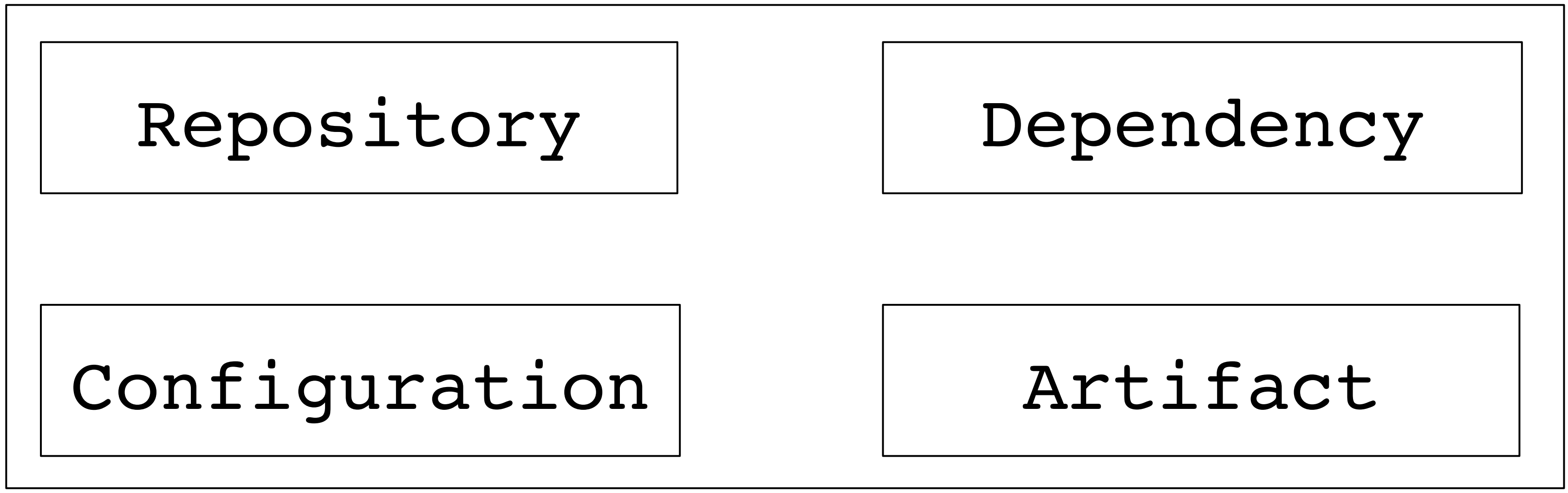
```
ant.importBuild 'build.xml'
hello.doFirst { println 'Here comes Ant' }
task intro << { println 'Hello, from Gradle'}
```

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

Dependencies

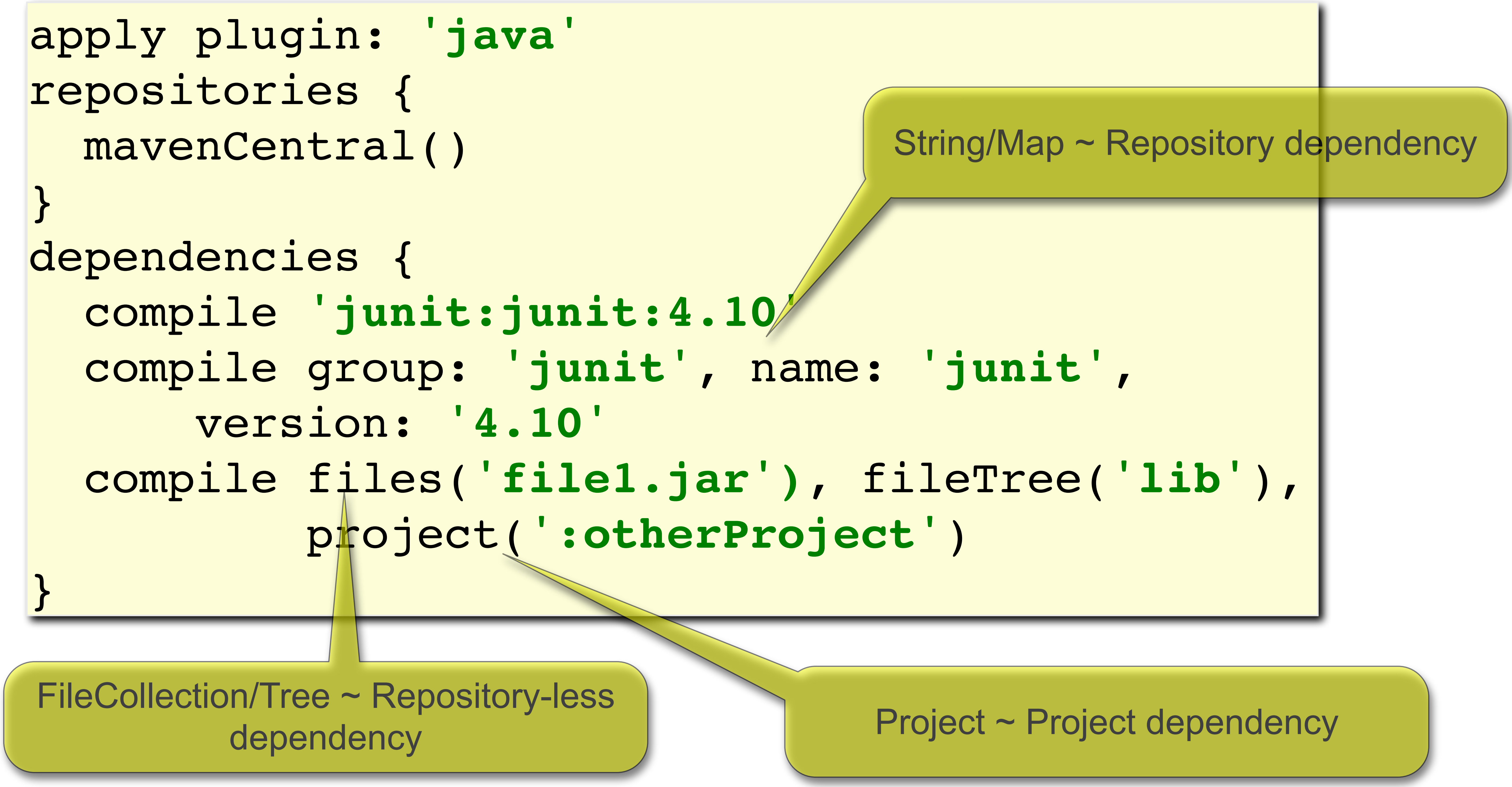
Dependencies

- Repository dependencies
  - e.g. from Maven Central
  - with module descriptors (pom.xml/ivy.xml)
- Repository-less dependencies (specified by path)
- Project dependencies in a multi-project build
- Artifacts you want to upload



The domain objects

Dependencies



```
apply plugin: 'java'
repositories {
  mavenCentral()
}
dependencies {
  compile 'junit:junit:4.10'
  compile group: 'junit', name: 'junit',
      version: '4.10'
  compile files('file1.jar'), fileTree('lib'),
          project(':otherProject')
}
```

String/Map ~ Repository dependency

FileCollection/Tree ~ Repository-less dependency

Project ~ Project dependency

Dependencies & Java Plugin

```
apply plugin: 'java'
configurations { myConf.extendsFrom compile }
dependencies {
  compile 'junit:junit:4.10'
  runtime group:'asm', name:'asm-all', version:'3.2'
  testCompile files('file1.jar')
  myConf 'log4j:log4j:1.2.9'
}
```

- The Java plugin adds configurations
- Many Java plugin tasks use those configurations as default input values (e.g. test)
- Configurations can extend each other

Working with Dependencies

- A configuration extends `FileCollection`
- Configuration has a rich API

```
configurations.runtime.each { file ->
   println file
}

configurations.runtime.dependencies { dep ->
  dep.group == 'org.gradle'
}.each { println it }

copy {
  from configurations.runtime
  into 'ideLib'
}
```

Labs

- Lab 09 - Dependencies

Transitive Dependency Mgmt

- Supported for repository dependencies
- pom.xml/ivy.xml describes transitive dependencies
- Default version conflict resolution is newest
- Transitive resolution is customizable

```
dependencies {
  compile('org.hibernate:hibernate:3.1') {
    force = true
    exclude module: 'cglib'
  }
  compile('org:somename:1.0') {
    transitive = false
  }
}
configurations.myconf.transitive = false
```

Repositories

- Any Maven/Ivy repository can be accessed
- Very flexible layouts are possible for non Maven repositories

```
repositories {
  mavenLocal()
  mavenCentral()
  maven {
    name 'codehaus'
    url 'http://repository.codehaus.org'
  }
  ivy {
    url 'http://repo.mycompany.com'
    layout 'gradle' // default
  }
  flatDir(dirs: ['dir1', 'dir2'])
}
```

Multiproject Builds

Multi-Project Builds

- Arbitrary directory layout
- Configuration injection
- Project dependencies & partial builds
- Separate configuration/execution hierarchy

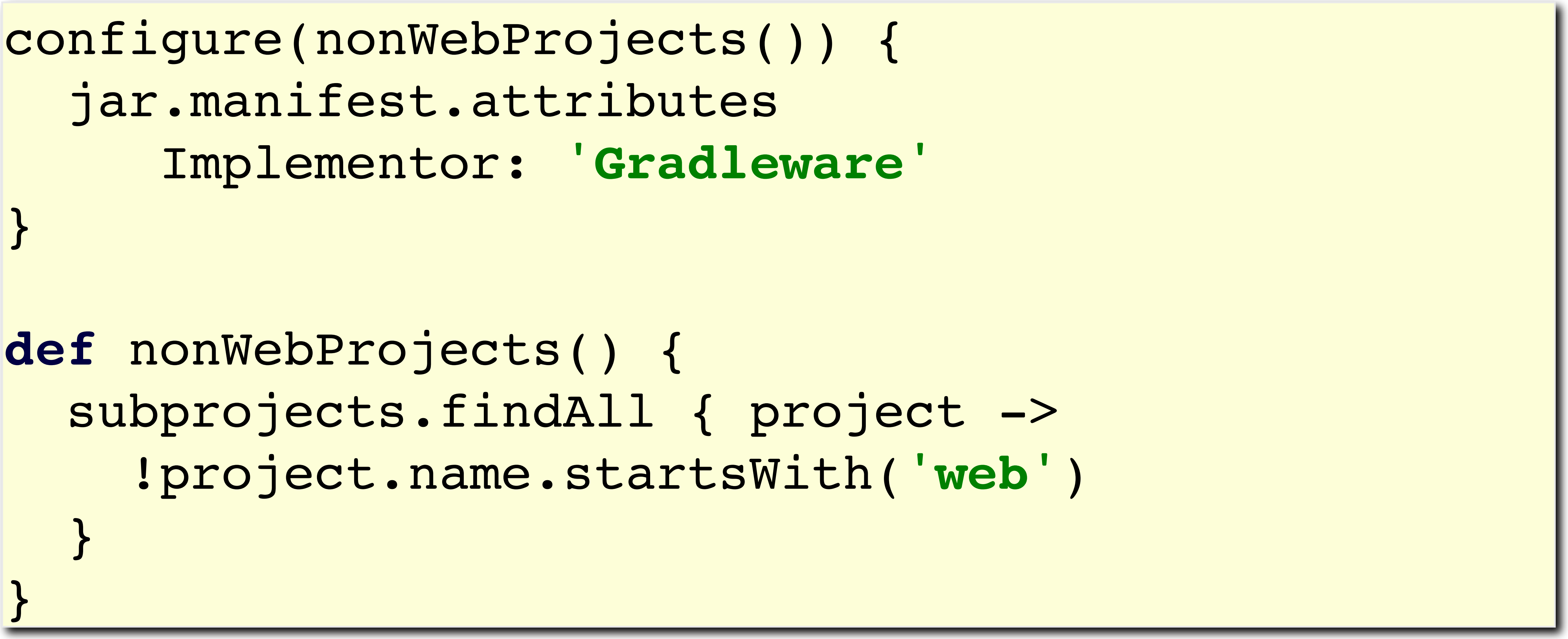
Configuration Injection

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

```
subprojects {
    apply plugin: 'java'
    dependencies {
        testCompile 'junit:junit:4.7'
    }
    test {
        jvmArgs '-Xmx512M'
    }
}
```

Filtered Injection

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



```
configure(nonWebProjects()) {
  jar.manifest.attributes
     Implementor: 'Gradleware'
}

def nonWebProjects() {
  subprojects.findAll { project ->
    !project.name.startsWith('web')
  }
}
```

Project Dependencies

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

```
dependencies {
  compile 'commons-lang:commons-lang:2.4'
  compile 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**

Name Matching Execution

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

```
>gradle build
>gradle classes
>gradle war
```

Task/Project Paths

- For projects and tasks there is a fully qualified path notation:
  - `:` (root project)
  - `:clean` (the clean task of the root project)
  - `:api` (the api project)
  - `:services:webservice` (the webservice project)
  - `:services:webservice:clean` (the clean task of webservice)

```
>gradle :api:classes
```

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 'api','shared','services:webservice'

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

Defaults to root dir name

Defaults to build.gradle

Labs

- Lab 10 - Multi-Project Builds

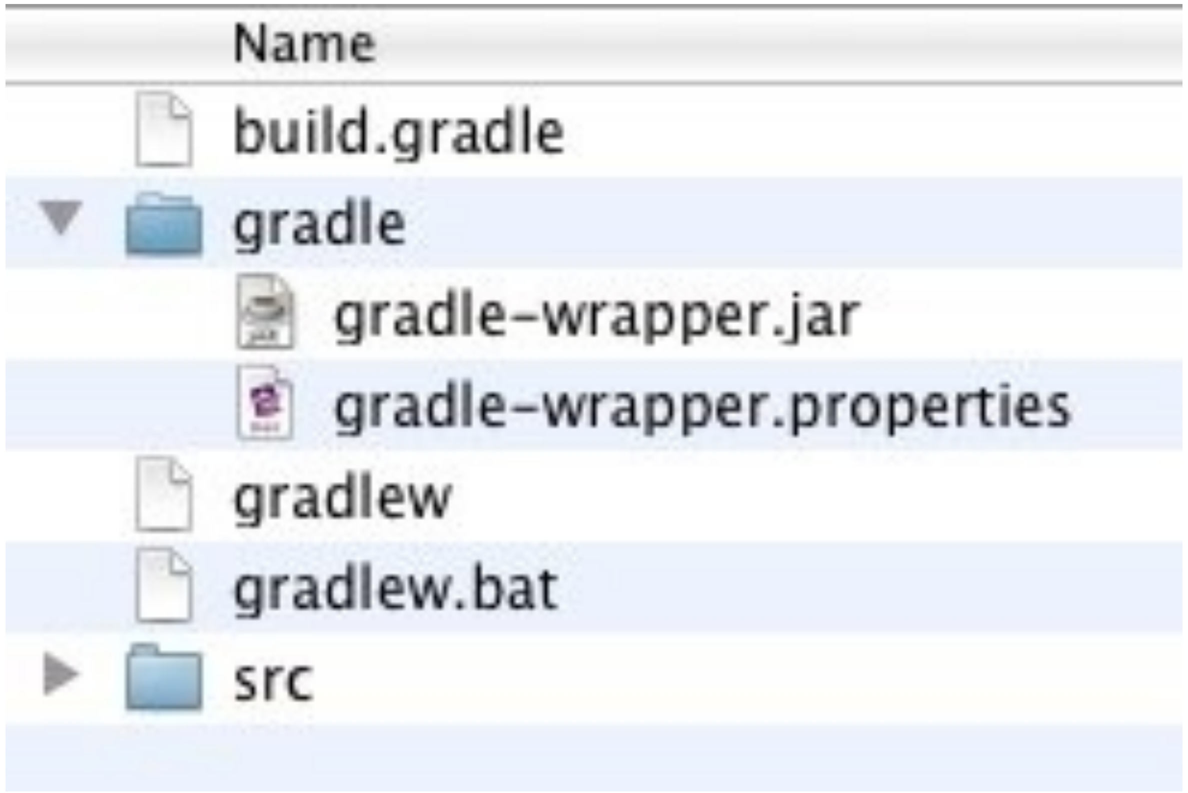
Wrapper

Wrapper Task

- Wrapper task generates:
  - wrapper scripts
  - wrapper Jar
  - wrapper properties

```
task wrapper(type: Wrapper) {
  gradleVersion = '1.0-rc-3'
}
```

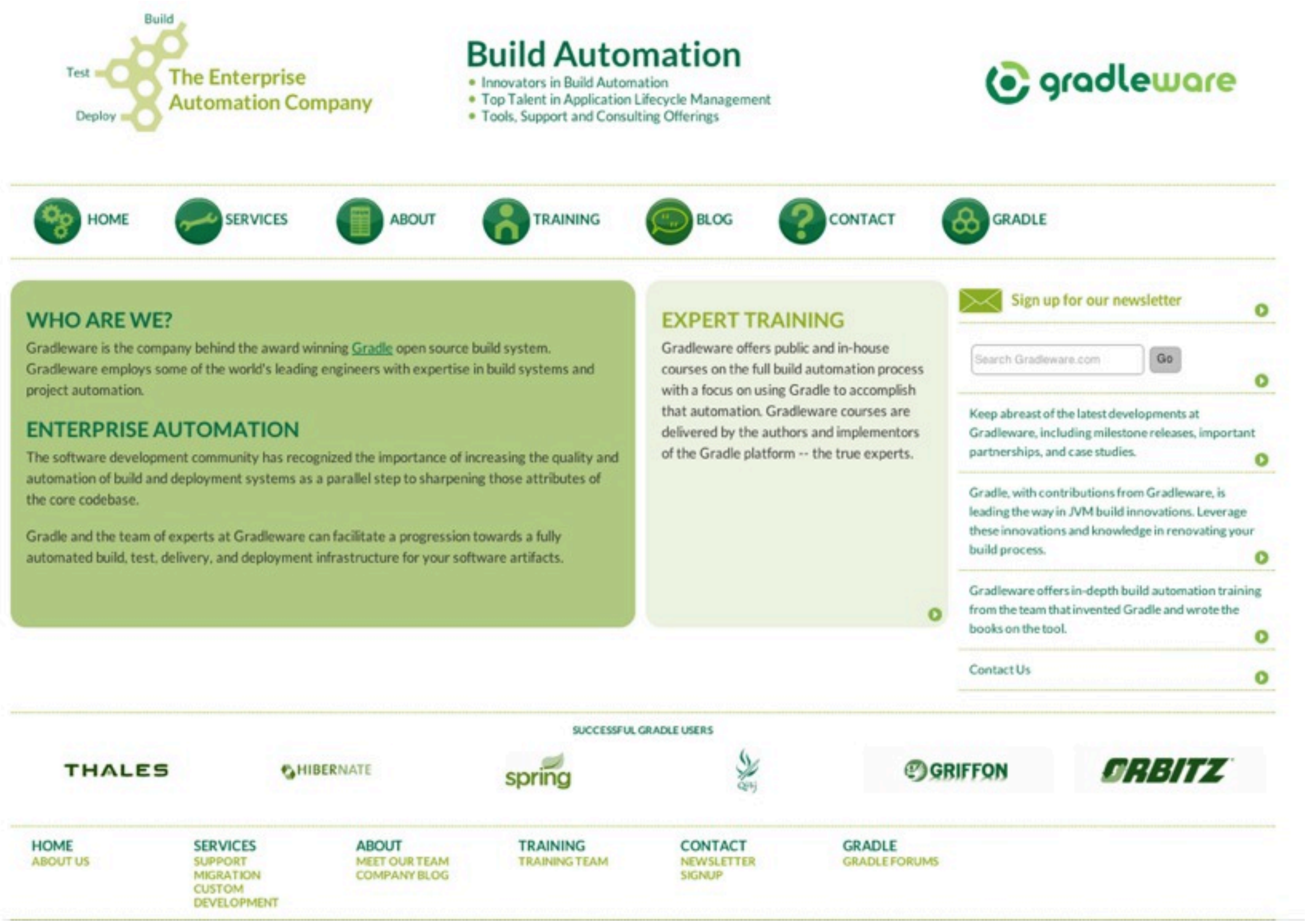
Wrapper Files



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

```
>./gradlew build
```

Commercial Support: *gradleware.com*



Build Automation

- Innovators in Build Automation
- Top Talent in Application Lifecycle Management
- Tools, Support and Consulting Offerings

WHO ARE WE?

Gradleware is the company behind the award winning Gradle open source build system. Gradleware employs some of the world's leading engineers with expertise in build systems and project automation.

ENTERPRISE AUTOMATION

The software development community has recognized the importance of increasing the quality and automation of build and deployment systems as a parallel step to sharpening those attributes of the core codebase.

Gradle and the team of experts at Gradleware can facilitate a progression towards a fully automated build, test, delivery, and deployment infrastructure for your software artifacts.

EXPERT TRAINING

Gradleware offers public and in-house courses on the full build automation process with a focus on using Gradle to accomplish that automation. Gradleware courses are delivered by the authors and implementors of the Gradle platform -- the true experts.

Keep abreast of the latest developments at Gradleware, including milestone releases, important partnerships, and case studies.

Gradle, with contributions from Gradleware, is leading the way in JVM build innovations. Leverage these innovations and knowledge in renovating your build process.

Gradleware offers in-depth build automation training from the team that invented Gradle and wrote the books on the tool.

Contact Us