ProTech *"Charting the Course ...*

protechtraining.com

... to Your Success!"

Gradle In-Depth for Native C Applications

Course Summary

Description

The Gradle In-Depth training course is an extensive hands-on three day Gradle training course for build masters and developers who are authoring their builds. Participants will learn how to use the Gradle build system to substantially increase their productivity.

This Gradle training class first runs through some Groovy fundamentals and then moves quickly into thorough in-depth coverage of the Gradle build system. This course puts additional emphasis and focus on building Native C applications and the Gradle C/C++ plugin.

Participants will spend a significant portion of the class with their hands on the keyboard, learning by doing, working through lab exercises designed to build on the presented material and explore the depths of Gradle. This class can take you from complete newcomer status to Gradle competence. Due to the in-depth treatment of the topics it is also very suitable for people who have experience working with Gradle. The class will be delivered by a Gradle Core Developer which gives you access to the deepest Gradle knowledge available. No question will remain unanswered.

Objectives

By the end of this course, students will be able to:

- Learn the basics of Gradle and Groovy
- Write your own Gradle plugins
- Learn about copy and archive tasks in Gradle
- Working with a powerful file system API
- Build migration best practices
- How to use the Gradle performance
- Learn the ins and outs of the C/C++ plugin
- Learn how to build C/C++ applications
- How to efficiently move away from your current build environment
- Learn how to deal with complex build requirements
- Learn how to deal with large multi module builds

Topics

- The Gradle Project
- The Groovy Language
- Gradle Introduction
- Tasks
- Logging
- Plugins
- Working with Files
- Extending the Model
- Task Inputs and Outputs
- The C/C++ Plugin
- Build Variants
- Multiproject Builds
- The Build Runtime

Audience

This course is designed for native developers and build masters who want to build C/C++ applications with Gradle. New and experienced users are encouraged to attend the Gradle In-depth training. This class has something for every level of Gradle user.

Prerequisites

Exposure to building C and/or C++ applications is assumed. Some code is initially easier to understand if you also have a basic understanding of the Groovy language but not a prerequisite. There will be a very short introduction to Groovy at the beginning of the course.

Duration

Three days

ProTech "Charting the Course ...
protechtraining.com
... to Your Success!"

Gradle In-Depth for Native C Applications

Course Outline

I. The Gradle Project
- A. About the project
- B. Documentation / Resources / Support

II. The Groovy Language
- A. Executing Groovy code
- B. Syntax basics
- C. Dynamic typing
- D. Domain Specific Languages & Gradle
- E. Groovy JDK extensions
- F. Closures

III. Gradle Introduction
- A. The Gradle philosophy
- B. Installing Gradle
- C. Gradle build scripts
- D. The build lifecycle
- E. The Gradle build daemon

IV. Tasks
- A. Declaring tasks
- B. Executing tasks
- C. Configuring tasks
- D. Implementing custom tasks
- E. Extending tasks
- F. Task dependencies
- G. Task exclusion
- H. Conditionally skipping tasks
- I. The Task Graph API
- J. Task rules

V. Logging
- A. Controlling log levels
- B. Logging from the build script
- C. Logging from classes
- D. Managing child process output

VI. Plugins
- A. What are Gradle plugins?
- B. The role of plugins
- C. Packaging and distributing plugins
- D. Review of core plugins

VII. Working with files
- A. The file tasks (e.g. Copy, Sync)
- B. Filtering, excluding, aggregating, renaming
- C. Working with CopySpecs
- D. Expanding and making archives (zip, tar etc.)
- E. The Gradle file APIs
- F. Best practices for working with files

VIII. Extending the model
- A. Extending existing objects
- B. Object extensions
- C. Object containers & configuration rules

IX. Task inputs and outputs
- A. The power of the input/output model
- B. Describing inputs and outputs
- C. Incremental builds
- D. Inferring task dependencies

X. The C/C++ plugin
- A. Executables and Libraries
  1. Project Conventions
  2. Source Sets
  3. Customizing Compiler Options
  4. Static and Shared Libraries
  5. Controlling Linkages
  6. Using Prebuilt Libraries
- B. Language Support
  1. Build Toolchains
  2. Building with C++
  3. Building with C
  4. Building with Assembler
  5. Building Windows Resources
  6. Polyglot Projects
  7. Visual Studio Support
- C. Build Variants
  1. Build Types
  2. Build Platforms
  3. Build Flavors
  4. Managing Variant Complexity

XI. Multiproject builds
- A. Defining a multiproject build
- B. Avoiding duplicate configuration with configuration injection
- C. Inter project dependencies
- D. Downstream and upstream partial builds

XII. The build runtime
- A. Extending all builds with init scripts
- B. Init script locations
- C. Using the Gradle wrapper
- D. Customizing the Gradle distribution with init scripts