SOEN 6431 Software Maintenance and Program Comprehension

Dependency Management and Continuous Integration

Concordia University

Dr. Juergen Rilling
Professor
Computer Science & Software Engineering

Tel **514-848-2424 ext. 3016**
Email **juergen.rilling@concordia.ca**
1455 De Maisonneuve Blvd. West, EV3.211
Montreal, Quebec, Canada H3G 1M8
www.rilling.ca

Maven

What is Maven? A Java build tool based on ANT

Advantages of using Maven:

- Facilitates build process
- Provides a uniform build system
- Simplifies dependency management
- breaks down the build lifecycle into phases
- plugins can be injected in any phase
- provides guidelines for development best practices

Disadvantages of Maven:

- Small learning curve

Introduction

- Java projects originally managed with **Ant**
  - Manual build process
  - **Ant** scripts to compile, generate IDL, jar
  - Low-level scripting
  - No dependency management
  - No version management

- **Maven** provides high-level features for build process

Introduction

- **What is Maven?**
  "Maven is a software management and comprehension tool based on the concept of Project Object Model (POM) which can manage project build, reporting, and documentation from a central piece of information"
- **What is POM?**

  "As a fundamental unit of work in Maven, POM is an XML file that contains information about project and configuration details used by Maven to build the project"

Objectives and Characteristics of MAVEN

Maven is more than just Build Tool

Maven was built considering certain objectives

Maven Provides:

- Easy Build Process
- Uniform Build System
- Quality Project Information
- Guidelines for Best Practices Development

Achieved Characteristics:

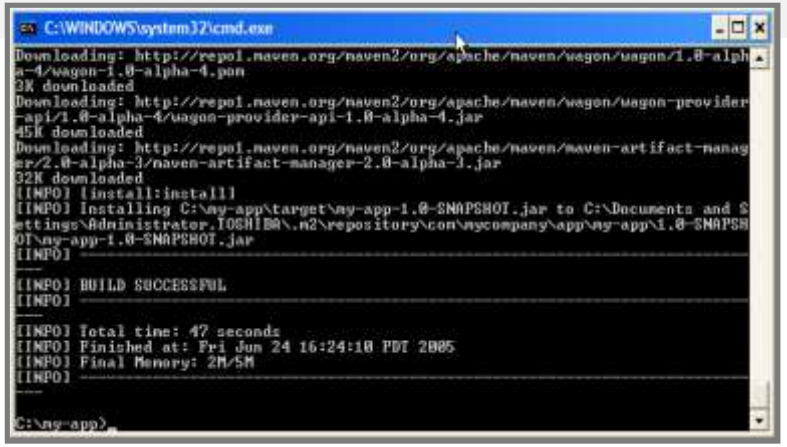
- Visibility
- Reusability
- Maintainability
- Comprehensibility “Accumulator of Knowledge”

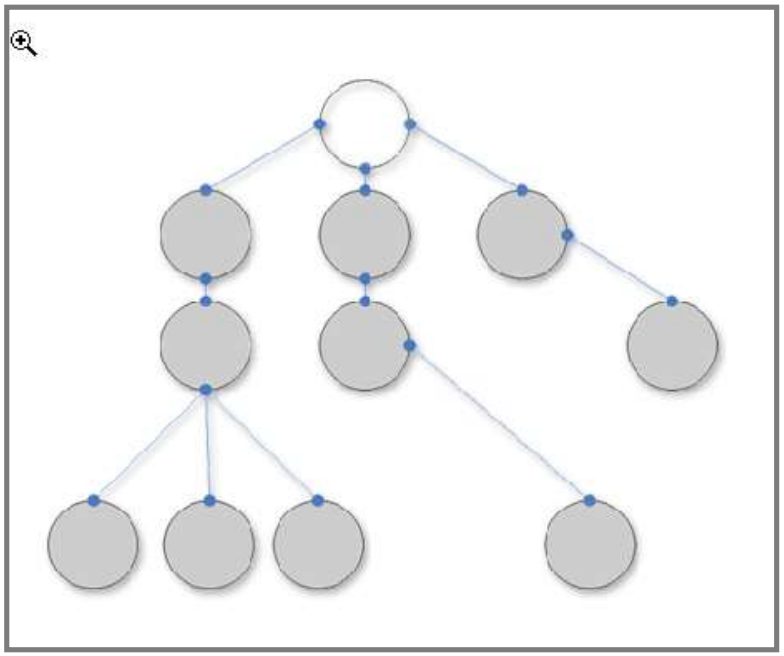
Provides a uniform build system

- Build process is external to IDEs
    - Guarantees that everyone can build it
- What about IDE debug mode?
    - IDE plugins exists for many platforms to allow this
- Build configuration is specified in pom.xml file
    - version information
    - classpath settings
    - dependencies
    - plugin data

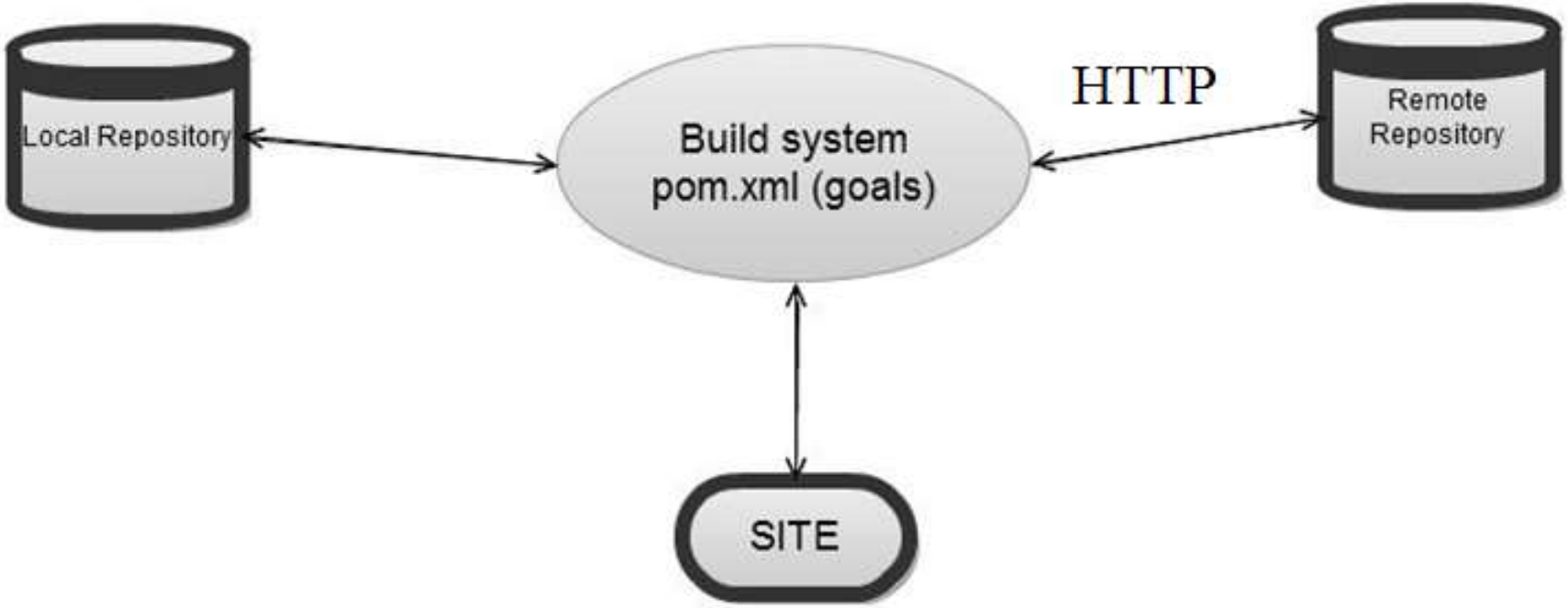
Main Features of MAVEN

- Build-Tool
- Dependency Management Tool
- Documentation Tool





Overview of Simple Architecture



Build Lifecycles

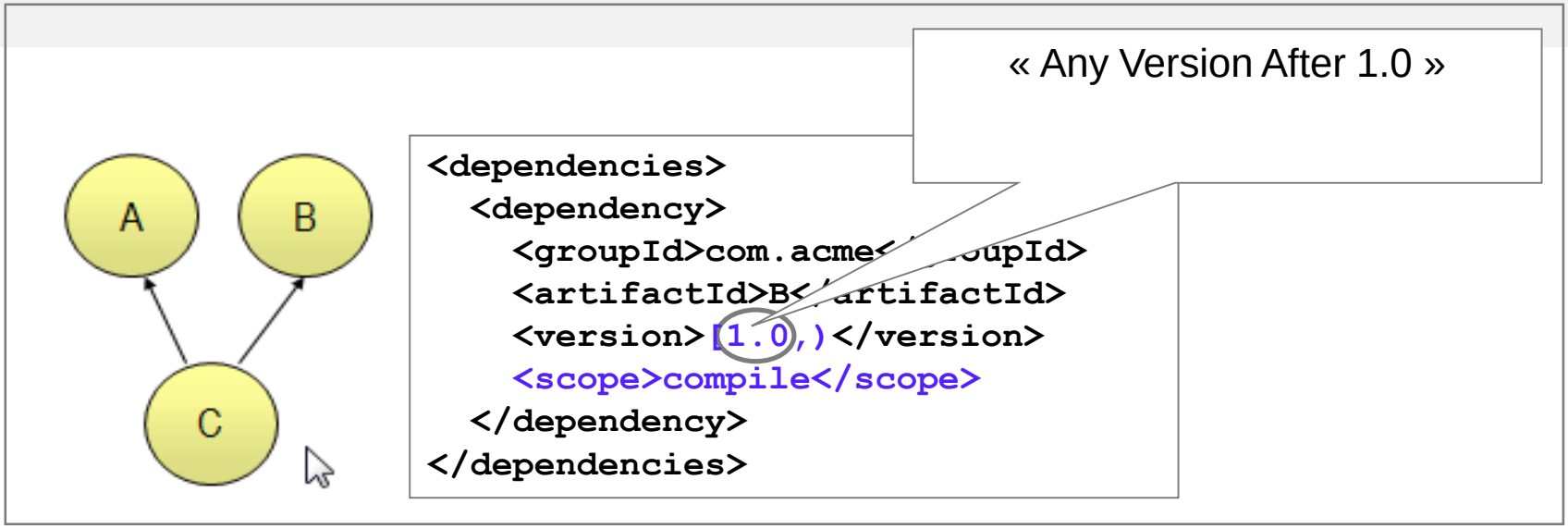
Made up of Phases. Some of the common ones are:

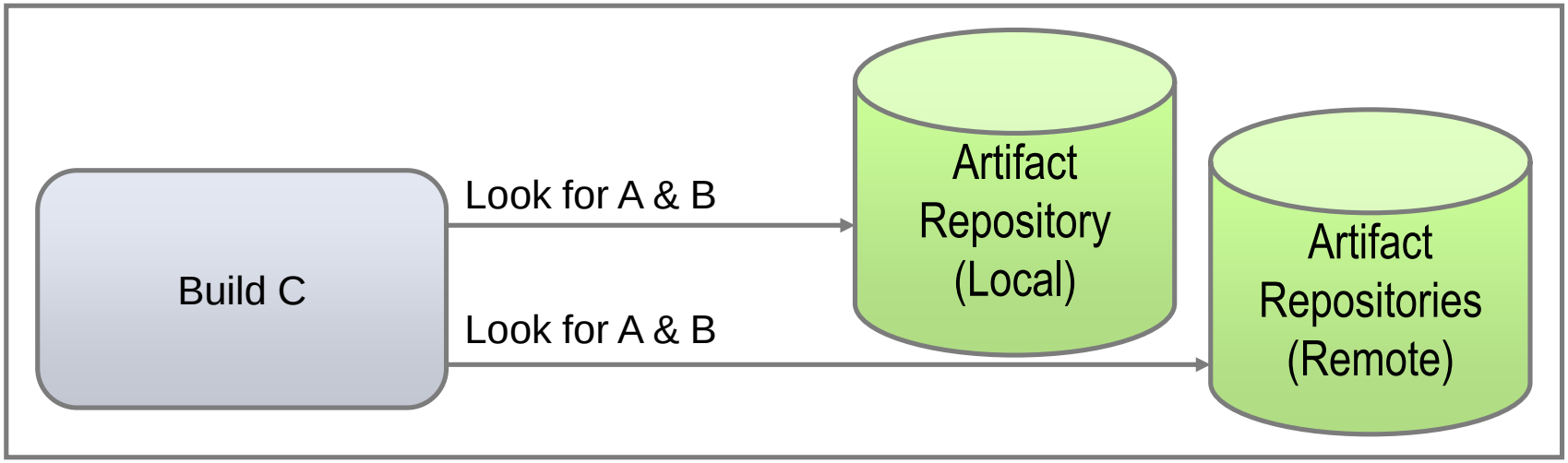
- validate - validate the project is correct and all necessary information is available
- compile - compile the source code of the project
- test - test the compiled source code using a suitable unit testing framework.
- package - take the compiled code and package it in its distributable format, such as a JAR.
- integration-test - process and deploy the package if necessary into an environment where integration tests can be run
- verify - run any checks to verify the package is valid and meets quality criteria
- install - install the package into the local repository, for use as a dependency in other projects locally
- deploy - done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

- There are ways to circumvent

- mvn intall -DskipTests=true

Dependency Management



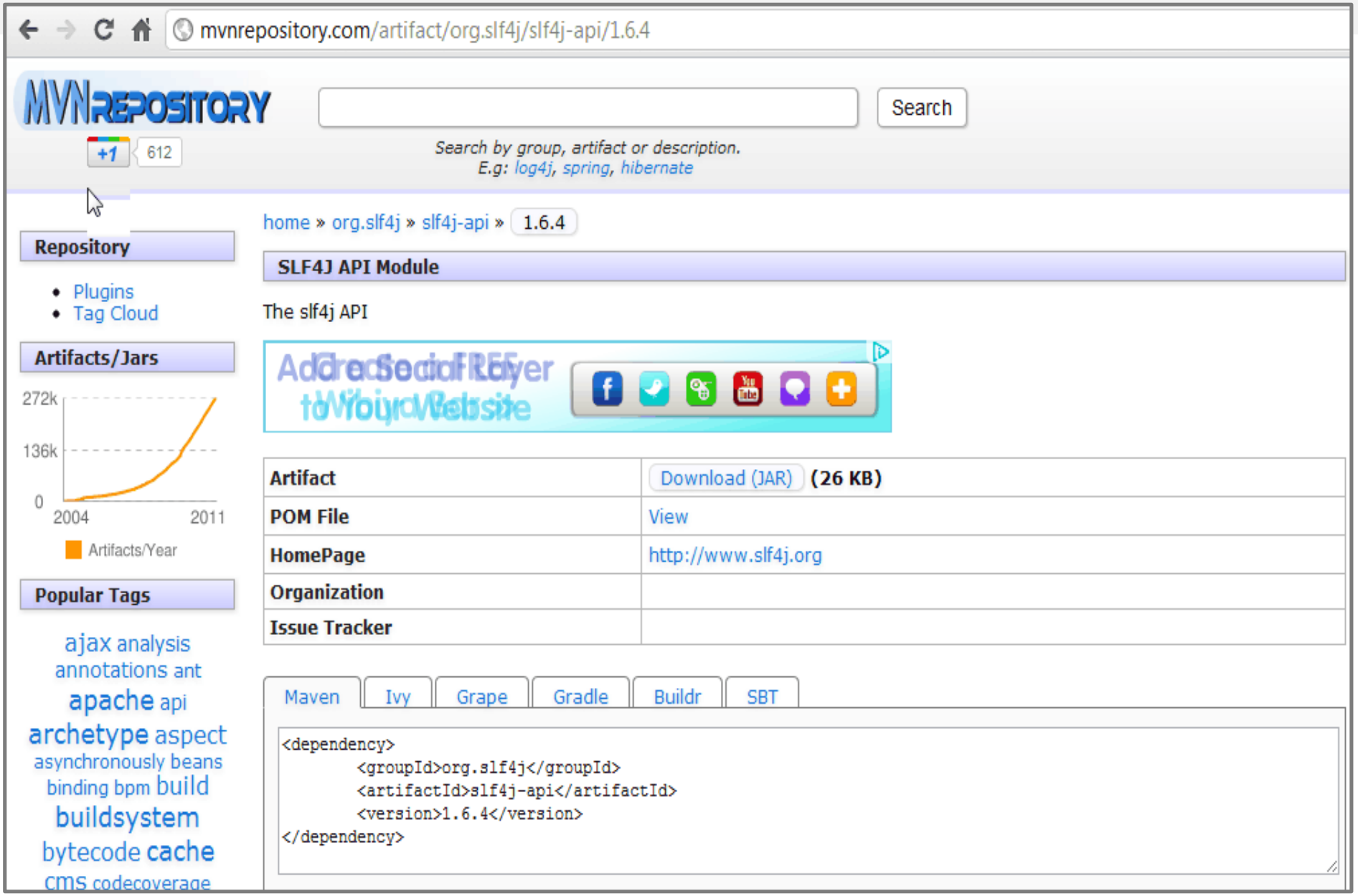


Simplifies dependency management

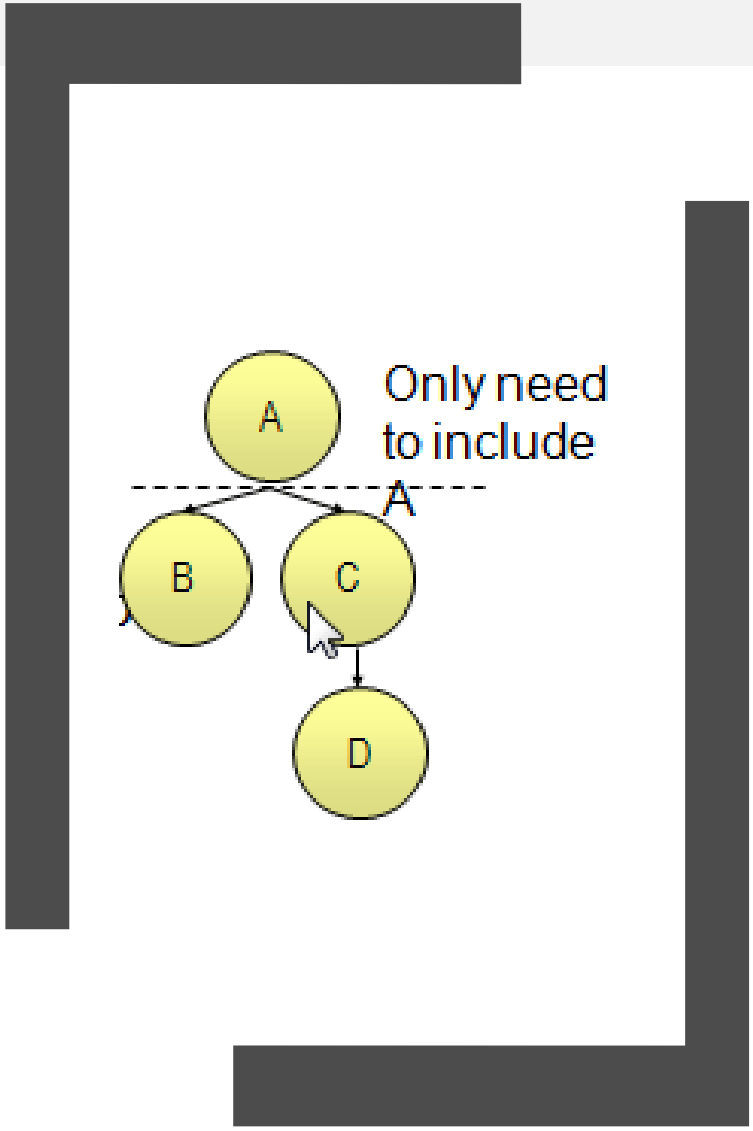
- All dependencies are specified in the pom
- Artifact repositories
    - Local
    - Remote
- No more JAR hunting on the web
- No more unknown versions of JARs in production releases
- Versions of common JARs can be specified in parent pom

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring</artifactId>
    <version>2.5.5</version>
</dependency>
```

Dependency Management

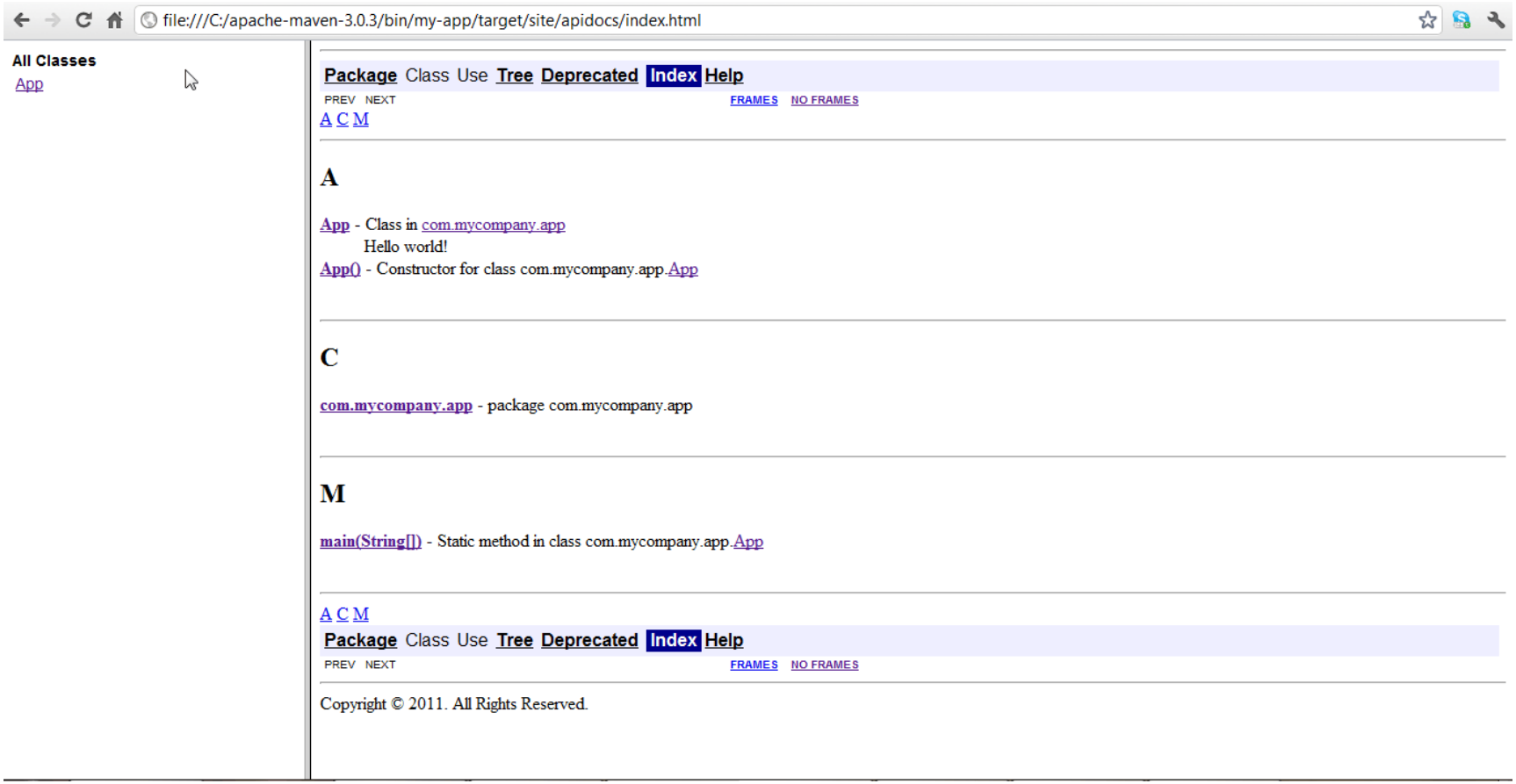


Transitive Dependencies



- Allows automatically inclusion of libraries
- Avoids the need to discover and specify the required libraries that your own

Report Generation



Continuous Integration (CI)

What ?

What is Continuous Integration?

- *" Continuous Integration is a software development practice where members of a team integrate **their work frequently, usually each person integrates at least daily** - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible."*

- http://martinfowler.com/articles/continuousIntegration.html

What is Continuous Integration?

"The Daily Build on Steroids"

- or…

"The **practice** of **integrating** source code **continuously**"

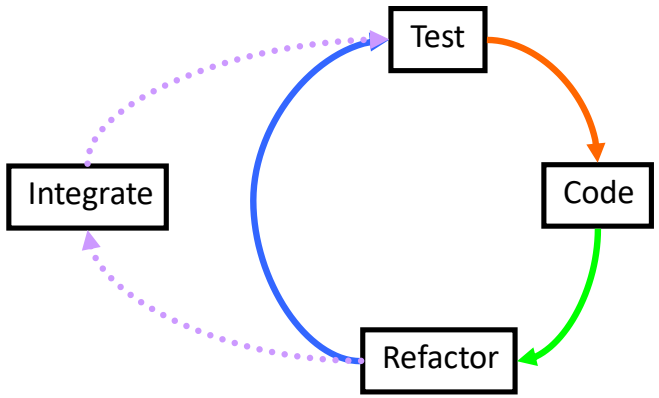
What is “integration” ?

- **at a minimum:**
  - Gather latest source together
  - Compile
  - Execute tests
  - Verify success
- **But it can also include other tasks such as:**
  - Rebuild the database
  - Build release distribution
  - Run code analysis and coverage tools
  - Generate documentation

How often is continuously?

- As frequently as possible
- More like once per hour than once per day
- Before leaving at the end of the day

When to integrate?



- Implement just enough, then integrate
- If using Test Driven Development, it forms a natural break in the cycle
- Taking small steps

Why?

- **Regular feedback**
  - For the integrator : “Did that work?”
  - For the rest of the team : “Is the build OK?
  - Reduces Risk overall
- **Reduce integration pain**
  - No more ‘merge hell’
  - XP Mantra: Do the ‘hard things’ often so they’re not hard any more
- **Enables concurrent development**
  - “We can both work on this today”
- **Increased automation**
  - Don’t repeat yourself - automate to increase speed and to make less mistakes

How?

Technical pre-requisites

- Source Code checked into Source Control
- Automated (fast) build
  - Compile
  - Test
  - command line without interaction
- Dedicated (communal) Integration Machine

Social pre-requisites

Developer discipline

- Continuous **means** continuous, not ‘once per week’

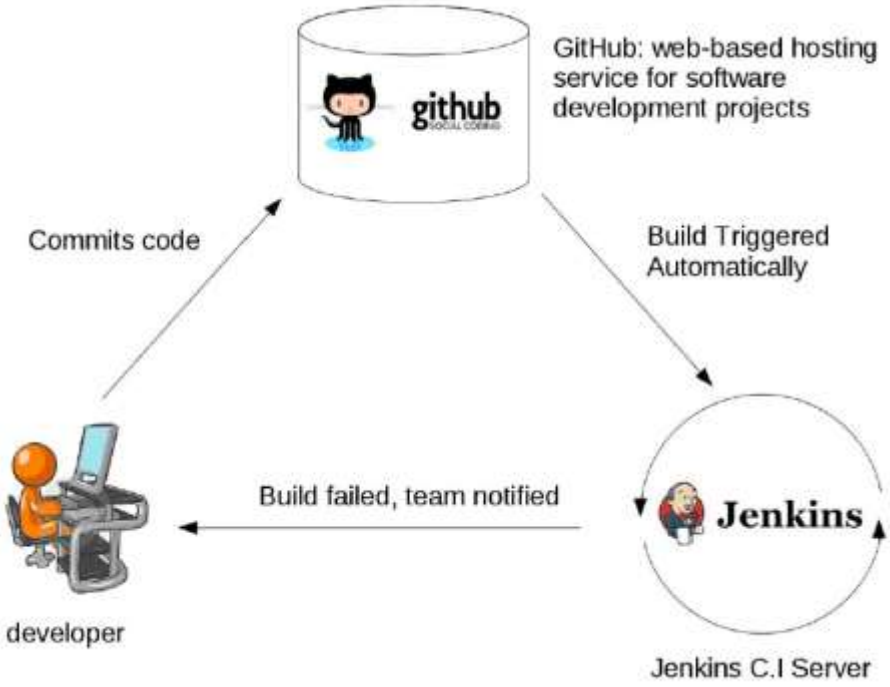
Shared ownership

Automated CI

Automated Continuous Integration Server

- CruiseControl, CruiseControl.NET, TeamCity, Bamboo, etc.
- Detects changes in source control
- Launches integration build
- Publishes results

How does Automated CI Work?



What is Continuous Integration?

- Developers commit code to a shared repository on a regular basis.
- Version control system is being monitored. When a commit is detected, a build will be triggered automatically.
- If the build is not green, developers will be notified immediately.

Why use Automated CI?

- Makes integration easy
- Guarantees integration happens
- Better feedback options
- Encourages test automation
  - Through metrics

Immediate Feedback is Key

Social Issues - CI Etiquette

Fixing a broken build is the highest priority

“You broke it, you organise fixing it”

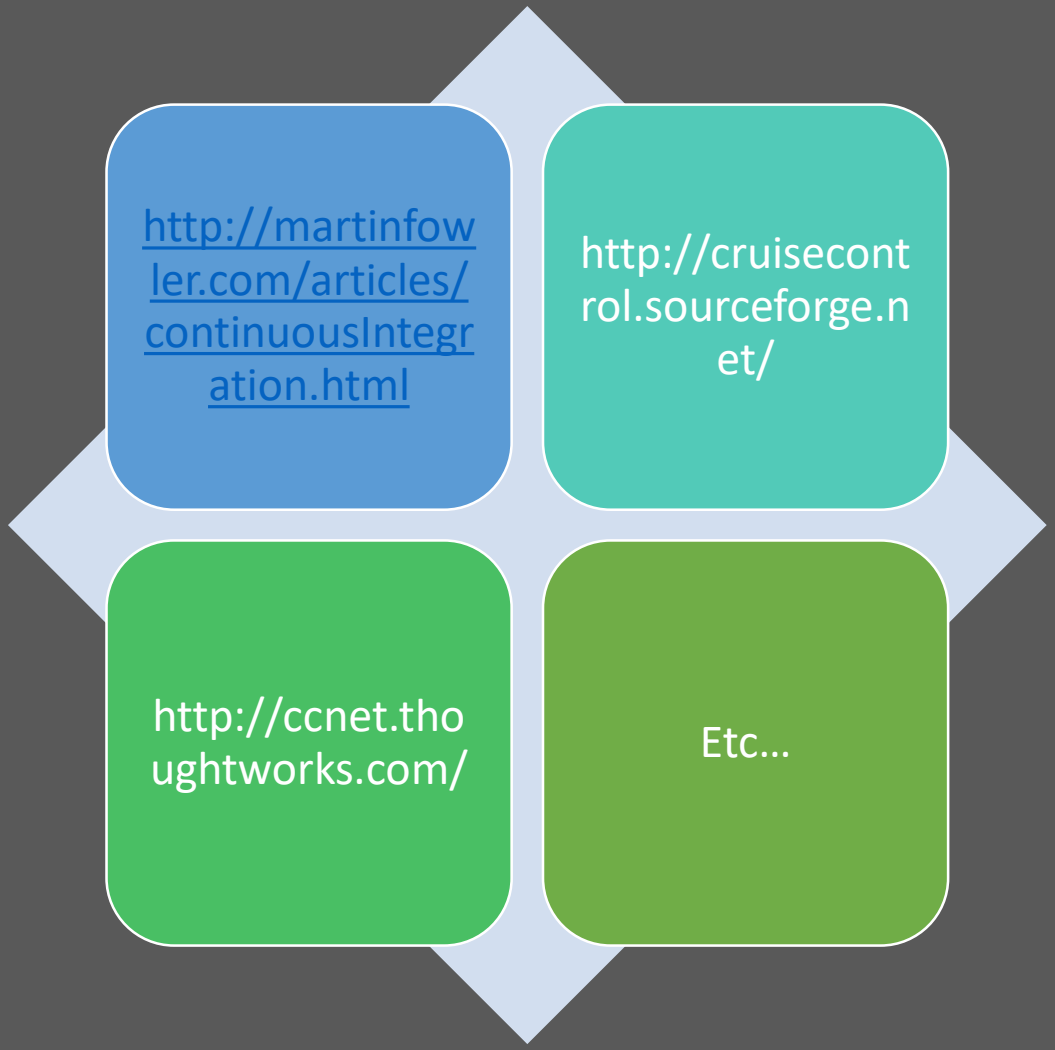
Do not check in on a broken build

It makes fixing it harder

Don’t leave until the integration runs successfully

Keep the build quick

Where next?


http://martinfowler.com/articles/continuousIntegration.html
http://cruisecontrol.sourceforge.net/
http://ccnet.thoughtworks.com/
Etc…