# Next Generation Maven Development Stack

What you're going to be using in the years to come

Jason van Zyl
http://twitter.com/jvanzyl

Sonatype

Writing build tools is always fun ... At least until people other then your friends start using them. Then you grow a very thick skin and try to listen patiently.

-- Every person who’s ever written a build system used by lots of people

## Agenda

What we’re going to talk about this session

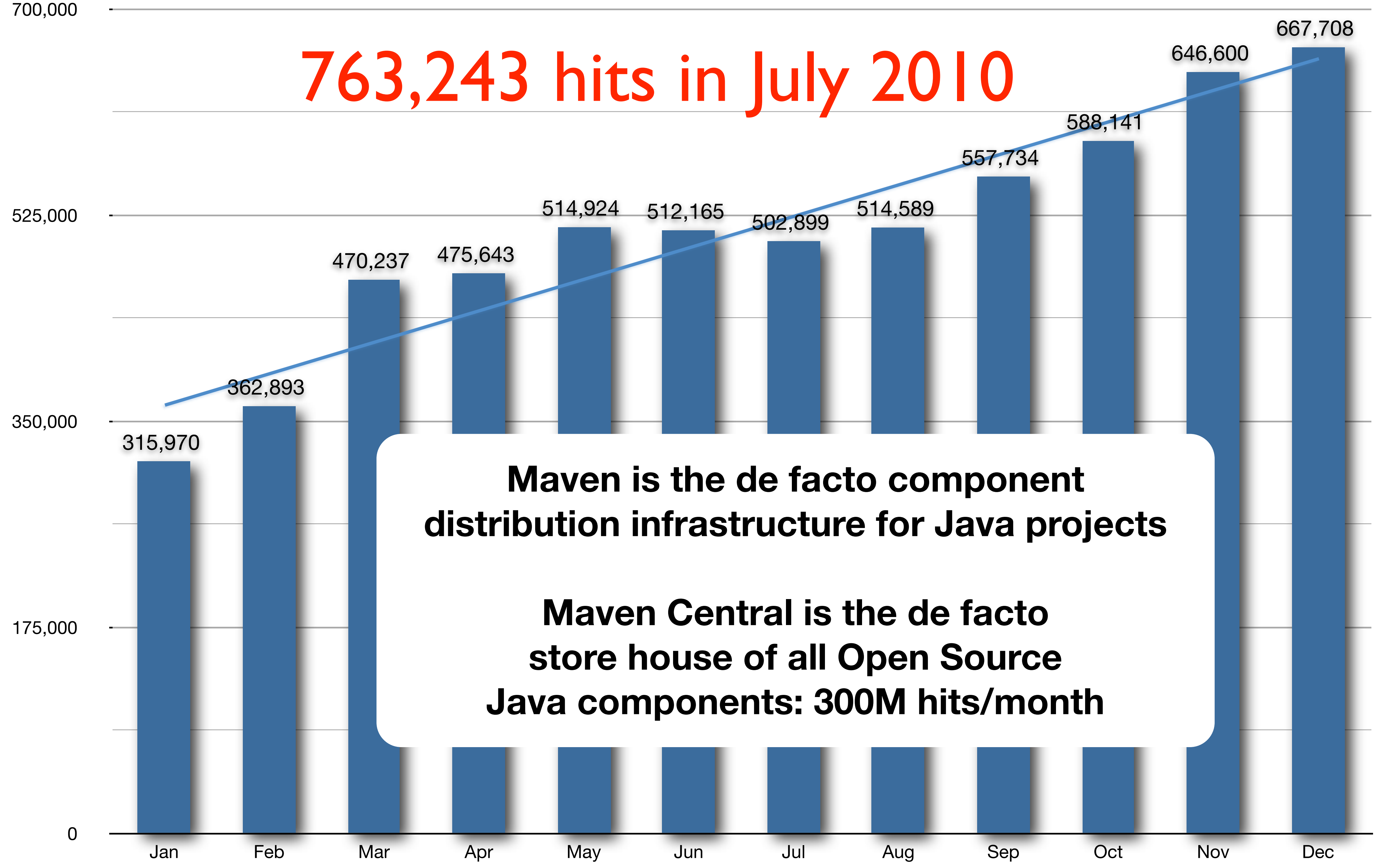
- News & Update
- Maven
- Tycho
- M2Eclipse & Maven Studio for Eclipse
- Proviso
- We’ll likely run out of time here ...
- Nexus
- Hudson
- JGitd
- Q & A

## New & Update

What’s going on at Sonatype & in Maven land?

- Maven Central & Quality
- Sonatype has released it’s second commercial
- Maven 3.0 is in beta and close to GA
- M2Eclipse 1.0 will follow closely
- Tycho is in the project review phase at Eclipse.org

## Maven Central Unique IPs / Month in 2009



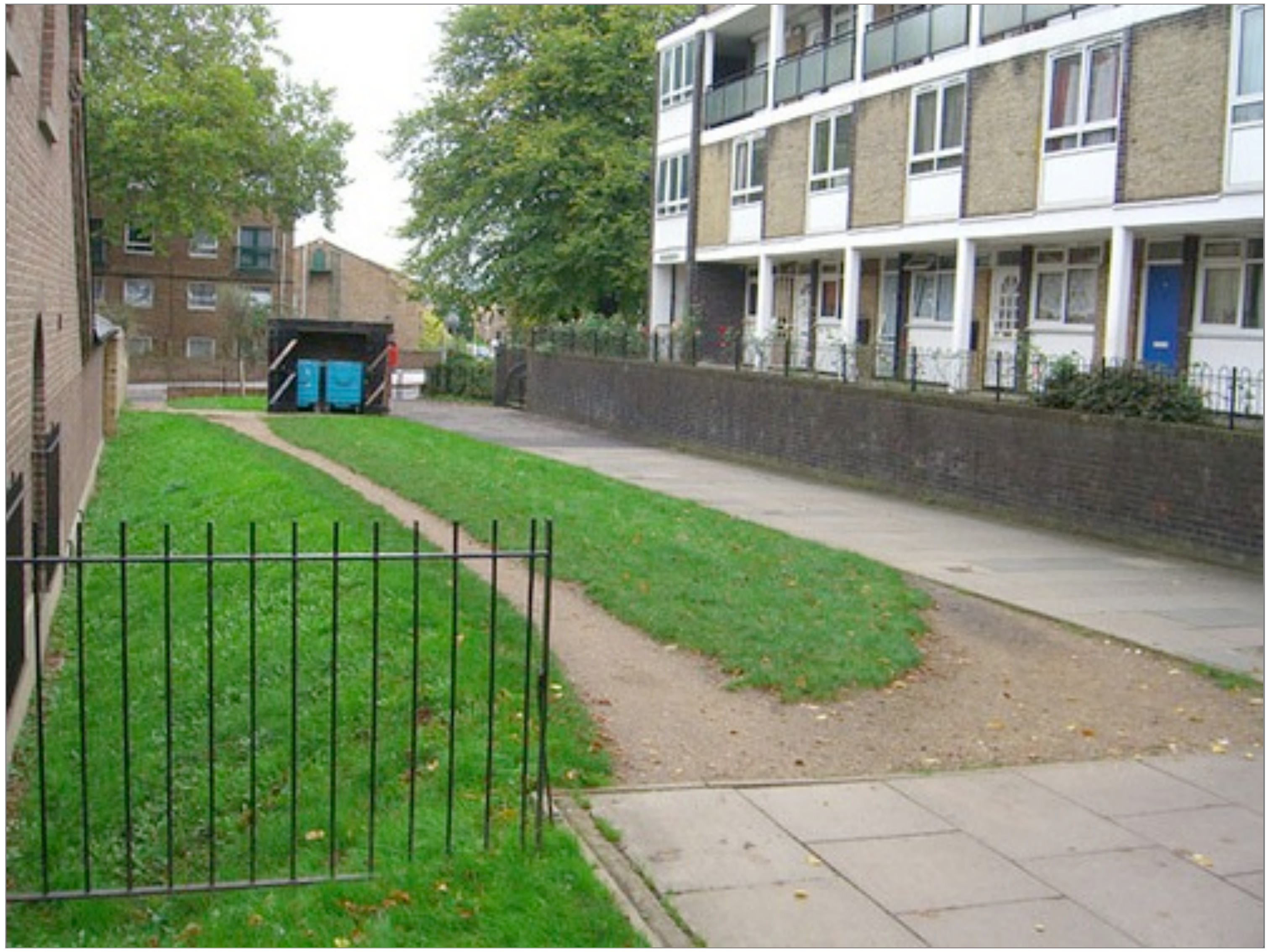
763,243 hits in July 2010

**Maven is the de facto component distribution infrastructure for Java projects**

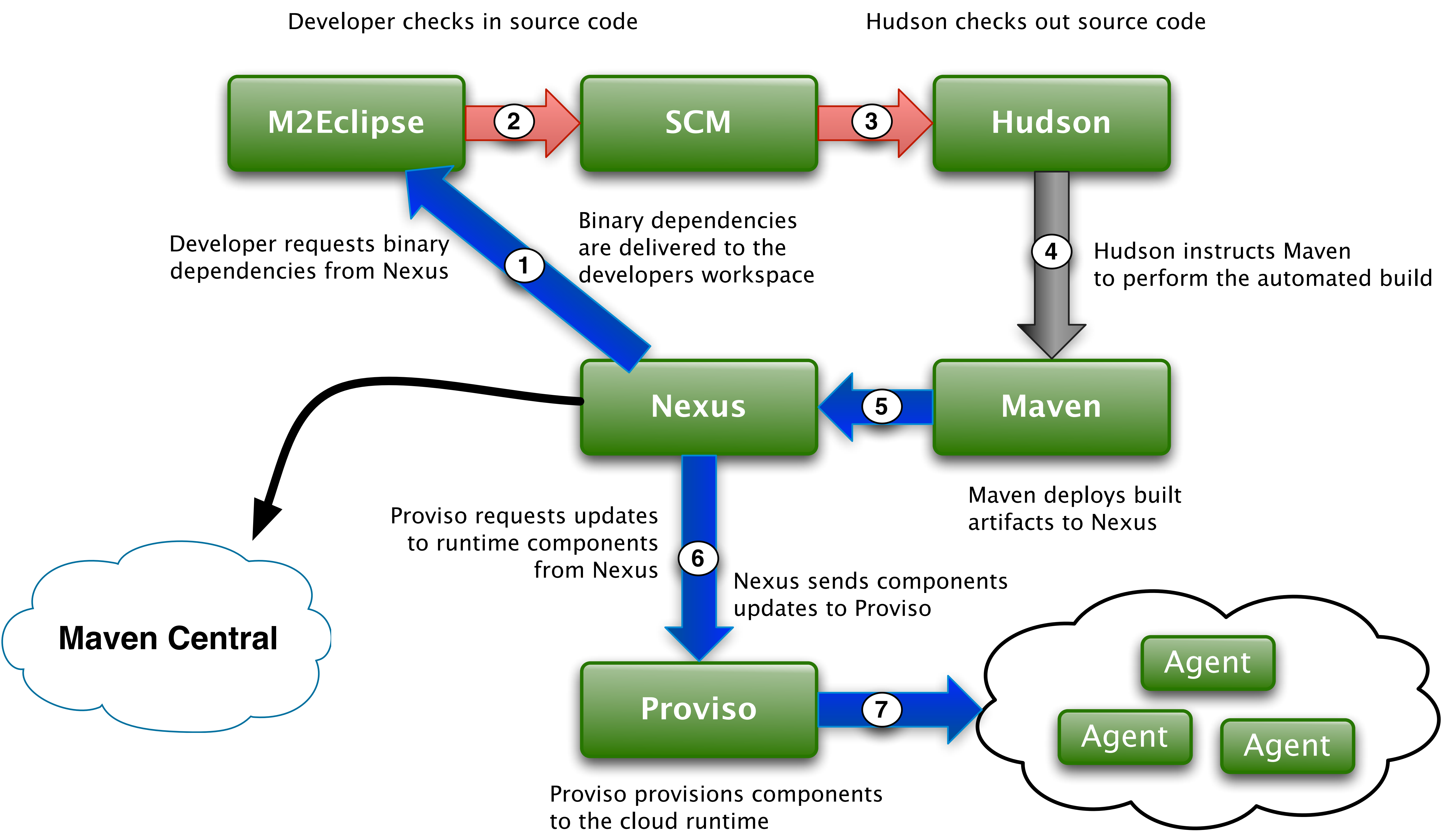
**Maven Central is the de facto store house of all Open Source Java components: 300M hits/month**

2008 Total Unique IPs: 1,836,709
2009 Total Unique IPs: 3,978,964

# Responding to invitations for improvement



# Maven is part of a bigger story: A complete business solution for Maven-based development


Developer checks in source code
Hudson checks out source code
M2Eclipse
2
SCM
3
Hudson
Developer requests binary dependencies from Nexus
1
Binary dependencies are delivered to the developers workspace
4
Hudson instructs Maven to perform the automated build
Nexus
5
Maven
Maven deploys built artifacts to Nexus
Proviso requests updates to runtime components from Nexus
6
Nexus sends components updates to Proviso
Maven Central
Proviso
7
Agent
Agent
Agent
Proviso provisions components to the cloud runtime

## Maven

What we're going to talk about

- Maven 3.x
- Maven Shell
- Polyglot Maven
- Improved Plug-ins

# Maven 3.0

A path forward for all Maven 2.x users

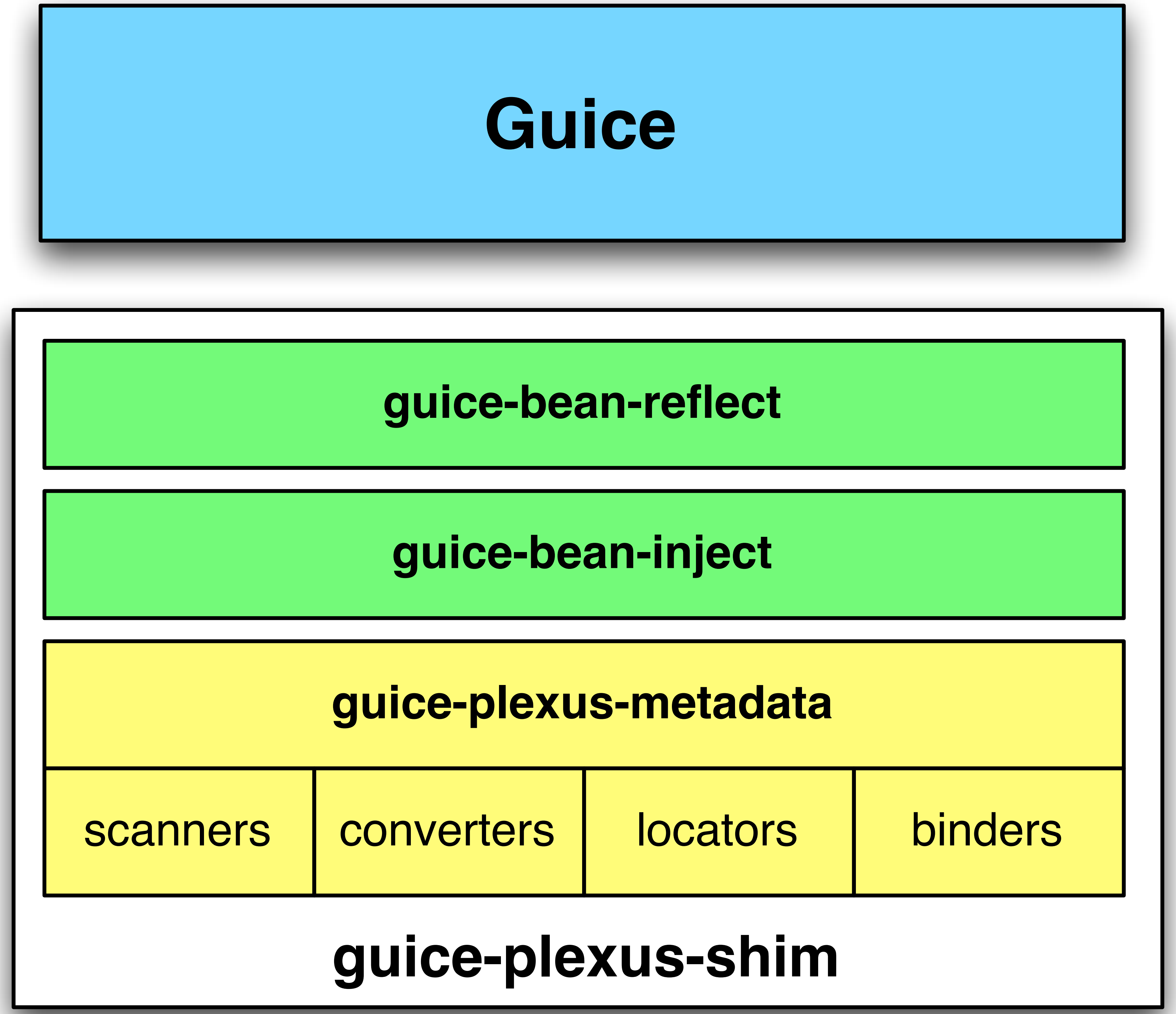
- We intend Maven 3.0 to be a drop in replacement for Maven 2.x users
  - Plugin API compatibility
  - Artifact Resolution API compatibility
  - POM format remains at version 4.0.0
- Overall speed improvements, especially with the parallel build support
- Complete internal overhaul for embedding include new incremental Plugin API
- Removal of site logic from the core -- everything has been moved to the site plugin
- Close to 700 integration tests
- Moving from Plexus to Guice
- Aether: the complete extraction of the repository API

# Moving from Plexus to Guice

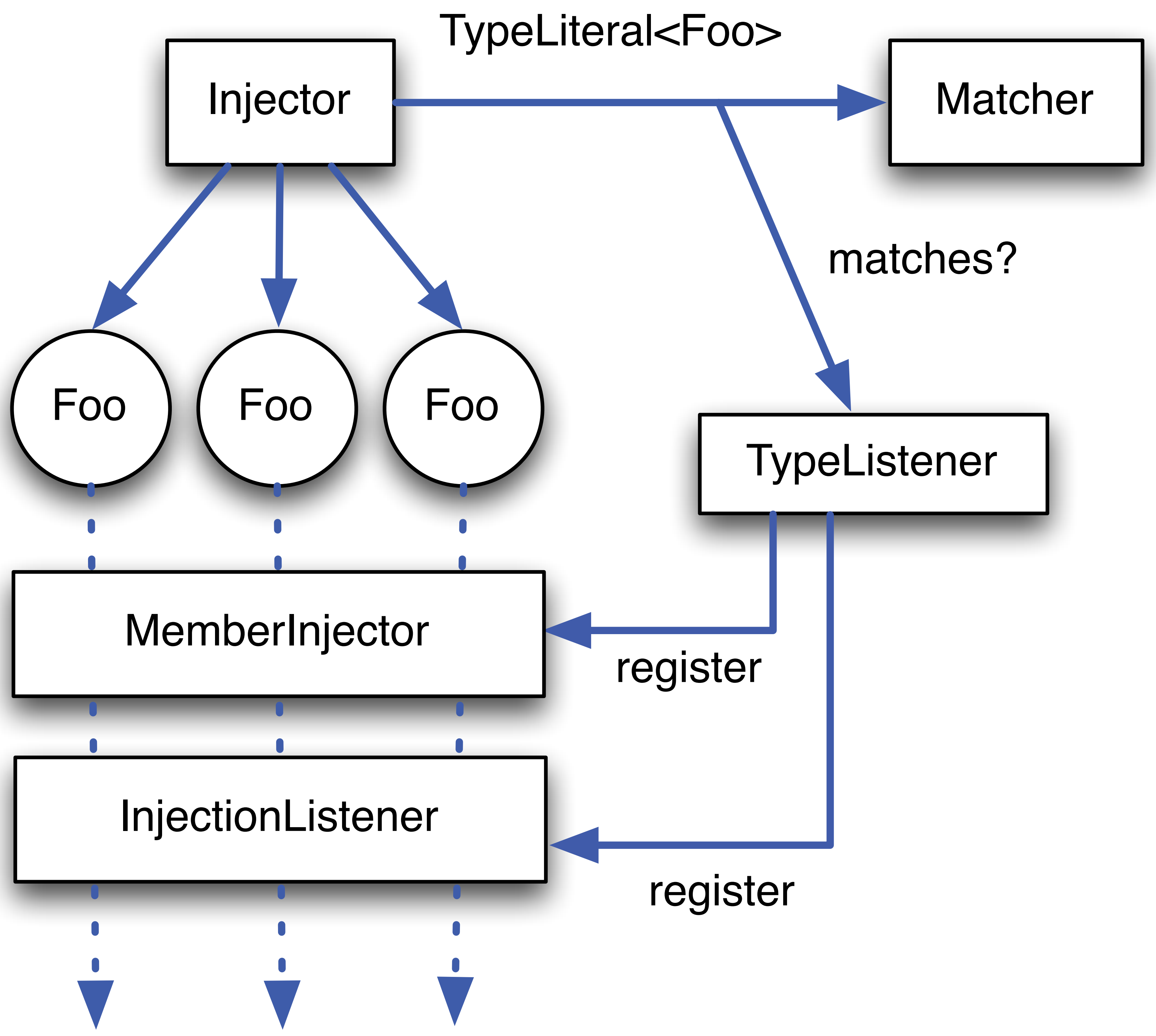
## Making it all work with Guice

- Everyone always asks ...
  - Why didn’t you use Spring?
  - Why aren’t you using Spring now?
- Requirements
  - Absolutely no code changes necessary for any Maven/Nexus user
  - Must support Plexus’ classpath/resource scanning
  - Must support Plexus’ dynamic component assembly based on discovered metadata
  - Must support Plexus’ configuration mechanism
  - When we need changes made to the used container need those changes to be timely
  - Support for arbitrary lifecycles
  - We need the container to be wed with OSGi -- for us the answer is Peaberry

# Plexus Guice



# Hooks for Custom Injections



## Aether

Overhauled Repository Artifact Resolution API

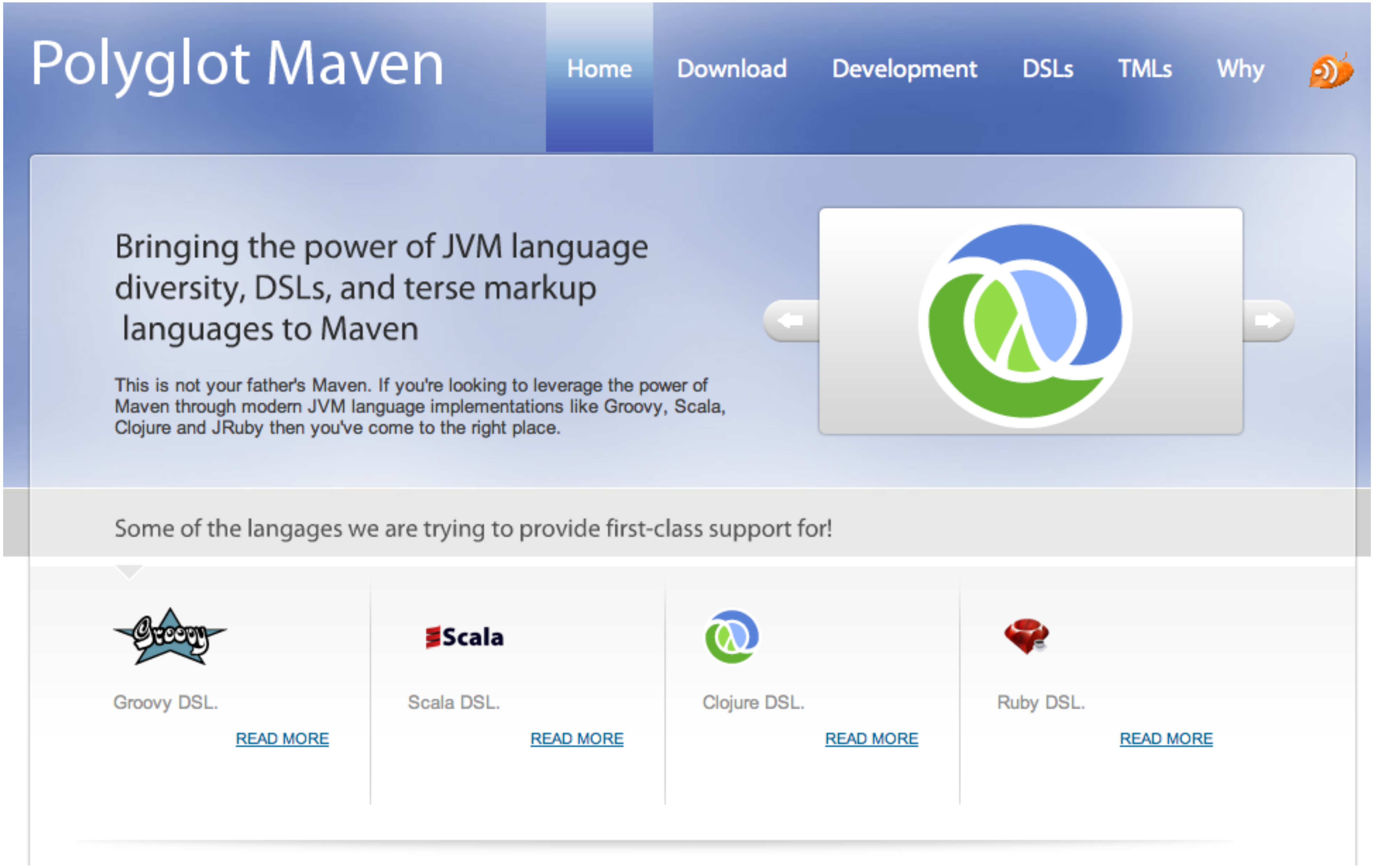
- We are really on Mercury take 2, the first attempt has been cast aside but it was a good learning experience
- The artifact resolution code has always been relatively decoupled, but Aether will be a completely separate project
- SSL support
- DAV support
- Transport
  - Originally based on the Jetty HTTP client ... but we're having some problems
  - We're now looking at the Async HTTP client being developed by Jean-francois Arcand at ~~Sun~~ ~~Oracle~~ ~~Ning~~ Sonatype
- Research by Pascal Rapicault and Daniel Le Berre to determine if P2 can be used to do Maven resolution. Ultimately we would like to merge our code into P2 and just use P2

## Maven 3.1

Taking advantage of changes in the Maven 3.x core

- POM format will change to version 4.1.0
  - Global excludes
  - Versionless parent elements
  - Mixins
  - Properties will return to dependencies
- A new settings system will be introduced
  - Repository manager element
  - Security manager element
  - Mirrors in profiles
- New Plugin API
  - Java5 annotations
  - Plugin extension points
- Dependency management akin to target platforms in Eclipse

http://polyglot.sonatype.org



Bringing the power of JVM language diversity, DSLs, and terse markup languages to Maven

This is not your father's Maven. If you're looking to leverage the power of Maven through modern JVM language implementations like Groovy, Scala, Clojure and JRuby then you've come to the right place.

Some of the langages we are trying to provide first-class support for!

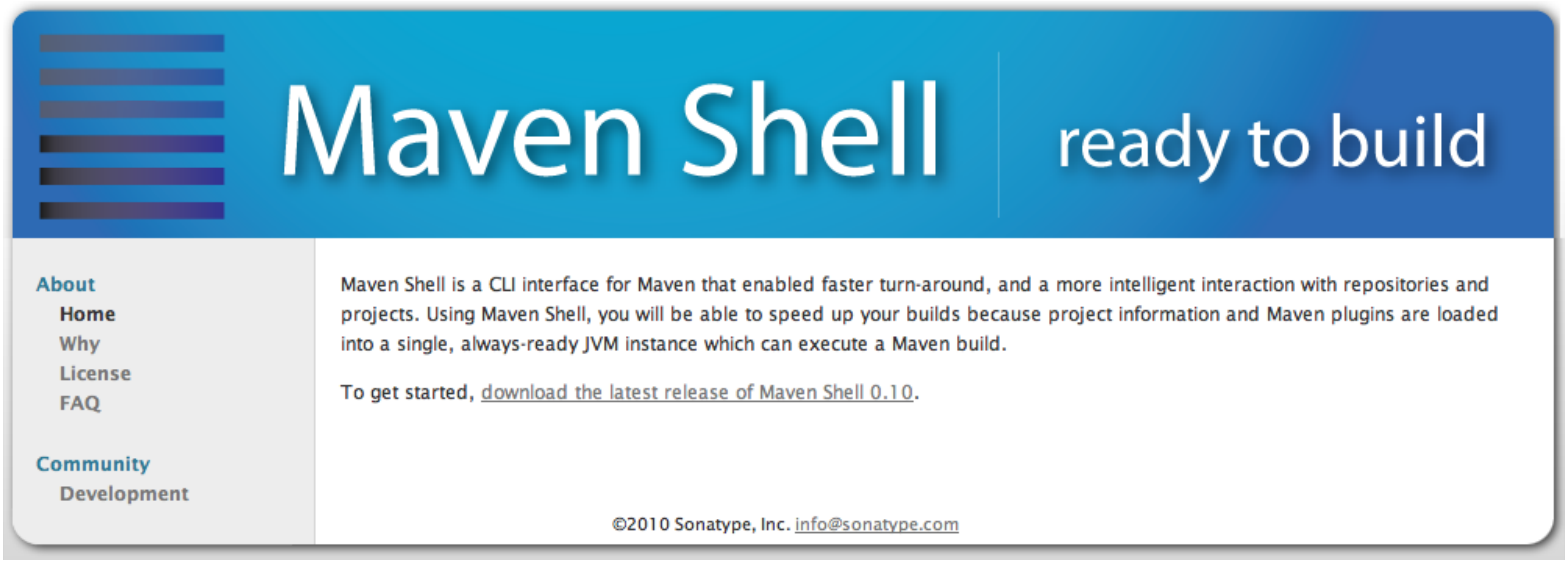
Groovy DSL. READ MORE

Scala DSL. READ MORE

Clojure DSL. READ MORE

Ruby DSL. READ MORE

http://shell.sonatype.org



Maven Shell is a CLI interface for Maven that enabled faster turn-around, and a more intelligent interaction with repositories and projects. Using Maven Shell, you will be able to speed up your builds because project information and Maven plugins are loaded into a single, always-ready JVM instance which can execute a Maven build.

To get started, download the latest release of Maven Shell 0.10.

©2010 Sonatype, Inc. info@sonatype.com

## NAR: AOL Classifier

Working with native code in Maven

- AOL Classifier specifies where the NAR file was created and where it will work:
  - Architecture
    - i386, x86, amd64, ppc, sparc, …
  - Operating System
    - Windows, Linux, MacOSX, SunOS, …
  - Linker
    - g++, gcc, msvc, CC, icc, icpc
- Examples:
  - x86-Windows-msvc, x86-Windows-g++
  - i386-Linux-g++, i386-Linux-icpc, amd64-Linux-g++
  - ppc-MacOSX-g++, i386-MacOSX-g++
  - sparc-SunOS-CC

## Tycho & The OSGi Ecosystem

### Making OSGi development a viable option

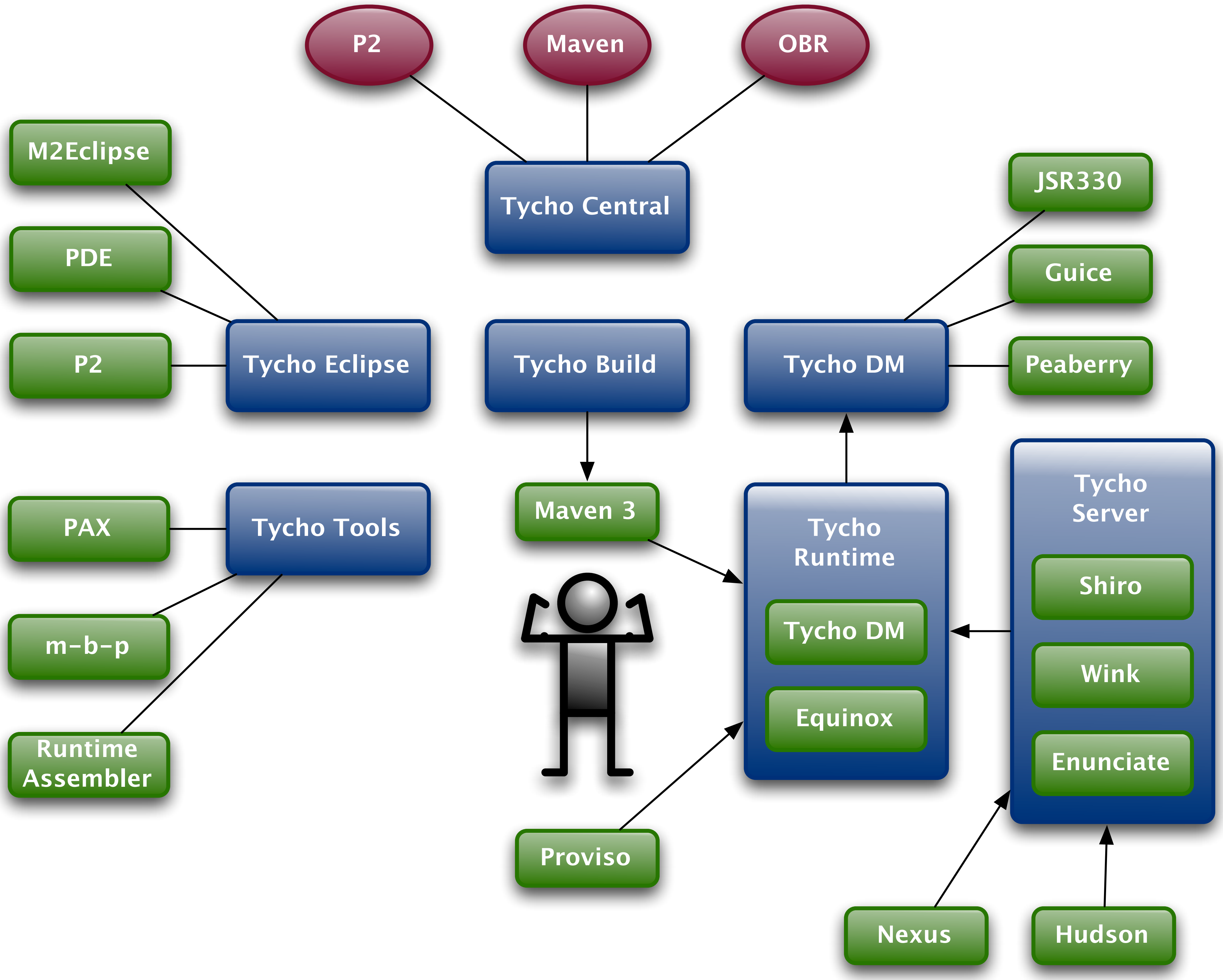
- Tycho has now become Sonatype’s overarching brand for OSGi
- We now call the Maven plugins that do OSGi “Tycho Build”
- We are absorbing most of PAX under the Tycho umbrella
- We are absorbing the maven-bundle-plugin under the Tycho embrella

## Tycho Build

A new way to build Eclipse plugins and OSGi Bundles

- Tycho attempts to be a complete replacement for PDE headless build, Buckminster, and everything else that attempts to build OSGi bundles and Eclipse plugins in a MANIFEST.MF-first way.
- Tycho already uses Eclipse/OSGi metadata to resolve project dependencies by OSGi rules and injects these dependencies into maven project model dynamically, at build time.
- Tycho uses JDT to make sure modularity rules are applied.
- Tycho uses the OSGi state resolver is used to make sure the resolution that occurs during build-time matches what you will need at runtime.
- Tycho supports bundles, fragments, features and update sites, as well as RCP applications. Tycho knows how to run JUnit test plugins using the OSGi runtime.
- Tycho has support for P2 repositories, Update Sites and Maven repositories.
- Tycho supports POM-first OSGi bundles.
- M2Eclipse has support for importing Tycho projects as Eclipse PDE projects.

# ~~Tycho~~ Sisu Ecosystem

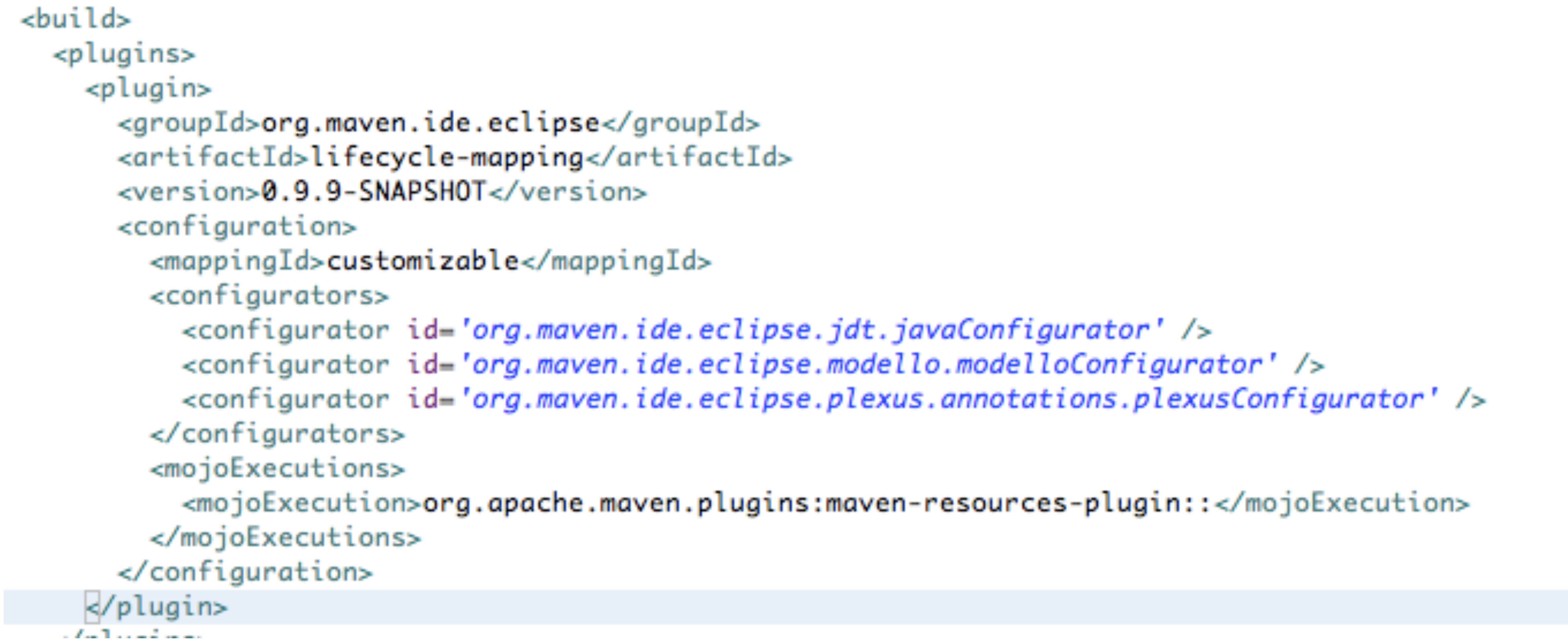


## M2Eclipse

What we're going to talk about

- Internal overhaul & aligned and synced with Maven 3.x
- Overhaul of the repository and index handling
- Customizable lifecycle mappings & Incremental APIs for Maven plugins -- pure speed
- Configuration framework
- The road to 1.0

# Customizable Lifecycle Mapping



```
<build>
  <plugins>
    <plugin>
      <groupId>org.maven.ide.eclipse</groupId>
      <artifactId>lifecycle-mapping</artifactId>
      <version>0.9.9-SNAPSHOT</version>
      <configuration>
        <mappingId>customizable</mappingId>
        <configurators>
          <configurator id='org.maven.ide.eclipse.jdt.javaConfigurator' />
          <configurator id='org.maven.ide.eclipse.modello.modelloConfigurator' />
          <configurator id='org.maven.ide.eclipse.plexus.annotations.plexusConfigurator' />
        </configurators>
        <mojoExecutions>
          <mojoExecution>org.apache.maven.plugins:maven-resources-plugin::</mojoExecution>
        </mojoExecutions>
      </configuration>
    </plugin>
```

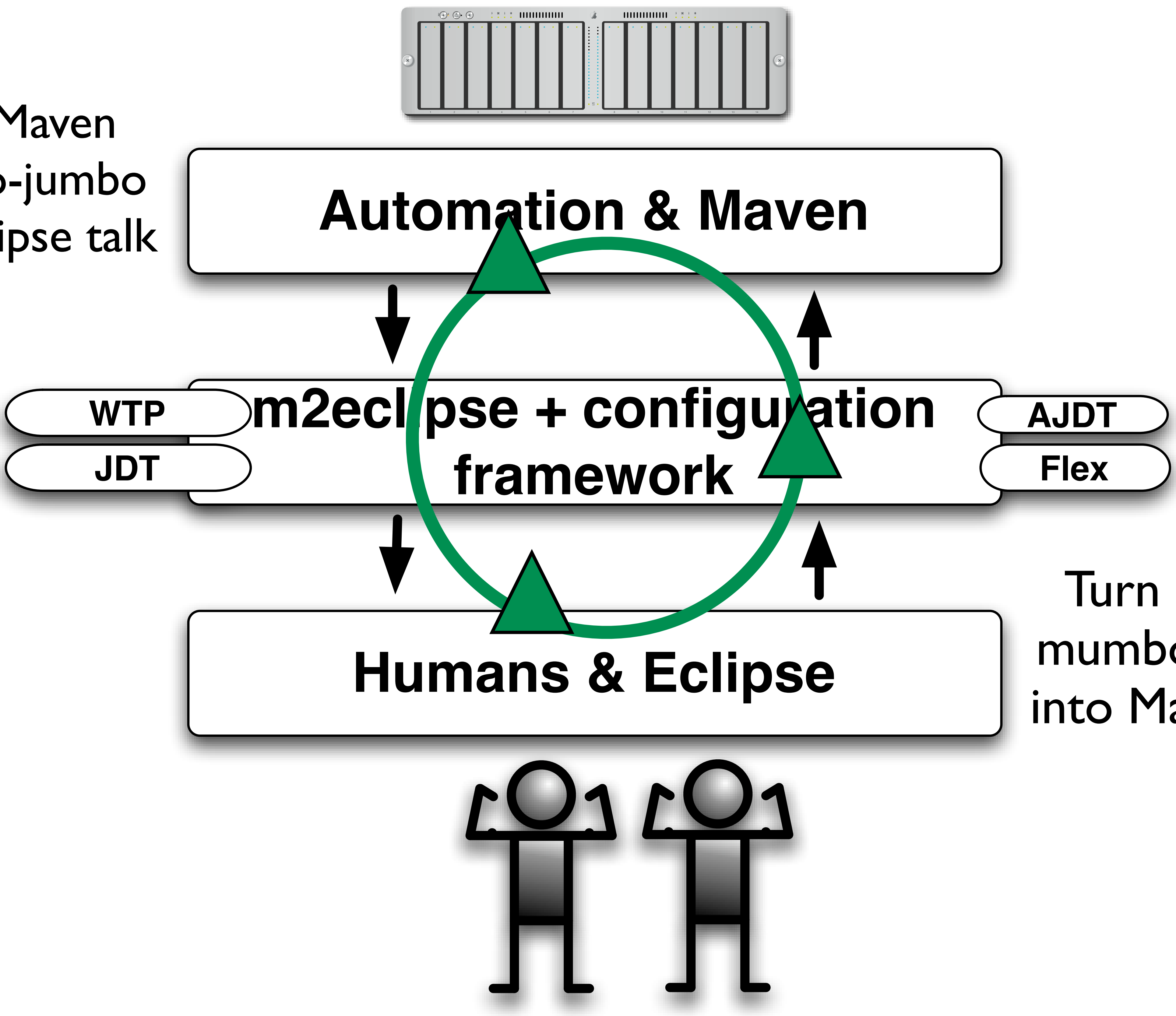
# M2Eclipse Configuration Framework

## Full access to Maven’s internals from Eclipse

- Initial project configuration during import or update configuration operations
- Maven project model change event notification
- Maven dependencies classpath container content manipulation
- Custom maven lifecycle mapping and build integration
- Full access to add/control natures and builders

# M2Eclipse Configuration Framework

Turn Maven mumbo-jumbo into Eclipse talk



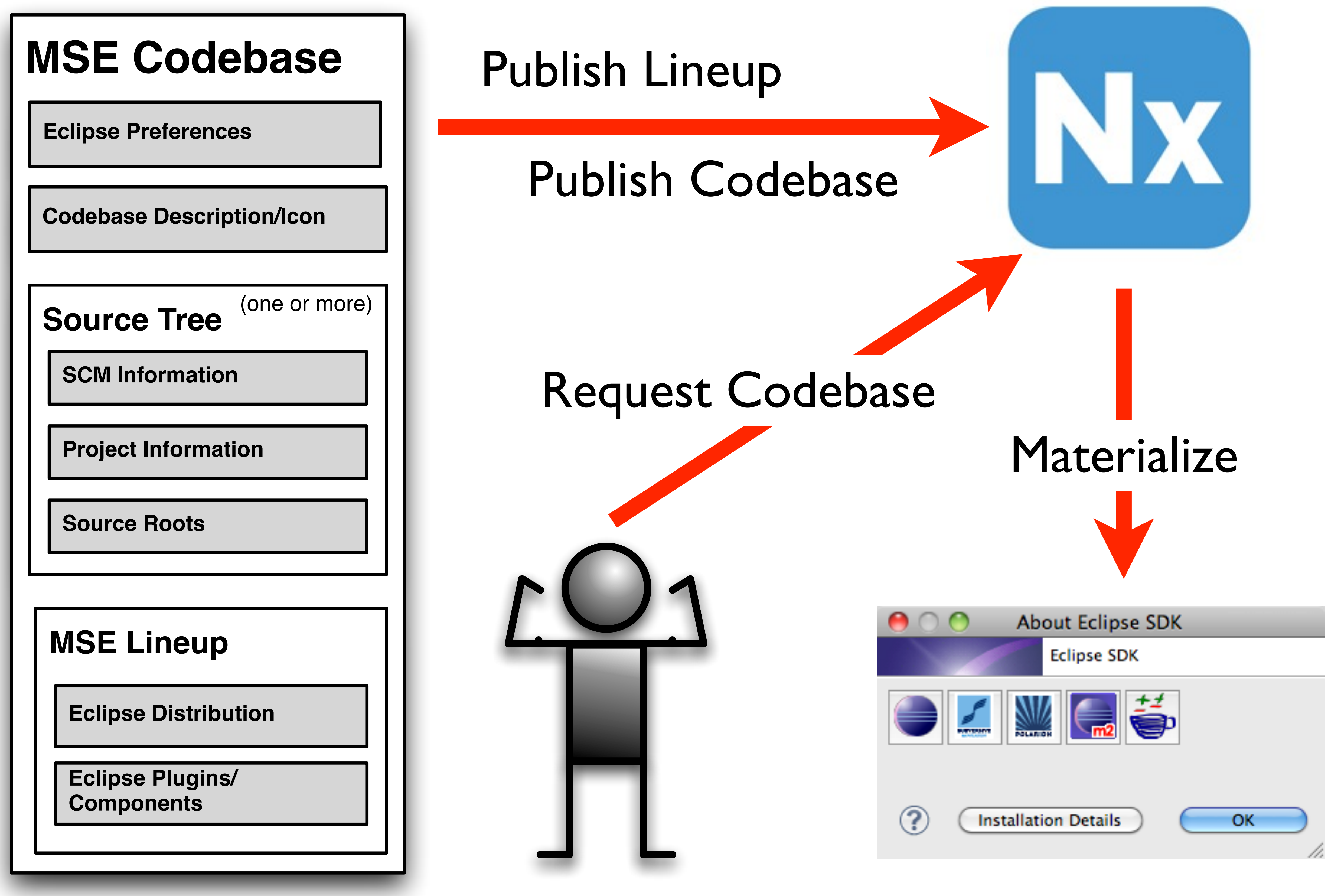
Turn Eclipse mumbo-jumbo into Maven talk

## Maven Studio for Eclipse

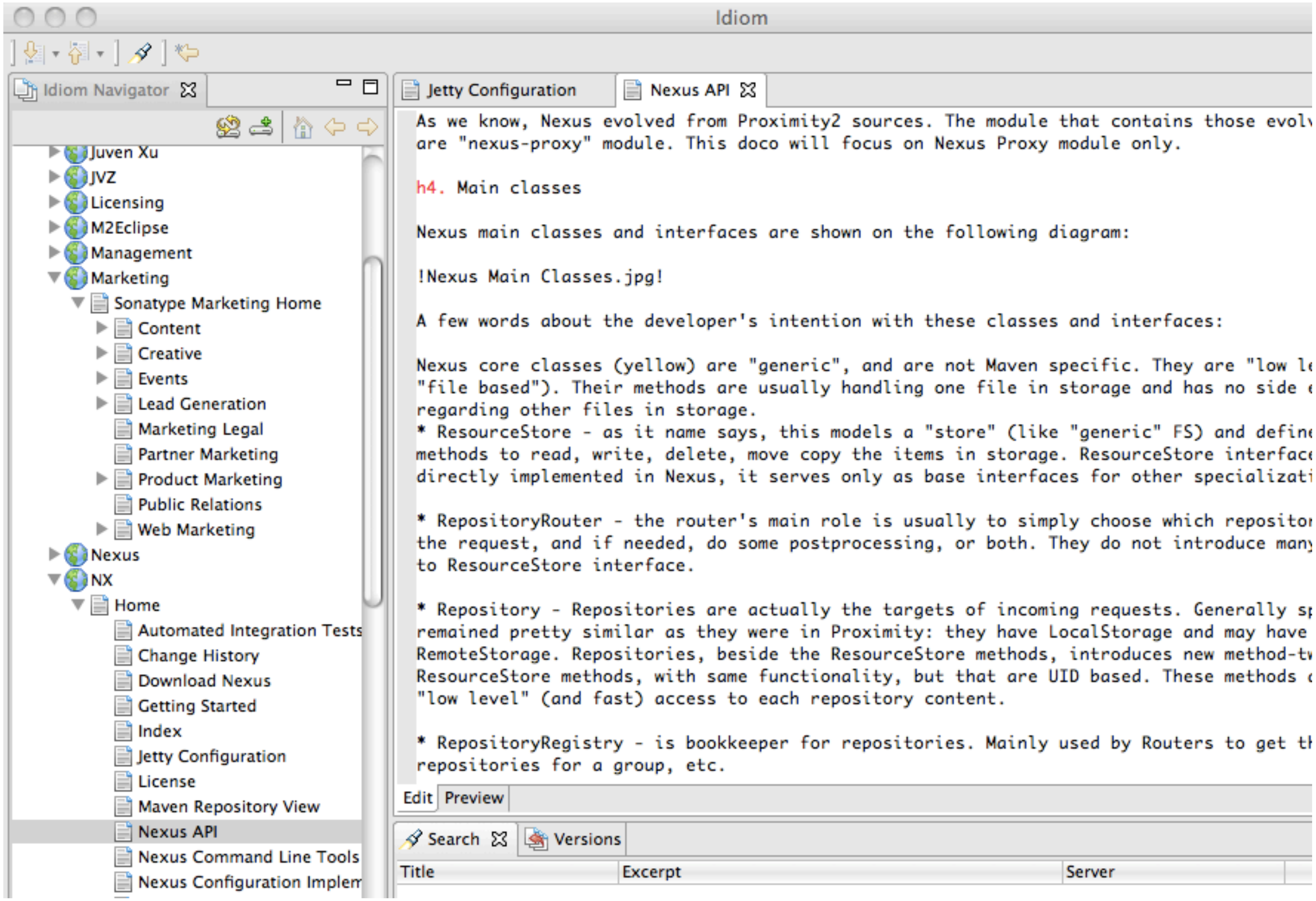
Sonatype’s commercial version of M2Eclipse

- 1.0
  - Developer onboarding & updating -- Huge! The ROI is the first day used!
  - Integration with Polarion’s Subversive to provide enterprise Subversion support
  - Integration with Hudson through an optimized REST interface Sonatype has created
- 1.1
  - Integration with Tomcat for rapid hot re-deploy for accelerated webapp development
  - Integration with Maven Archetype++ which optimizes creating and managing custom archetypes
  - Integration with Confluence & MediaWiki
- 1.2
  - Integration with JRebel
  - Integration with Nexus to be governance & compliance aware from project initialization
  - Integration with GWT

# Developer Onboarding & Updating



# Idiom: Confluence & Mediawiki Support



```
As we know, Nexus evolved from Proximity2 sources. The module that contains those evol
are "nexus-proxy" module. This doco will focus on Nexus Proxy module only.

h4. Main classes

Nexus main classes and interfaces are shown on the following diagram:

!Nexus Main Classes.jpg!

A few words about the developer's intention with these classes and interfaces:

Nexus core classes (yellow) are "generic", and are not Maven specific. They are "low le
"file based"). Their methods are usually handling one file in storage and has no side 
regarding other files in storage.
* ResourceStore - as it name says, this models a "store" (like "generic" FS) and define
methods to read, write, delete, move copy the items in storage. ResourceStore interface
directly implemented in Nexus, it serves only as base interfaces for other specializati

* RepositoryRouter - the router's main role is usually to simply choose which repositor
the request, and if needed, do some postprocessing, or both. They do not introduce many
to ResourceStore interface.

* Repository - Repositories are actually the targets of incoming requests. Generally sp
remained pretty similar as they were in Proximity: they have LocalStorage and may have
RemoteStorage. Repositories, beside the ResourceStore methods, introduces new method-tw
ResourceStore methods, with same functionality, but that are UID based. These methods 
"low level" (and fast) access to each repository content.

* RepositoryRegistry - is bookkeeper for repositories. Mainly used by Routers to get th
repositories for a group, etc.
```