Meister – Moving Beyond Maven

A summary whitepaper reviewing the differences between OpenMake Meister and Apache Maven

ASERVO Software

ASERVO Software GmbH

Konrad-Zuse-Platz 8 / 81829 München
Tel.: +49 (0) 89 7 16 71 82-40 / Fax: +49 (0) 89 7 16 71 82-55
Email: rmayr@aservo.com / www.ASERVO.com

OpenMake Software
www.openmakesoftware.com
312.440.9545 800.359.8049

Summary

Even though OpenMake Meister is a full DevOps solution, we often are asked how Meister is different from Maven. There are many similarities between OpenMake Meister's Build Services and Apache Maven. The most important of these similarities is the original purpose for their creation - simplifying and standardizing the compile/link process. With this common goal, both the developers of Apache Maven and OpenMake Meister came to similar conclusions on just how to accomplish a more simplified process.

At the core of both solutions is the implementation of standard templates, called Archetypes in Maven and Build Services in Meister. It is through this template concept that both products deliver a more standardized method for addressing the compile process. The implementation of Templates drastically reduces the need for developers to code build scripts from scratch every time they begin a new project. In addition, by employing a template methodology, build standards can be enforced making it much easier for anyone referencing that build script to understand what it does. Both Maven and Meister accomplish this goal.

How they are different

Even though both Maven and Meister's Build Services share a common thread, there are many differences between the two solutions. The biggest difference between the two tools is the way in which the concept of templates has been leveraged. With Maven, you use the Archetypes to initially create your Maven build scripts. Once you generate that initial script, you edit that script directly as your project changes and grows. In other words, the maintenance of the script is manual even though the creation of the script was automated based on the Archetype. With OpenMake Meister, the on-going maintenance of your project build script continues to be automated.

For example, with the Maven Eclipse integration, you can execute your Maven script inside of Eclipse, but Maven does not know about changes in your Eclipse project. This is done manually by editing the Maven scripts via the Eclipse editor. With Meister you can execute your Meister build inside of Eclipse, like Maven, but Meister goes one step further. Instead of manually editing your Meister build script, Meister does it for you by synchronizing the IDE project file with the build script before the build occurs. By providing a build process that is automated throughout the project's life, Meister leverages the benefits of templating beyond the first version of the build. The fact is that source code changes so the build must change as well. Meister delivers a consistent and repeatable build process by addressing the standardization of the build scripts beyond the initial creation.

Other difference that should be noted is that Apache Maven was written to support a Java development environment. OpenMake Meister, being a full DevOps solution, was designed to support multiple languages and operating systems including .Net, C-Unix, Oracle Forms and over 200 out of the box Build and Deploy Services. Meister Build and Deploy Services can be customized to support any type of compiler or development tool including products like Install Shield or Adobe.

OpenMake Meister includes a full Workflow engine with the support of distributed remote servers for managing the workload of the workflow and a built-in continuous integration server. Maven on the other hand must be implemented along with a workflow management solution such as Hudson, Cruise Control, or a commercial workflow solution if a higher level of workflow automation is needed or continuous integration is being used. Meister includes these features providing a complete integrated solution.

Dependency Management is an extremely important feature of any build solution. Apache Maven provides a level of dependency "scopes" for the management of compilers and runtime libraries, an important feature. However, Meister performs a much lower level of dependency discovery by scanning source code and interrogating IDE project files. It is through this level of dependency management that allows Meister to deliver accelerated build processing as well as providing a transparent build audit report where you can clearly see every artifact that was used in the build and where it came from. In addition, Meister uses this information to provide impact analysis data for making good development decisions and determining if a binary is ready to run in the "production" environment.

Acceleration of builds and deploys as well as the ability to perform incremental builds and deploys is also a feature exclusive to Meister. Because Meister automatically scans source for dependency relationships it can execute full builds in parallel reducing build and deploy times by as much as 50% or execute incremental builds and deploys. Incremental builds and deploys only re-builds and re-deploys the files that have been impacted by the change and avoids re-building and re-deploying the entire application.

Below is a feature comparison of the similarities and differences between Meister and Maven:

Feature Description		
Templates – Modular design of code	Meister	Maven
Use of Archetype Templates for supporting Best Practices	√	√
Ability to define best practices that fit unique environments. No need to restructure coding to fit into an out-of-box best practice.	√	
Archetype Templates for Cross Language support including .Net, Java, C-Unix	√	
Out of the box solution for Java build automation	√	√
Out of the box solution for non-Java build automation	√	
Dependency Management	Meister	Maven
Automatic Source to Source Dependency Management (.c to .h, .java to .java, .jar to .java, .c to .lib, etc.)	√	
Transitive Dependency Management	√	√
• Compile Scope	√	√
• Provided Scope	√	√
• Runtime Scope		√
• Test Scope		√
• System Scope	√	√
• Import Scope	√	√
Audit Trail of all Source and Compile Dependencies	√	
Impact Analysis of Source and Dependency Hierarchy	√	
Incremental build based on only changed source and dependencies	√	
Use of Approved Source and Compile Dependencies Only	√	
Automatic Download of Runtime Dependencies from Internet		√
Multiple Source and Compile Dependency Paths for same Project	√	
Deploy	Meister	Maven
Local Deployment of Shared Artifacts	√	√
Remote Deployment of Shared Artifacts	√	√

Support	Meister	Maven
Online and phone technical support	√	
Professional Services and Training	√	
Centralized Reporting	Meister	Maven
Aggregation of Report Output to a single repository for all Platforms	√	
Real-Time Monitoring of Build Results	√	
Build Acceleration	Meister	Maven
Parallelized Builds	√	
Build Avoidance (Incremental Build processing)	√	
Workflow Management (Build Lifecycle)	Meister	Maven
Standard Application Lifecycle	√	√
Customizable Application Lifecycle	√	
Reusable Application Lifecycle	√	
Distributed Remote Agents	√	
Scheduling	√	
Continuous Integration	Meister	Maven
Built-in Continuous Integration Server	√	
Check-in Changes monitored on Quite Period	√	
Monitor of SCM repository for changes	√	
Configurable CI processing based on Project	√	
Standards	Meister	Maven
Common Build Structure	√	√
Centralized Management of Compile Options	√	
Build can be standardized to a custom structure	√	
Higher level of reusability between Java builds	√	√
Higher level of reusability between non-Java builds	√	
Easy ramp up for new Java developers	√	√
Easy ramp up for new non-Java developers	√	
Calls Ant Tasks and Plug-ins	√	√

Distributed development	Meister	Maven
Supports development teams in different geographical locations	√	√
Centralized shared knowledge of all Build meta data	√	
IDE Support	Meister	Maven
External Build to Eclipse Integration (Maven to Eclipse)	√	
Eclipse to External Build Integration (Eclipse to Meister)	√	√
Full Support of Eclipse with an Eclipse RCP	√	√
.Net to External Build Integration	√	
Automatic generation of build scripts from Eclipse project data	√	
Automatic generation of build scripts from .Net project data	√	

Conclusion

The core benefit of both Meister and Maven is the use of a template methodology to deliver simplified builds. A template approach is critical for achieving a more standardized and consistent software build. Moving away from static scripting is a shared goal between both Maven and Meister. Meister takes this template approach and applies it throughout the software development process from creation through maintenance. Maven offers a strong alternative to OpenMake Meister for smaller java development teams where the application is not changing frequently and where clear audit trails are not required. OpenMake Meister delivers a broader solution supporting larger enterprise efforts where multiple languages and operating systems are used, the build itself needs to be accelerated, the application changes over time, or where transparency is needed to confirm that the source code managed in the version repository is actually reflected in the production executables.

Meister and Maven share a similar philosophy when it comes to the correct method for improving builds. Meister provides substantial benefits by leveraging that common philosophy and delivering a100% DevOps solution.

About OpenMake Software

OpenMake Software is the DevOps Authority providing an enterprise scale DevOpsframework for streamlining the build, test and deploy life cycle. Our solutions enable customers to reduce software development cycle times, improve communication between development and production control teams, increase developer productivity, and provide management with actionable audit and traceability reports. We go beyond what our competitors can offer by providing developers and production control a secure and dynamic operations framework that does not depend on brittle static scripts. Our solutions deliver on-demand cloud based server provisioning, environment configuration management, dependency management and continuous integration.
Over 100,000 users at 400 companies worldwide rely on the DevOps framework from OpenMake Software. For more information about OpenMake Meister or about OpenMake Software go to www.openmakesoftware.com or contact us at 312.440.9545, toll free 800.359.8049. You can email us at request-info@openmakesoftware.com