JSR Proposal: Java Configuration
Idea and Overview

Scot Baldry, Anatole Tresch
September 2014
**Configuration: Overview**

What is Configuration, anyway?

- Mechanism to change behavior of software for well defined aspects without rebuilding the code.
- Divergent views
  - Setup for a given server environment – virtualization, installation, localization
  - Deployment setup – which applications/modules to deploy
  - Technology-specific configuration – beans, wirings, interceptors
  - Resources – data sources, message queues, etc.
  - Scripting facility
  - Anything else!
- Different granularities, varying levels of applicability
- Java EE Config’s focus mainly on deployment and resources
- This JSR focuses for the general basic mechanism to enable configuration in a generic way
Configuration: Current State

- Several frameworks/solutions available for quite some time.
- Key/value based design has proven to work well in most scenarios.
- Configuration is layered and environment dependent.
- Java EE Configuration based on XML deployment descriptors.
- Java Environment Model (java.lang.System) is very limited.
- Other Java “configuration” mechanisms are insufficient/limited.
  - Preferences API has major flaws.
  - java.util.Properties lacks several functionalities.

-> Though Problem Domain is well known, no Standard is in place. Leads to unnecessary complexity affecting the whole Java platform.
Configuration: Java SE and Java EE

• SE Configuration focuses
  • on basic mechanisms
  • overriding and extendability
  • Application/use case specific configuration

• EE Configuration
  • Manages deployment and resources
  • With dynamic hooks for external logic (e.g. CDI, Bean Validation) SE configuration can be used as provider
  • EE application code can benefit from this JSR a lot
  • Generally Java EE as a platform is not affected by this JSR
Java Configuration: Overview about the proposed JSR

• **Functionality**
  • A flexible and extendible basic mechanism for Configuration
  • Supports layered and dynamic Configuration, composite mechanisms
  • A flexible abstraction for Environment
  • Provide functional extension points for advanced use cases
  • Consider Location Transparency and Remote Support

• **Design**
  • Create an initially modest service
  • Define Environment Specific Configuration and Stages
  • Define a complete SE API, including some annotations
  • Support Integration with CDI and other solutions
  • Provide a flexible bootstrap mechanism

• **Usability**
  • A solution that offers a great CX (Configuration Experience!) for DevOps and developers
Configuration: Summary
Benefits of a Java Config SE JSR

• Standardization would leverage the whole Java platform
• Simplifies module integration (Jigsaw compatibility must be ensured) and application code
• Reduces Complexity in many areas
• Doing it on SE level does not affect EE and enables its use throughout all relevant use cases.
• Focus on common configuration problem domain
• Basically no intersection with deferred Java EE Configuration
Links

- Java.net Project: http://java.net/projects/javamconfig
- Apache Deltaspark: http://deltaspark.apache.org
- Java Config Builder: https://github.com/TNG/config-builder
- Apache Commons Configuration: http://commons.apache.org/proper/commons-configuration/
- Jfig: http://jfig.sourceforge.net/
- Carbon Configuration: http://carbon.sourceforge.net/modules/core/docs/config/Usage.html
- Comparison on Carbon and Others: http://www.mail-archive.com/commons-dev@jakarta.apache.org/msg37597.html
- Spring Framework: http://projects.spring.io/spring-framework/
Q & A

???

???
The End

Thank you!