Apache Tamaya

Configuring your Containers...

About Me

Anatole Tresch
Principal Consultant, Trivadis AG (Switzerland)

Star Spec Lead
Technical Architect, Lead Engineer
PPMC Member Apache Tamaya

@atsticks
anatole@apache.org
anatole.tresch@trivadis.com

Agenda

- Motivation
- Containers, Config and Java
- Apache Tamaya
- The API
- Demo & Outlook

Motivation

What is Configuration ?

Key/value pairs?

Typed values?

When is Configuration useful?

How is it stored?

Remote or locally?

Classpath or file?

Format?

Multiple Sources?

When to configure?

Build time?

Deployment time?

Dynamic?

Configuration Lifecycle?

Static?

Dynamic?

Refreshing?

Changes triggered?

Do I need a runtime ?

Java SE?

Java EE?

OSGI?

Common approaches ?

- Hardcode everything
- Configure everything
- Use a monolithic configuration system
- Let each project/team decide (and implement !)

Microservices and Configuration

Using Java SE

- Environment Properties
- System Properties
- CLI arguments
- Properties, xml-Properties
- Proprietary solutions (Spring, Archaia etc)

Using Java EE

- Well known and established
- Deployment Config only
- CDI for „Application Configuration“ !
- New Config JSR for EE 8 in preparation !
- Mostly everything is XML

Using Something else...

- Files
- REST APIs

Java Based Solutions

- BYO (build your own)
- Spring Configuration
- Netflix Archaia
- Apache Tamaya
- Many more…

docker ?

Microservices run in Docker

- Configuration on deployment by **environment properties**:

```
docker run -e stage prod  -d -n MyApp user/image
```

Configuration with **Dockerfile/Docker Image**:

```
FROM java:8-jre
ADD /hello-drop-1.0.jar //
ADD /hello-config.yml //
EXPOSE 8090 8091
ENV stage prod
ENTRYPOINT ["java", "-jar", "/hello-drop-1.0-.jar",\
    "server", "/hello-config.yml"]
```

Problem Scope

- Multiple sources
- Multiple formats
- Multiple lifecycles
- Multiple priorities
- …
- Configuration Context, e.g. application name, stage

What do we need ?

An Abstraction.

An API.

Apache Tamaya

History of Apache Tamaya

- **2012**: Configuration was voted an important aspect for Java EE 8
- **2013**:
  - Setup of Java EE Configuration JSR failed
  - Standardization on SE Level did not have enough momentum
- **2016**
  - Concepts and API are clear
  - Release 0.2-incubating, 0.3-incubating until end of September ca.
  - New Config EE JSR in preparation by Oracle

The Objectives of Apache Tamaya

- Common API for configuration
  - Minimalistic
  - Flexible, pluggable and extendible
- Compatible with Java 7 and beyond
- Provide a reference implementation
- Provide Extension Modules for additional features
- Build up a community
- Create a Standard!

Decouple your code from...

- Format
- Storage
- Lifecycle and versioning
- Security
- Distribution
- Consistency

Injection API

```
@ConfigDefaultSections(”com.mycomp.tenantAdress”)
public final class MyTenant{

  private String name;

  @Config(defaultValue=”2000”)
  private long customerId;

  @Config({
   ”private”,”business”,”[company.adress]”
  })
  private String address;
  ...
}
```

```
MyTenant t = new MyTenant();
ConfigurationInjection
     .getConfigurationInjector()
     .configure(t);
```

```
@RequestScoped
public class MyClass{
   @Inject
   private MyTenant t;
   …
}
```

Programmatic API

```
Configuration config =
                ConfigurationProvider.getConfiguration();

String name = config.getOrDefault("name", "John");

int ChildNum = config.get("childNum", int.class);

Map<String,String> properties = config.getProperties();
```

Programmatic API

```
Configuration config =
                ConfigurationProvider.getConfiguration();

String name = config.getOrDefault("name", "John");

int ChildNum = config.get("childNum", int.class);

Map<String,String> properties = config.getProperties();
```

What else do we need ?

An abstraction for configuration sources and their ordening

=

PropertySource + Ordinals

PropertySource

```
public interface PropertySource {

  PropertyValue get(String key);
  Map<String,String> getProperties();
  boolean isScannable();
  String getName();
  int getOrdinal();
}
public final class PropertyValue{
  public String getKey();
  public String getValue();
  public String get(String key);
  public Map<String,String> getConfigEntries();
  ...
}
```

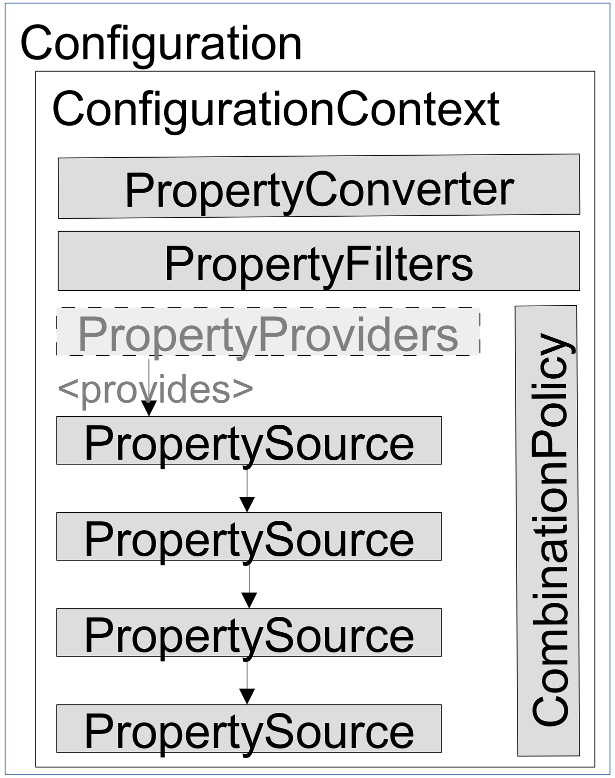
An abstraction for configuration sources…

=

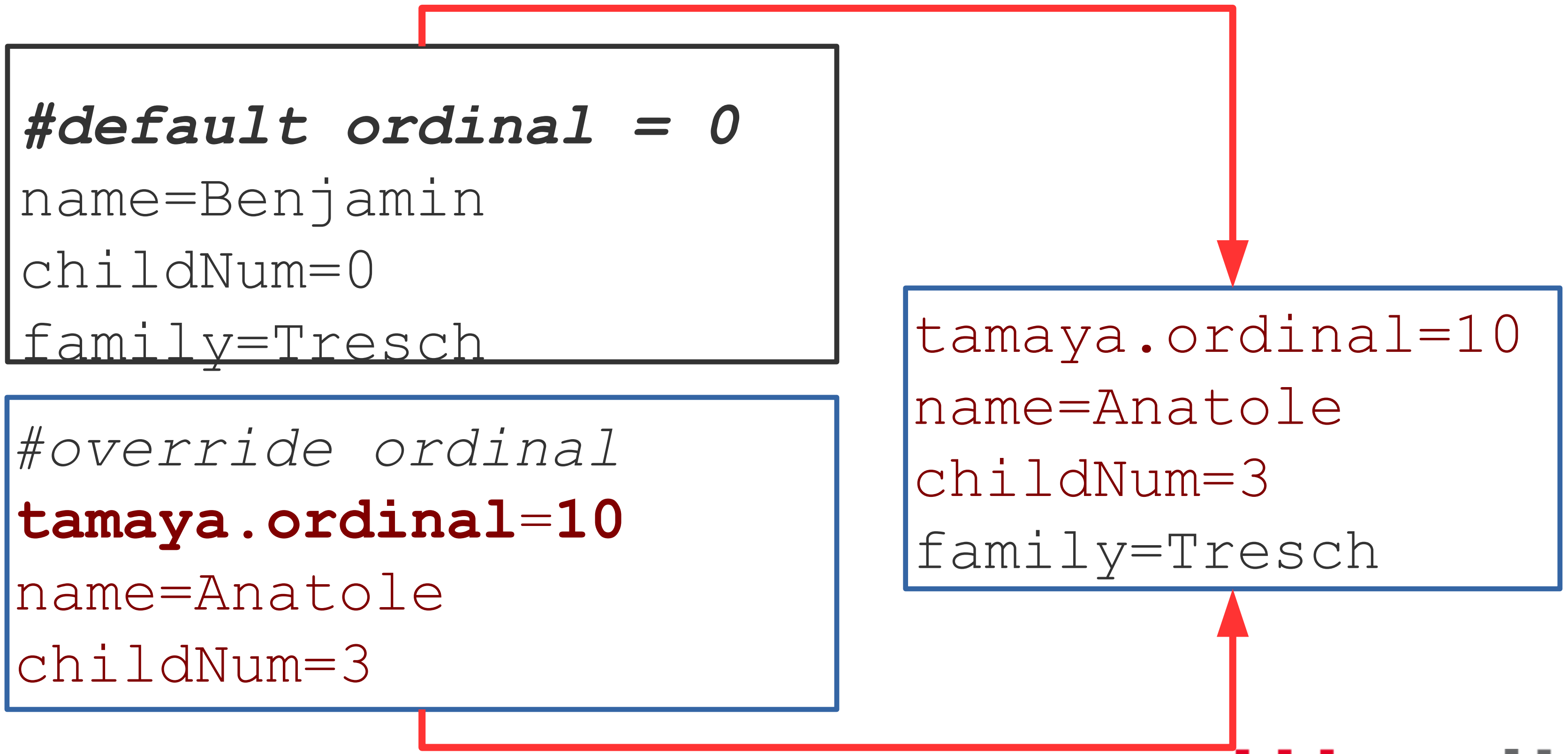
PropertySource

Tamaya Design in 120 Seconds...

1. `Configuration` = ordered list of `PropertySources`
2. Properties found are **combined** using a `CombinationPolicy`
3. Raw properties are **filtered** by `PropertyFilter`
4. For typed access `PropertyConverters` have to do work
5. **Extensions** add more features (discussed later)
6. **Component Lifecycle** is controlled by the `ServiceContextManager`



Configuration Overriding



```
#default ordinal = 0
name=Benjamin
childNum=0
family=Tresch
```

```
#override ordinal
tamaya.ordinal=10
name=Anatole
childNum=3
```

```
tamaya.ordinal=10
name=Anatole
childNum=3
family=Tresch
```

Upcoming Features ?

- Configuration Description and Validation
- Meta-Configuration: META-INF/configuration-sources.xml
- Integrations

Demo

- 1 Microservice
- Running on Java EE 7 (Wildfly)
- Multiple Configuration Sources:
  - Environment Properties
  - System Properties
  - Classpath
  - Files
  - Etcd Server

There is more! - Tamaya Extension Modules

Extensions: a topic on its own!

- *Tamaya-**spi-support**:* Some handy base classes to implement SPIs
- *Tamaya-**functions**:* Functional extension points (e.g. remapping, scoping)
- *Tamaya-**events**: Detect and publish ConfigChangeEvents*
- *Tamaya-**optional**: Minimal access layer with optional Tamaya support*
- *Tamaya-**filter**: Thread local filtering*
- *Tamaya-**inject-api**: Tamaya Configuration Injection Annotations*
- *Tamaya-**inject**:* Configuration Injection and Templates SE Implementation (lean, no CDI)
- *Tamaya-**resolver**:* Expression resolution, placeholders, dynamic values
- *Tamaya-**resources**:* Ant styled resource resolution
- *Format Extensions:* yaml, json, ini, … including formats-SPI
- Integrations with **CDI, Spring, OSGI*, Camel, etcd**
- *Tamaya-**mutable-config***: Writable ConfigChangeRequests*
- *Tamaya-**model****: Configuration Model and Auto Documentation
- *Tamaya-**collections****: Collection Support

…

* experimental

Summarizing...

- A Complete thread- and type-safe Configuration API
- Compatible with all major runtimes
- Simple, but extendible design
- Extensible
- Small footprint
- Base for current Java EE 8 spec ?

You like it ?

It is your turn !“

- *Use it*
- *Evangelize*
- *Join the force*

Links

Project Page: http://tamaya.incubator.apache.org
Twitter: @tamayaconfig
Blog: http://javaeeconfig.blogspot.com
Presentation by Mike Keith on JavaOne 2013:
https://oracleus.activeevents.com/2013/connect/sessionDetail.ww?SESSION_ID=7755
Apache Deltaspike: http://deltaspike.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/
Owner: http://owner.aeonbits.org/

Q&A

Thank you!