trivadis
makes IT easier.

APACHECON EUROPE

Anatole Tresch & Werner Keil

Trivadis AG

@atsticks @wernerkeil

trivadis
makes IT easier.

APACHECON EUROPE

Apache Tamaya

Configuration Management

Bio

Anatole Tresch

- Principal Consultant, Trivadis AG (Switzerland)
- Star Spec Lead JSR 354
- Technical Architect, Lead Engineer
- PPMC Member Apache Tamaya

- Twitter/Google+: @atsticks
- anatole@apache.org
- anatole.tresch@trivadis.com
- JUG Switzerland, Zurich Java Community

Bio

Werner Keil

- Consultant – Coach
- Creative Cosmopolitan
- Open Source Evangelist
- Software Architect
- JCP EC Member
- Tamaya Committer
- Java EE | DevOps Guy …

- Twitter/Google+: @wernerkeil
- wkeil@apache.org

Agenda

- General Infos
- What is Configuration?
- Use Cases
- Apache Tamaya
  - Core Concepts
  - Extensions
- Outlook

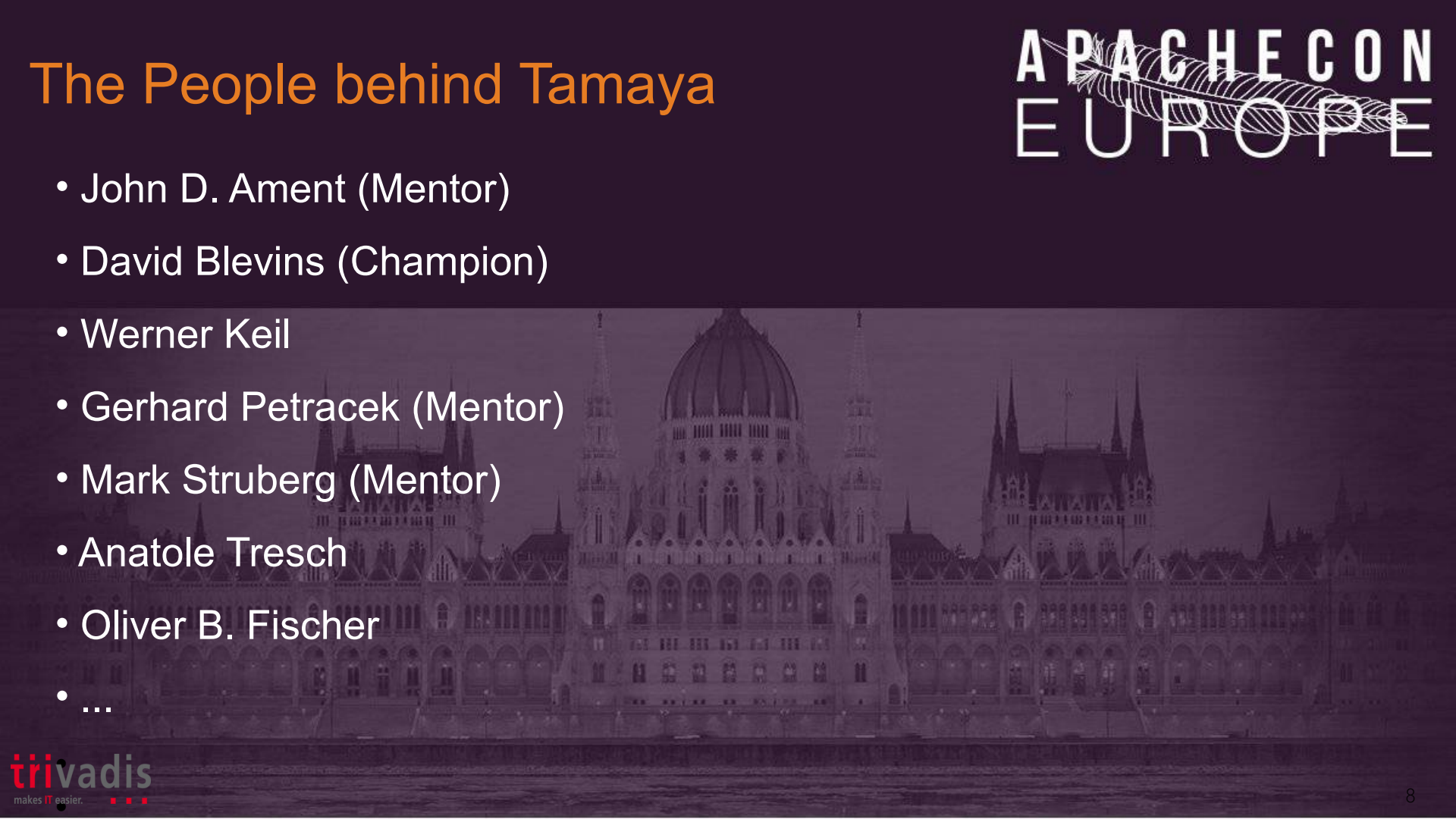
General Infos

APACHECON EUROPE

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
- **BUT**:
  - Configuration is a crucial cross cutting concern
  - There is no (really NO!) defacto standard
  - Reinventing the wheel is daily business

The People behind Tamaya



- John D. Ament (Mentor)
- David Blevins (Champion)
- Werner Keil
- Gerhard Petracek (Mentor)
- Mark Struberg (Mentor)
- Anatole Tresch
- Oliver B. Fischer
- ...

The Objectives of Apache Tamaya

- Define a common API for accessing configuration
  - Minimalistic, also fits into ME
  - Flexible, pluggable and extendible design
  - support functional programming style
- Be compatible with Java 7 and beyond
- Provide a reference implementation
- Provide Extension Modules for Additional Features
- If possible, create a Standard!
-

What is Configuration?

What people think is configuration?

- Many different interested stakeholders
- Many things to configure
- Divergent views
  - Setup for a server environment
  - Parameters of a runtime (staging, localization etc.)
  - Deployment descriptors
  - Technology-specific components (beans, wirings etc.)
  - Resource-specific settings (data sources, message queues etc.)
  - Dynamic scripting facility, Different scopes (global, ear, app, …)
- Different granularities, varying levels of applicability, Different formats …

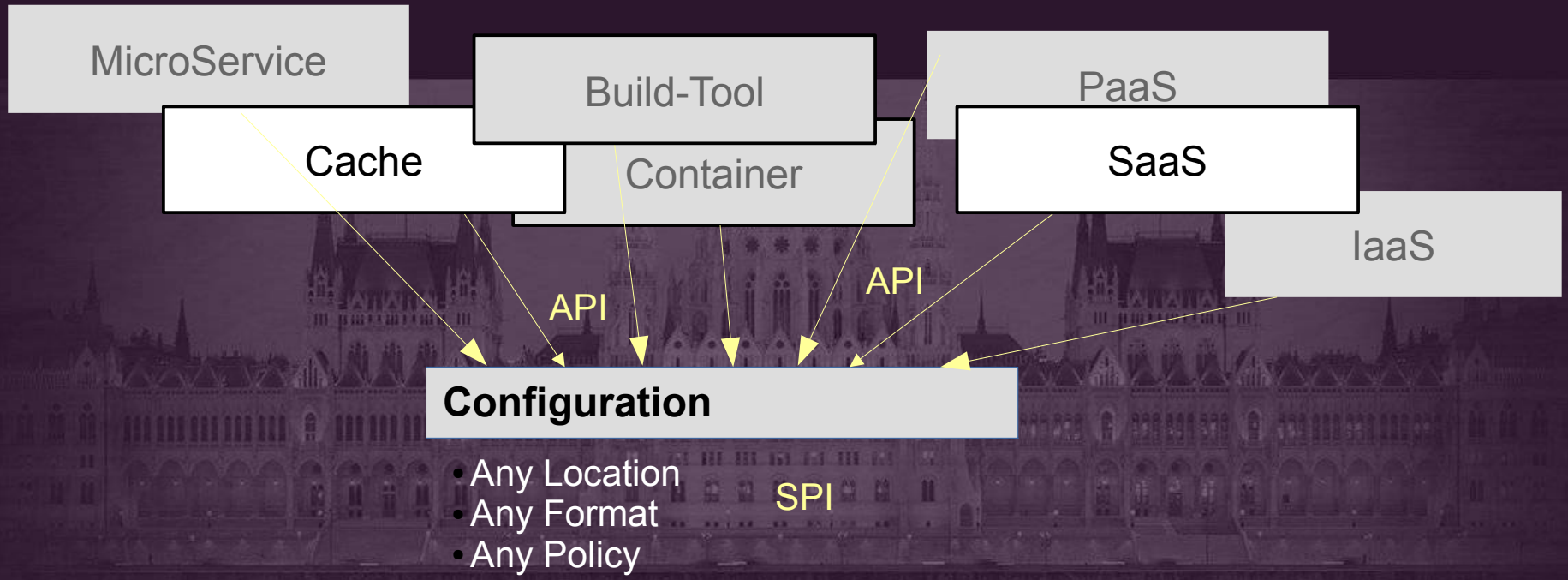
What we think is configuration

Configuration:
- Data, Value Types
- Transitive Deps
- Scopes
- Key, value pairs
- Static
- Serializable

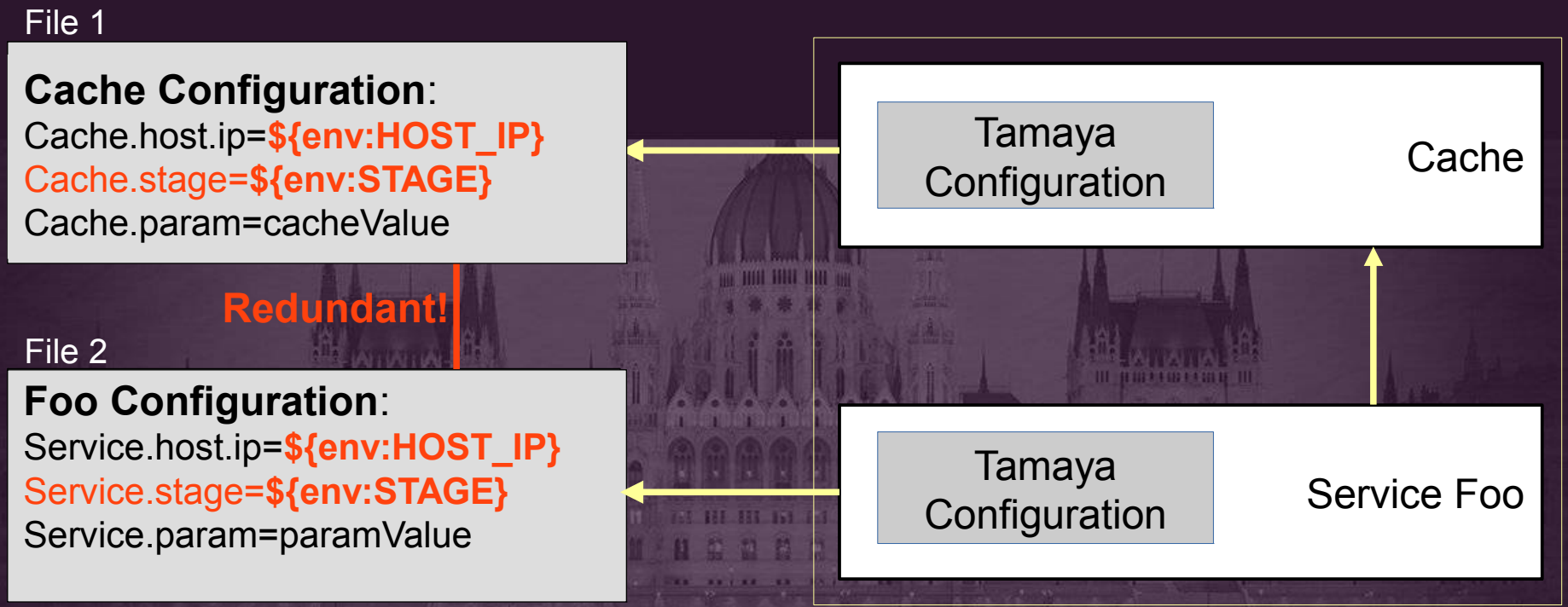
Components:
- Behaviour and state
- Transitive Deps
- Scopes
- Code
- Dynamic
- Not serializalbe

Use Cases

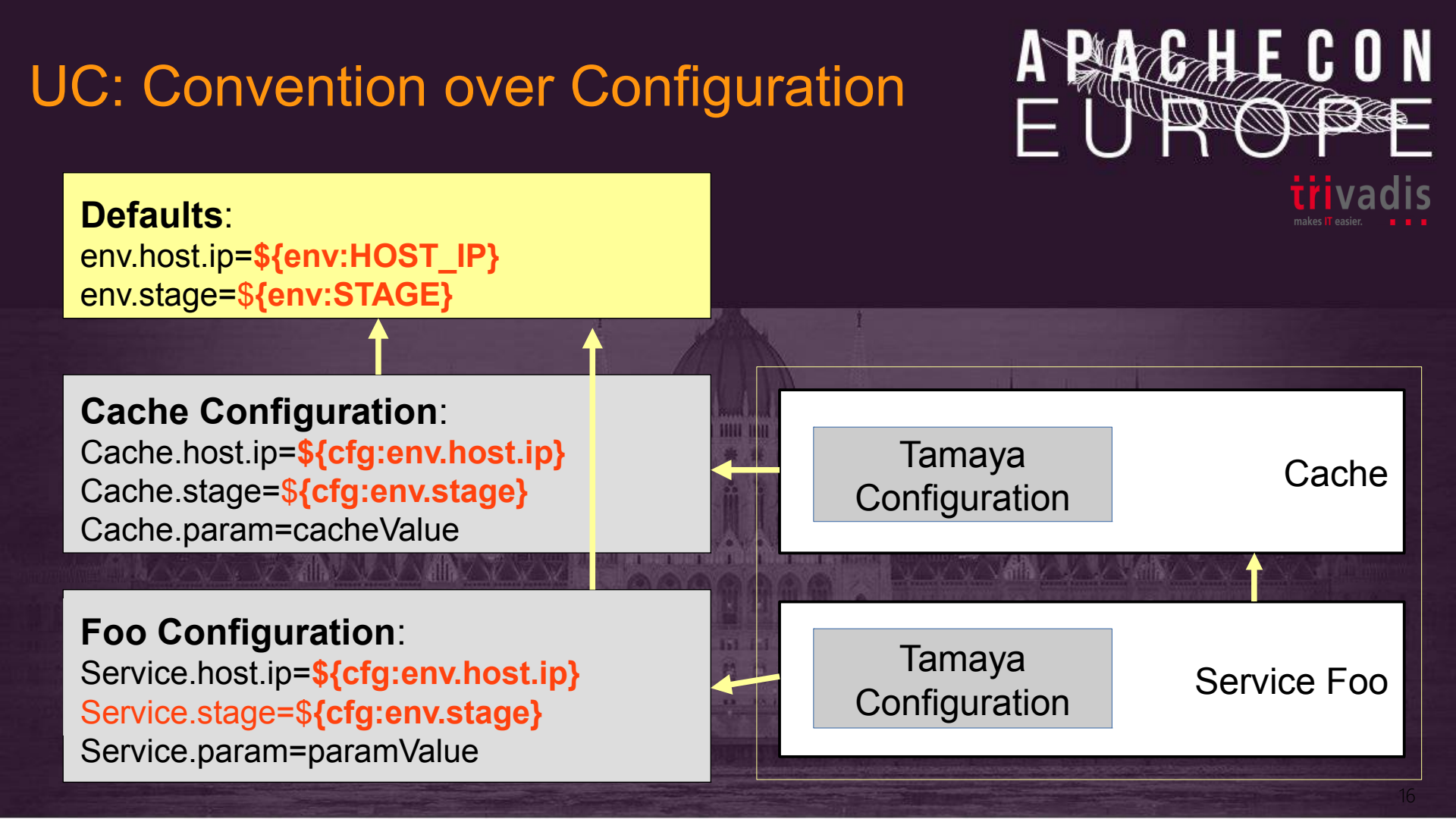
UC: Access Configuration Similarly



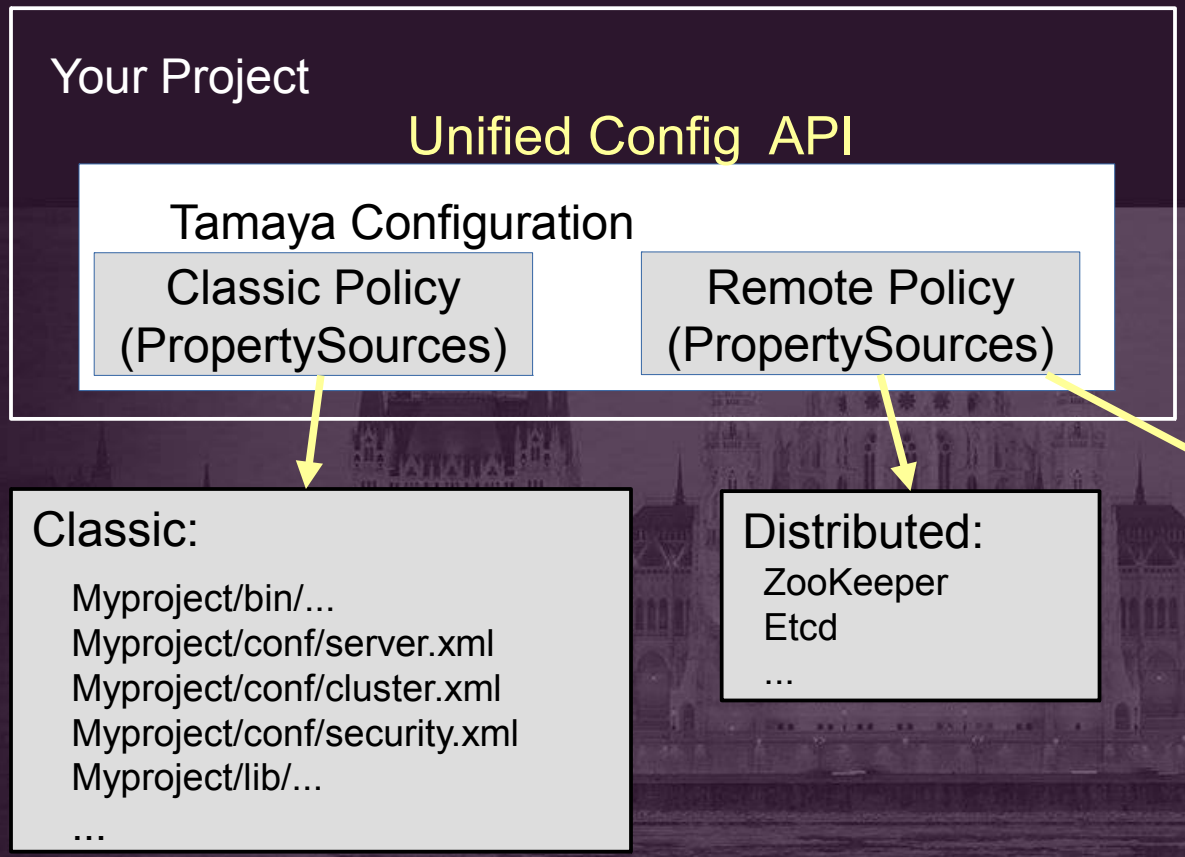
UC: Reduce Redundancy



UC: Convention over Configuration

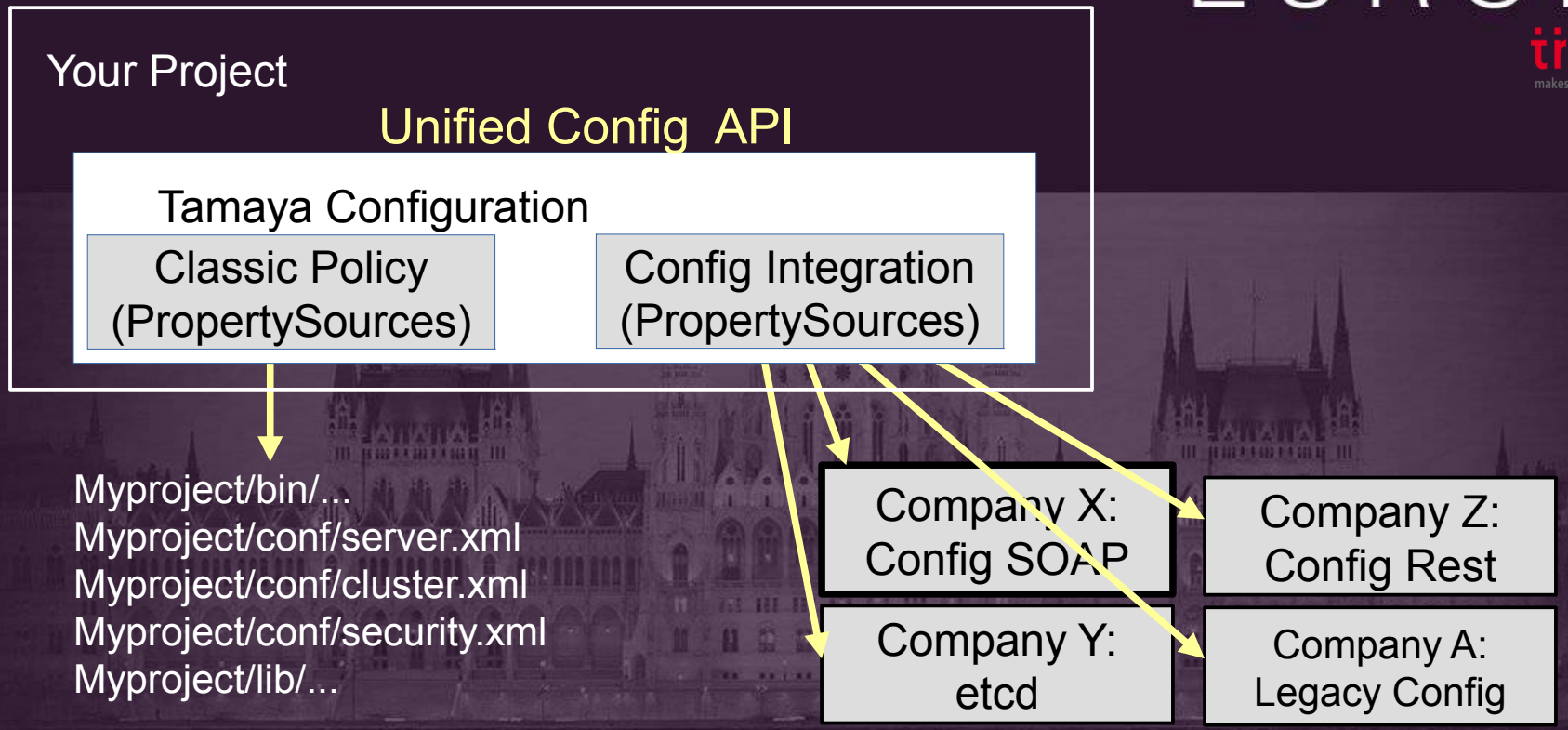


UC: Pluggable Config Backends



- Unified API for configuration access
- Policies can be provided as jar-artifacts separately
- Additional benefits: config documentation

UC: Enterprise Integration



UC: Enforceable Policies

Define, implement and distribute Company Wide Configuration Policy

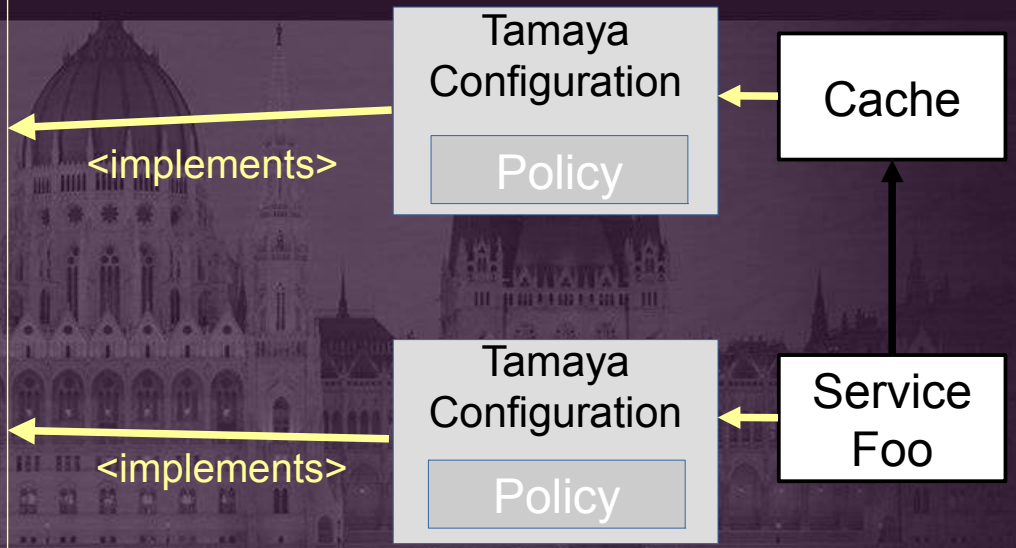
Config Policy

Locations:

- `file:${filedir}`
- `classpath:META-INF/defaults/configuration.*`
- `classpath:META-INF/${STAGE}/configuration.*`
- `url:http://myserver.int/cfg`
- System properties
- Env. Properties

Formats:

- properties
- xml
- json
- yaml



Summary: Why we need Tamaya?

- Reinventing the wheel is daily business, leading to
  - Higher Efforts in Enterprise Integration
  - Configuration Redundancies and Inconsistencies
  - Unclear or Complex Handling when remote Configuration should be supported as well
- Make Projects Integration Raedy with Company Infrastructure
- Make your Configuration Backends Pluggable

The API

APACHECON EUROPE

Let's start simple!

- Add dependency

```
org.apache.tamaya:core:1.0-incubating
```

- Add Config to `META-INF/javaconfiguration.properties`
- Use it!

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

String name = config.getOrDefault("name", "John");
int ChildNum = config.get("childNum", int.class);
```

ConfigurationProvider

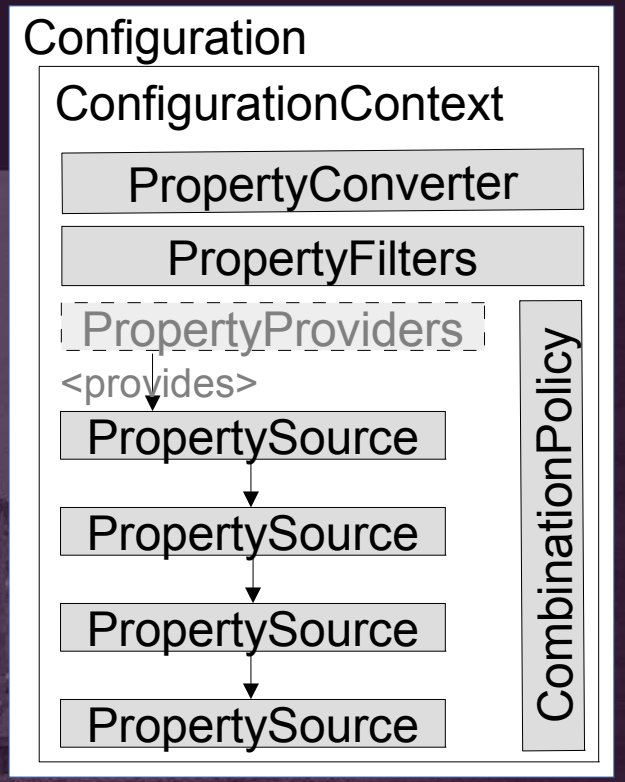
```
public class ConfigurationProvider{

  public static Configuration getConfiguration();
  public static ConfigurationContext getConfigurationContext();

  public static ConfigurationContextBuilder
                                  getConfigurationContextBuilder()

  public static void setConfigurationContext(
                                  ConfigurationContext context);
}
```

Configuration

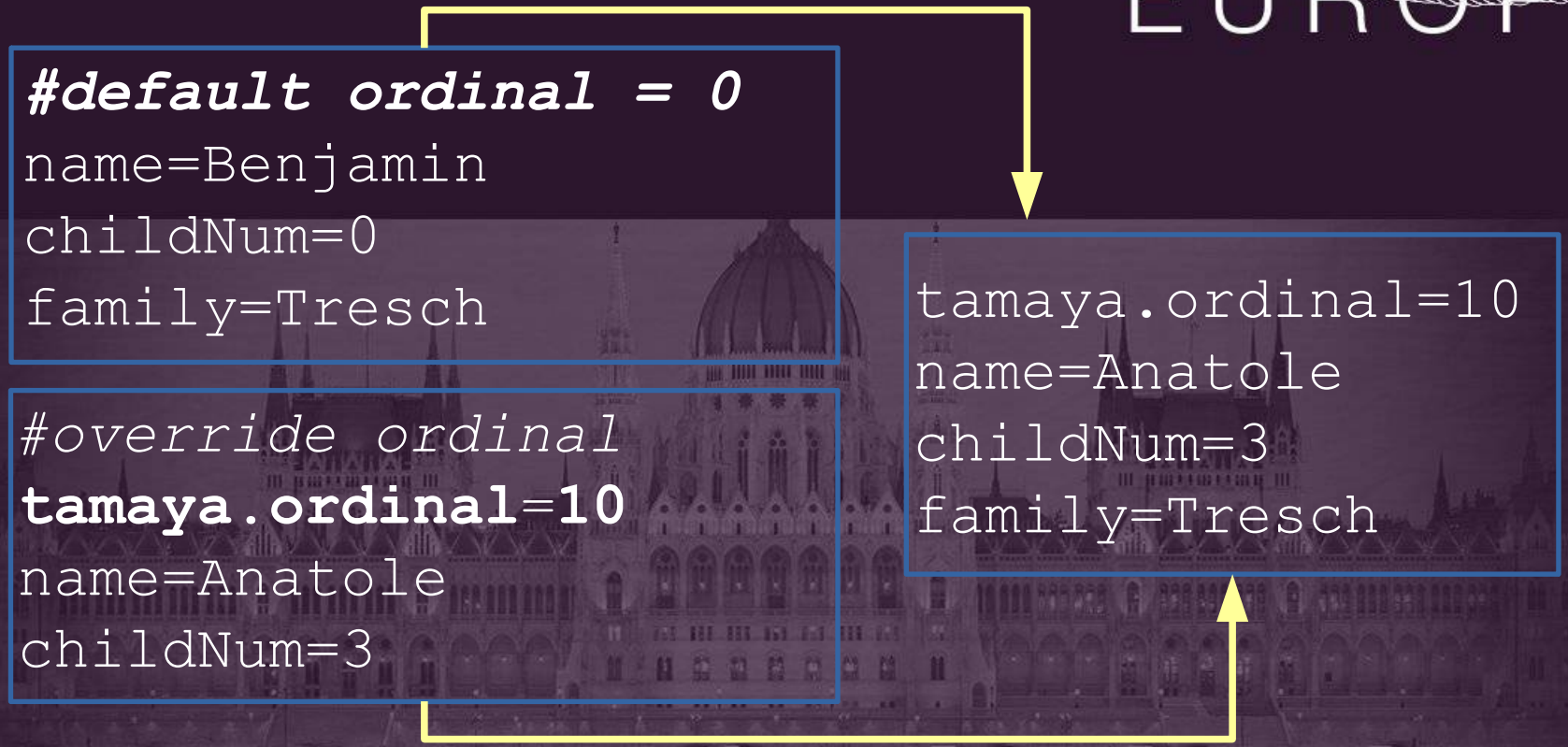
```
public interface Configuration{

  String get(String key);
  String getOrDefault(String key, String defaultValue);

  <T> T get(String key, Class<T> type);
  <T> T get(String key, TypeLiteral<T> type);
  <T> T getOrDefault(String key, Class<T> type, T defaultValue);
  <T> T getOrDefault(String key, TypeLiteral<T> type, T defaultValue);

  Map<String,String> getProperties();

  // Functional Extension Points
  Configuration with(ConfigOperator operator):
  <T> T query(ConfigQuery<T> query);
}
```

Core Concepts

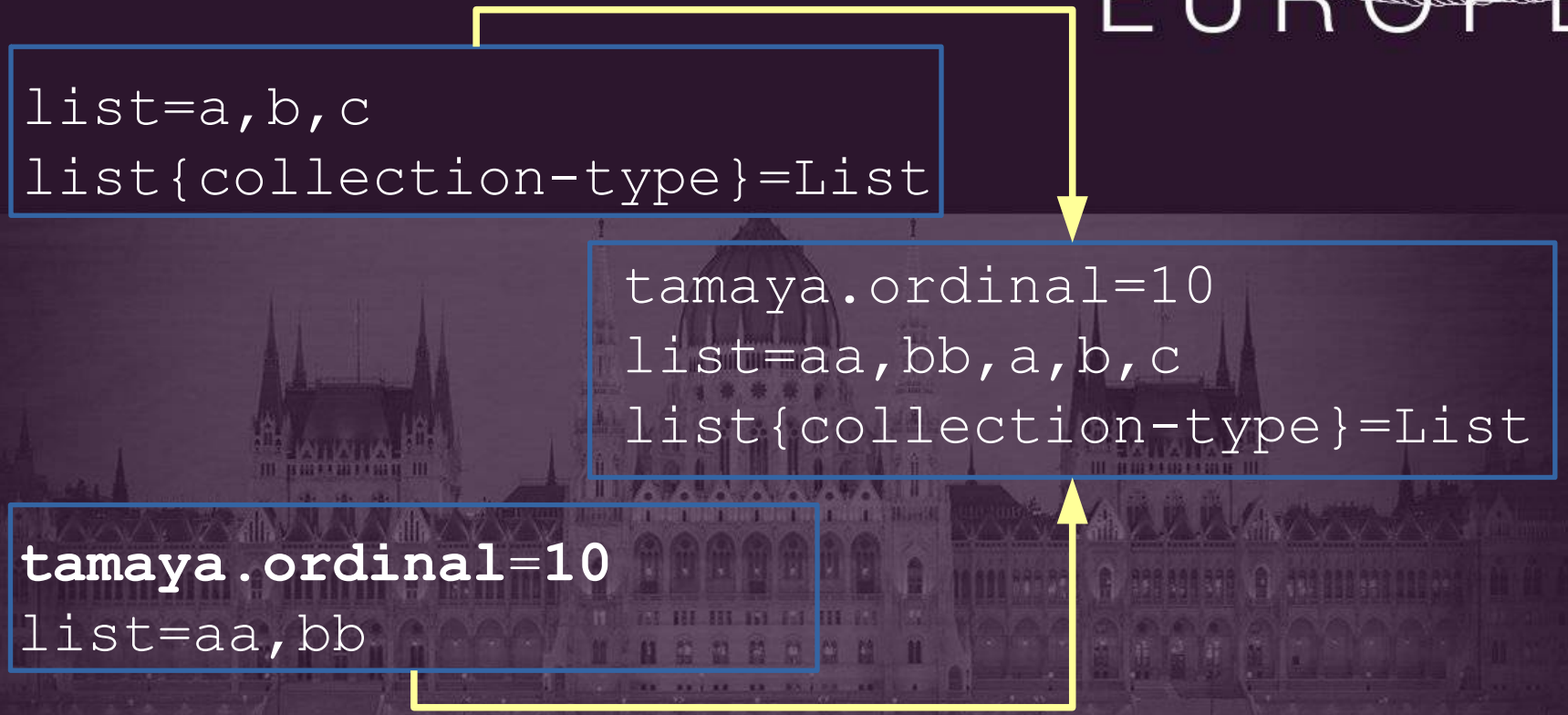
- **`Configuration`** provides properties
- built up by an ordered list of **`PropertySources`**
- **SPI**
  - `PropertyConverter`
  - `CombinationPolicy`
  - `PropertyFilter`
- **Extensions** (discussed later)



Overriding Basics



CombinationPolicy



SPI: PropertySource PropertySourceProvider


```
public interface PropertySource {

  static final String TAMAYA_ORDINAL = "tamaya.ordinal";

  String getName();
  int getOrdinal();
  String get(String key);
  Map<String, String> getProperties();
}

public interface PropertySourceProvider {
  Collection<PropertySource> getPropertySources();
}
```

Extension Modules

APACHECON EUROPE

Available Plugins

- *Tamaya-inject:* Configuration Injection and Templates
- *Tamaya-resolver:* Expression resolution, placeholders, dynamic values
- *Tamaya-resources:* Ant styled resource resolution
- *Tamaya-format*: Abstraction of a format, separating parsing from semantic mapping to configuration
- *Format Extensions:* yaml*, json, ini, ...
- *Tamaya-spring*: Integration with Spring
- *Tamaya-classloader-support:* Managing Tamaya Services within Classloading Hierarchies
- *Tamaya-cdi:* Integration with CDI
- *Tamaya-server:* REST/JSON Configuration Server
- *Tamaya-remote*: Client PropertySource matching server component
- *Tamaya-docs**: Configuration Documentation

* work in progress

Planned Features

- *Java EE:* Configuring EE, where possible
- Karaf Integration
- Source Integrations:
  - Commons-config
  - Etcd
  - Zookeeper
  - ...
- Runtime Integrations:
  - ???

Configuration Injection


```
@ConfiguredType(defaultSections="com.mycomp.tenantAdress")
public final class MyTenant

  private String name;

  @ConfiguredProperty
  @DefaultValue("2000")
  private long customerId;

  @ConfiguredProperty(keys
   "privateAddress","busin
  })
  private String address;
  ...
}
```

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

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

Configuration Template


```
@ConfiguredType(defaultSections="com.mycomp.tenantAdress")
public interface MyTenant{

  public String get

  @ConfiguredProper
  @DefaultValue("20
  public long getCustomerId();

  @ConfiguredProperty(keys={
   "privateAddress","businessAdress","[my.qualified.adress]"
  })
  public String getAddress();

}
```

```
MyTenant t =
ConfigurationInjection
      .getConfigurationInjector()
      .createTemplate(MyTenant.class);
```

Demo

Summary

Apache Tamaya Provides

- A Complete API based on Java SE 7, compatible with SE 6
- String key/value based thread-safe Configuration Model
- Support for Type-Safe Configuration Value Conversion
- Functional extension points
- Simple Filtering and Overrides
- Small footprint
- Extendible with plugins

We need help

Apache Tamaya is great, but we need you...

- … using it!
- … asking for features!
- … envangelizing it!
- … loving it!
- … extending it!
- ...

Links

- 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!

Anatole Tresch
Trivadis AG
Principal Consultant
Twitter/Google+: @atsticks
anatole@apache.org
anatole.tresch@trivadis.com