JSR 382: Config API

Spec leads: Emily Jiang
Mark Struberg

JSR-382: Config JSR

- API, Spec, TCK Repo: https://github.com/eclipse/ConfigJSR
  - Based on microprofile-config (https://github.com/eclipse/microprofile-config/)
  - Renamed the package from org.eclipse.microprofile to javax.config
  - License: Apache License v2
  - Issue tracker: https://github.com/eclipse/ConfigJSR/issues
- Expert Group
  - 15 JCP EG
    - Eclipse Foundation (Emily Jiang, Mark Struberg),
    - IBM (Emily Jiang), Red Hat (Jeff Mesnil), Tomitribe (David Blevins, Jean-Louis Monteiro),
    - Fujitsu (Kenji Kazumura), Canoo AG (Hendrik Ebbers), Sparta Systems (John Ament),
    - Trivadis AG (Anatole Tresch), Oracle (Dmitry Kornilov, Tomas Langer),
    - Individuals: Sebastian Daschner, Tilen Faganel, Werner Keil, Ivar Grimstad
  - 3 contributors: Lilian Beniot, Hiromi Takahashi (Mitsubishi UFJ Information Technology, Ltd), Jeyvison Nascimento

JSR 382: Config API

- Formal communications
  - Gitter chat: https://gitter.im/eclipse/ConfigJSR
  - Weekly calls (webex meeting) on Thursdays 3-4pm UK time
  - Mailing list:
    - Expert Group mailing list: configJsr-experts@eclipse.org
    - Discussion mailing list: configjsr-discuss@eclipse.org

JSR382: Config API 1.0

- Why?
  - Configure applications without repacking them; write once run/config everywhere
- How?
  - Specify the configuration in configure sources, such as environment variables, system properties, javaconfig.properties, other custom config sources
  - The config sources have associated ordinal to denote its ranking
  - Access configuration via
    - Programmatically lookup

```
Config config = ConfigProvider.getConfig();
config.getValue("myProp", String.class);
```

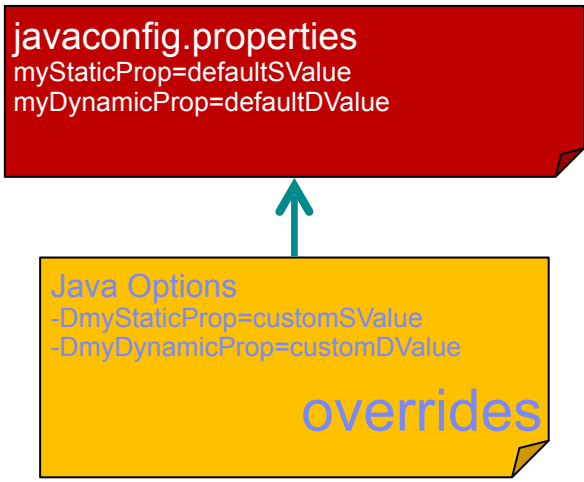
    - Via CDI Injection

```
@Inject @ConfigProperty(name="my.string.property") String myPropV;
```

JSR 382: Config API

- Static Config

```
@Inject
@ConfigProperty(name="myStaticProp")
private String staticProp;
```



- Dynamic Config

```
@Inject
@ConfigProperty(name="myDynamicProp")
private Provider<String> dynamicProp;
```

JSR 382: Config API

- Optional Config

```
Config config = ConfigProvider.getConfig();
config.getOptionalValue("myProp",
String.class).orElse("myDefault");
```

```
@Inject
@ConfigProperty(name="myStaticProp", defaultValue="blah")
private String staticProp;
```

```
@Inject
@ConfigProperty(name="myDynamicProp")
private Optional<String> dynamicProp;
```

JSR 382: Config API

- Array support

myPets=dog,cat,dog\\,cat specified in a config source

```
private String[] myPets = config.getValue("myPets", String[].class);
```

```
@Inject @ConfigProperty(name="myPets") private String[] myArrayPets;
@Inject @ConfigProperty(name="myPets") private List<String> myListPets;
@Inject @ConfigProperty(name="myPets") private Set<String> mySetPets;
```

- Upcoming features (https://github.com/eclipse/ConfigJSR/milestone/1)
  - Dynamic config sources - pushing vs. polling model
  - Define a mapping rule from config property name to environment variables
  - Support common sense converters