

Explore the New Jakarta EE Core Profile



Topics

- Jakarta EE Core Profile
- CDI 4.0 (Lite)
- JAX-RS SE Bootstrap API
- Modularity
- Some runtimes





Rudy De Busscher

- **Jakarta EE Expert**
 - Owner of Atbash
- **Involved in**
 - Committer in Eclipse EE4J groups (Jakarta EE)
 - Committer of MicroProfile
 - Java EE Security API Expert group member



@rdebusscher



<https://www.atbash.be>



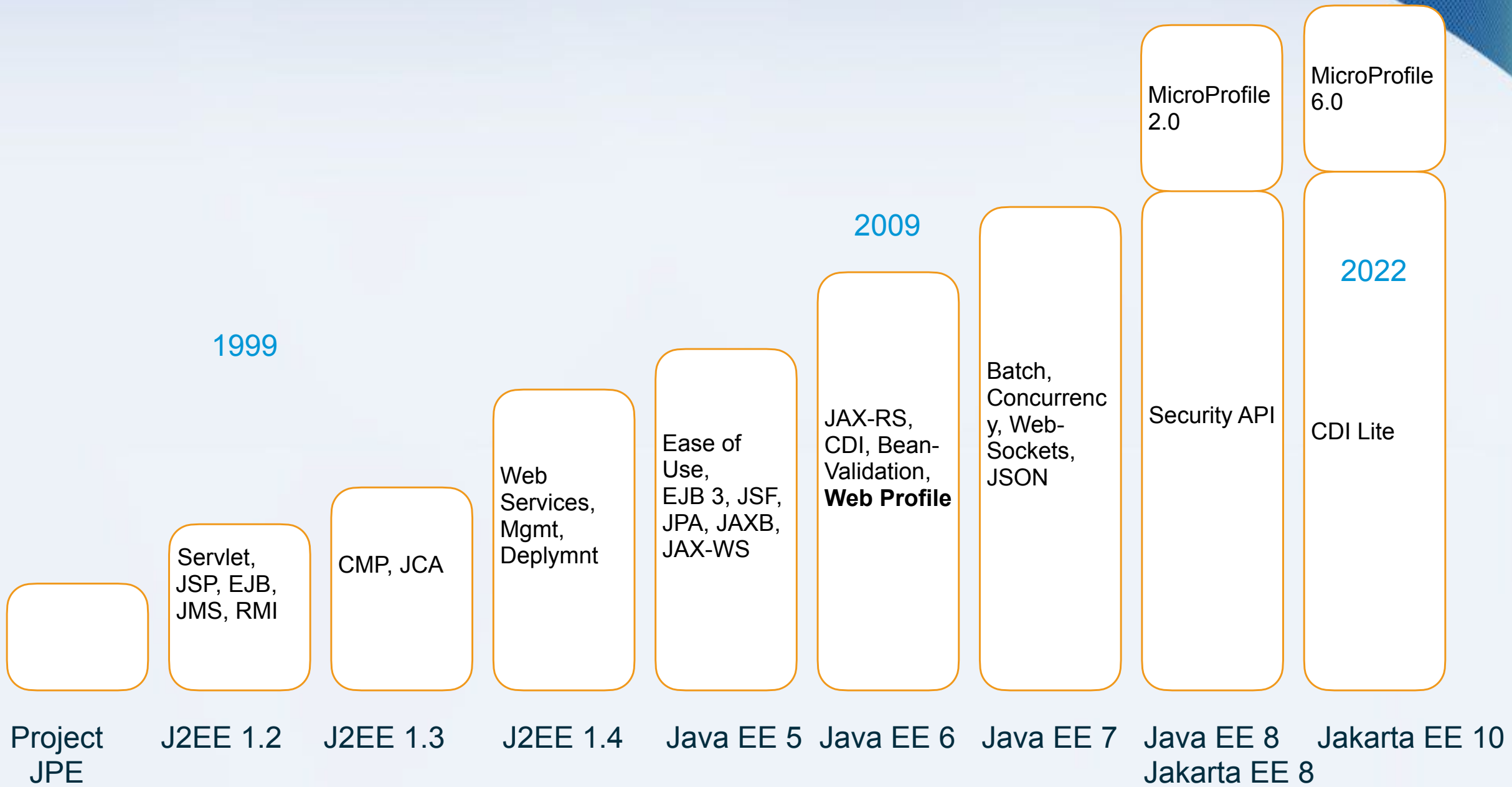
@rdebusscher/mastodon.online



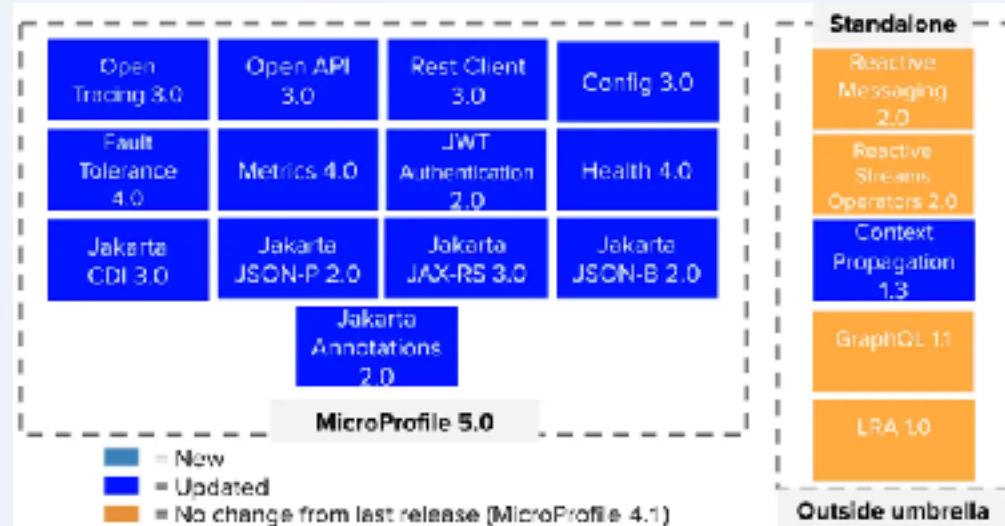
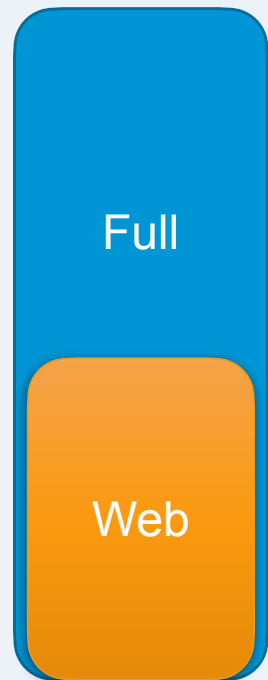
Core Profile

- Why a new Profile?
- What are the use cases

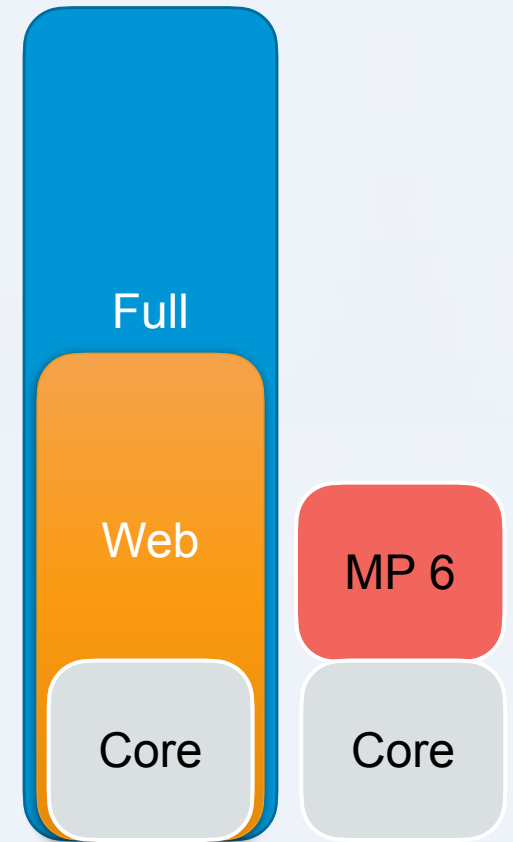




Java EE 6



Jakarta EE 10



Core Profile goals

- **Core Profile**
- Modern cloud applications
- MicroServices
- Ahead-of-time-compilation
- Modularisation

From spec document



Jakarta EE 10 Web Profile



	Persistence 3.1	RESTful Web Services 3.1
Authentication 3.0	Server Pages 3.1	JSON Processing 2.1
Concurrency 3.0	WebSocket 2.1	JSON Binding 3.0
CDI 4.0	Bean Validation 3.0	Annotations 2.1
Faces 4.0	Debugging Support 2.0	Interceptors 2.1
Security 3.0	Enterprise Beans Lite 4.0	Expression Language 5.0
Servlet 6.0	Managed Beans 2.0	Dependency Injection 2.0
Standard Tag Libraries 3.0	Transactions 2.0	CDI Lite 4.0

Updated

Not Updated

New

Jakarta EE 10 Core Profile



Servlet

??

RESTful Web Services 3.1

JSON Processing 2.1

JSON Binding 3.0

Annotations 2.1

Interceptors 2.1

Expression Language 5.0

(Optional)

Dependency Injection 2.0

CDI Lite 4.0

Config

??




Updated

Not Updated

New

Compatible Products

Jakarta EE 10 Core Profile Compatible Products

Product		Certification Results
	Open Liberty	22.0.0.10-beta, Java 17 22.0.0.10-beta, Java 11
	Payara Server Community	6.2022.1.Alpha3
	WildFly	27.0.0.Alpha5, Java SE 17 27.0.0.Alpha5, Java SE 11 Preview 27.0.0.Alpha1, Java SE 17 Preview 27.0.0.Alpha1, Java SE 11

Web

Core



CDI Lite - Why?

- CDI
 - Determines beans at startup
 - Scanning - Programmatic (Extension)
- Fast startup
 - Much can be done upfront
 - ~~Native compilation~~




CDI Lite

- Largest part of CDI
 - Upfront execution possible and less reflection at runtime
- Does not contain
 - BeanManager -> BeanContainer
 - Extension -> BuildCompatibleExtension
 - Decorator / Specializes
 - SessionScoped / ConversationScoped
- Optional



BeanManager

- `CDI.current().getBeanManager();`
- `CDI.current().getBeanContainer();`

```
 public interface BeanManager extends BeanContainer {
```



Expression Language

- Within *BeanManager* interface
 - *public ExpressionFactory wrapExpressionFactory(ExpressionFactory expressionFactory);*
 - *public ELResolver getELResolver();*
- To resolve CDI beans from EL expressions
- Requires to bring in Expression Language spec in Core Profile as optional.



BuildCompatibleExtension

- Works in a similar way as Extension (CDI full)
- Prepares metadata about beans at compile time (possible)
- Faster start up (since less work to do)



Build Phases

1. Discovery

- Add annotations that will also be discovered
- Use annotation that isn't a qualifier.

2. Enhancement

- Allow transforming of annotations
- For ex: Assume `@Inject` for each qualifier usage.



Build Phases

3. Registration

- Get access to defined beans
- Called a second time for all Synthetic beans after Synthesis phase.

4. Synthesis

- Define beans programmatically



Build Phases

5. Validation

- Custom validation of CDI Bean metadata
- After default validation



CDI Implementations

- Weld
 - CDI Full
- OpenDI
 - CDI Lite
 - In progress
 - Based on Micronaut
 - <https://github.com/eclipse-ee4j/odi>



CDI Lite Problem

- 1 artifact (1 API jar) for entire CDI
- Includes also CDI full only classes
 - These are optionally supported in Core Profile.
- Compile against API jar
- Fails at runtime
 - Potentially
 - When I use CDI full only classes and runtime doesn't support it.



Core Profile tester



- Am I using CDI full classes?

```
<dependency>
  <groupId>be.atbash.jakarta</groupId>
  <artifactId>core-profile-tester</artifactId>
  <version>1.0.0</version>
  <scope>test</scope>
</dependency>
```

```
@Test
void testCompliance() {
    TestCoreProfile tester = new TestCoreProfile(packageName: "be.atbash");
    Assertions.assertThat(tester.violations()).isEmpty();
}
```

JAX-RS new features

- Support for multipart media type
- Automatic loading of Provider extensions
- Better alignment with JSON-B
- Java SE Bootstrap API



Java SE Bootstrap API

- Run JAX-RS implementation directly on Java SE
- No need for WAR deployment
- Also CDI can be started with pure SE
 - Since CDI 2.0 / Java EE 8





The background of the slide features a photograph of a silver laptop with a black keyboard and a light blue ceramic mug filled with tea, both resting on a dark, textured wooden surface. A white diamond-shaped graphic is overlaid on the right side of the image, containing the word "Demo" in orange text.

Demo

Modular

- Part of goal of Core Profile
- Make runtime as small as possible
- Load only what is really needed
 - Helps in faster startup*

WildFly - Galleon

- Galleon : Provisioning tool designed to create and maintain software distributions
- Define modules that are loaded/started
- Binaries include always everything
- *`./galleon.sh install wildfly/final#27.0.0.Final --layers=jaxrs,cdi,jsonb,jsonp --dir=<target>`*
 - *Adds also Undertow (Servlet), Bean Validation, SmallRye Config (MP Config)*



OpenLiberty



- OSGI based
- Most modular Jakarta runtime
- Defines features in *server.xml*
- (Jakarta EE 10 = Beta = not modular yet)

```
<featureManager>  
  <feature>cdi-4.0</feature>  
  <feature>restfulWS-3.1</feature>  
  <feature>jsonb-3.0</feature>  
</featureManager>
```



Other runtimes

- Besides the certified runtimes
 - Not certified ones (yet)
 - In progress
 - Might create a Core Profile version



Helidon

- Helidon MP
 - MicroProfile specs ~ Core profile
- JDK 17+
- Native compilation available
- Modular
 - Core artifact
- Status
 - 3.0.x -> Jakarta EE 9.1



Quarkus

- Quarkus 3.0 development started
- Support for Jakarta namespace
- JDK 11+
- Native compilation available
- Modular



Micronaut

- Micronaut 3.x
 - javax namespace
- Not all specs supported
- JDK 11+
- Native compilation available
- Modular



Piranha Cloud

- Version 22.0.9 +
- Core Profile runtime
- JDK 17+
- No Native compilation available by default
- Single artefact
 - Servlet (partial), EL Expression, Bean Validation
 - But is small



Atbash Runtime

- 1.0.0.alpha
- Core Profile runtime
- JDK 11+
- No Native compilation available by default
- Modular



Atbash Runtime

- As runtime
 - *java -jar atbash_runtime.jar <path/to/app.war>*
 - Server mode - Remote deployment
- As Runner
 - *public static void main*
 - Using same core and modules (service loader)
- Extensions for MicroProfile Config, JWT Auth, ...



Core Profile - Conclusion

- Typical for today's REST based applications
- Should be small, modular and support native compilation
- Easy to create your runtime
- Near future 5 - 10 supporting products
- Challenges
 - Missing critical elements (Config, Security, Data access)
 - Potential CDI Lite pitfalls



Code Repository

- <https://github.com/atbashEE/core-profile-demo>
- Core profile tester source code
- BCE @Enhancement example
- BCE @Synthesis example
- Running Jakarta Core application on Java SE
- Example with OpenLiberty



Q & A



Thank you



- Atbash
- Blog
 - <https://www.atbash.be>
- Github
 - <https://github.com/atbashEE>

