Desarrollo de micro servicios cloud-native con Quarkus

JoeDayz | 2025

Administrando la configuración de la aplicación en Microservicios

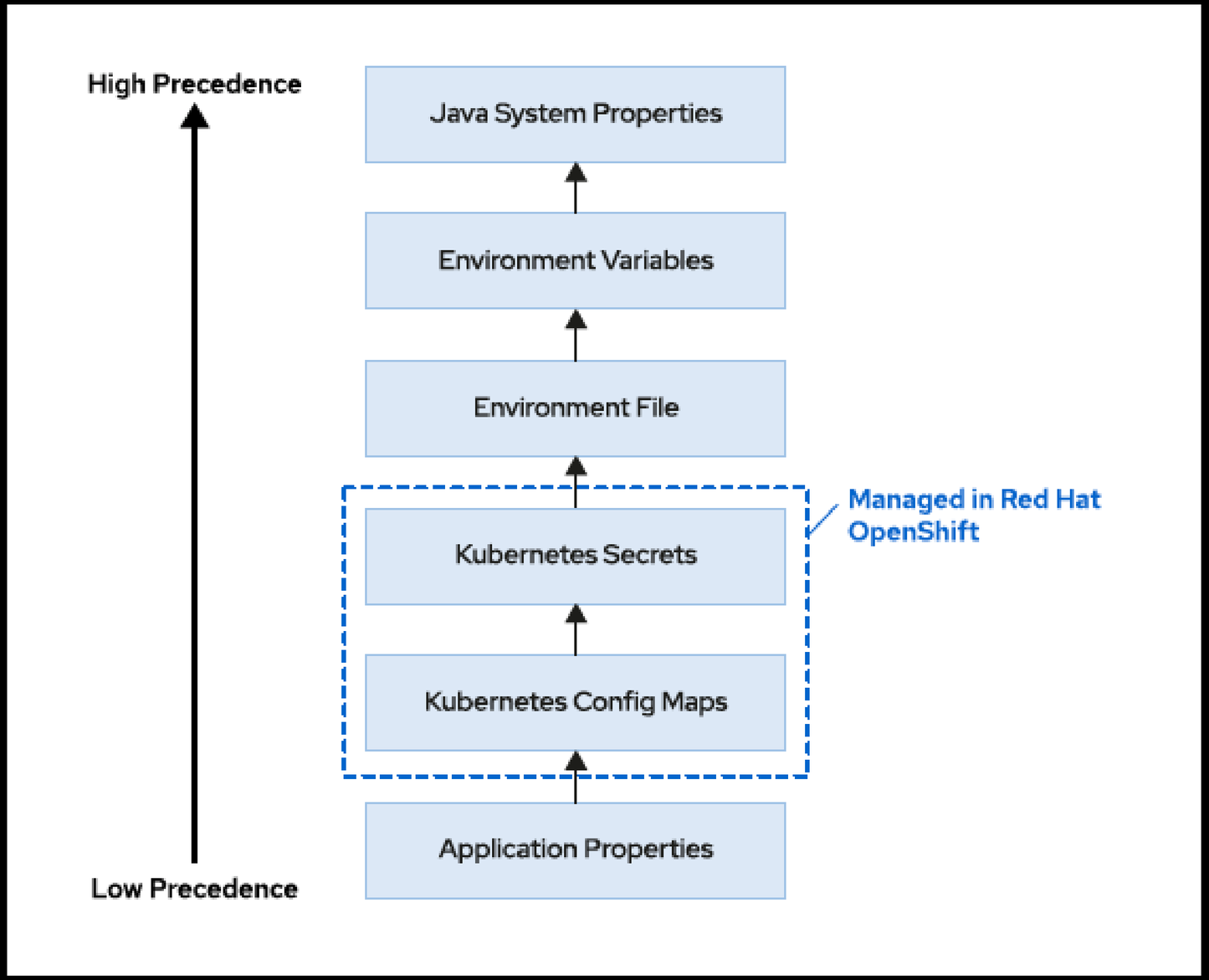
MicroProfile 7.0

MicroProfile 7.0 is a major release that introduces significant changes, notably replacing MicroProfile Metrics with the more comprehensive MicroProfile Telemetry 2.0. Additionally, it specifies Jakarta EE 10 Core Profile as a dependency, simplifying integration. This release also includes updates to several specifications, and allows certification on Java SE 11 and higher.

MicroProfile 7.0 Presentation

SmallRye Config Library

Orden de precedencias en orígenes de configuración



Src/main/resources → application

Src/test/resources → Tests

Desarrollando microservicios basados en REST

Especificaciones

JakartaEE y Microprofile

- ArC -> Jakarta Context and Dependency Injection
- RESTEasy -> JAX-RS
- Jackson, un framework que mapea objetos Java hacia y desde JSON

JakartaEE / MicroProfile	Spring
ArC (CDI)	Spring DI (IoC Container, @Component, @Autowired, etc.)
RESTEasy (JAX-RS)	Spring Web MVC (Spring REST Controllers)
Jackson	Jackson (ya integrado en Spring Boot)

Common Class-level Lifecycle Annotations

Annotation	Scope
`@ApplicationScoped`	ArC instantiates the object one time and reuses the instance for all injection points.
`@RequestScoped`	ArC instantiates a new object for each HTTP request.
`@SessionScoped`	When you use the `quarkus-undertow` extension, the `javax.servlet.http.HttpSession` object manages the lifespan of a session-scoped bean.

Annotation	Scope
`@Dependent`	A dependent scope binds a property object instance to its parent Java object. Arc provides a new instance to each parent object, and destroys the instance with the parent object.

Comparativa con Spring

CDI / ArC	Spring Framework	Descripción en Spring
`@ApplicationScoped`	`@Scope("singleton")` (por defecto en Spring)	Una sola instancia por aplicación (contenedor IoC). Todas las inyecciones comparten la misma instancia.
`@RequestScoped`	`@RequestScope`	Una nueva instancia por cada petición HTTP.
`@SessionScoped`	`@SessionScope`	Una instancia por sesión HTTP, mientras dure la sesión del usuario.
`@Dependent`	`@Scope("prototype")`	Una nueva instancia **cada vez que se inyecta** o se solicita explícitamente el bean.

```java
// ApplicationScoped (singleton)
@Component
@Scope("singleton")  // en realidad, no es necesario, porque es el valor por defe
public class AppBean { }

// RequestScoped
@Component
@RequestScope
public class RequestBean { }

// SessionScoped
@Component
@SessionScope
public class SessionBean { }

// Dependent -> Prototype
@Component
@Scope("prototype")
public class PrototypeBean { }
```

Descubrimiento de Beans en la Inyección de Dependencias

Clases descubiertas por el ArC dependency injection

1. Clases Application que son anotadas con anotaciones como @ApplicationScoped
2. Dependencias en el archivo beans.xml
3. Clases referenciadas en el indice Jandex
4. Clases referenciadas por el quarkus.index-dependency property en el application.properties

Limitaciones ArC

1. Java Reflection. No se recomienda usar campos privados.

```
@ApplicationScoped
public class GreeterBean {

  @Inject
  GreeterService greeter;
}
```

Es mejor usar inyección por constructor:

```
@ApplicationScoped
public class GreeterBean {

  private GreeterService greeter;

  public GreeterBean(GreeterService greeter) {
    this.greeter = greeter;
  }
}
```

Limitaciones ArC

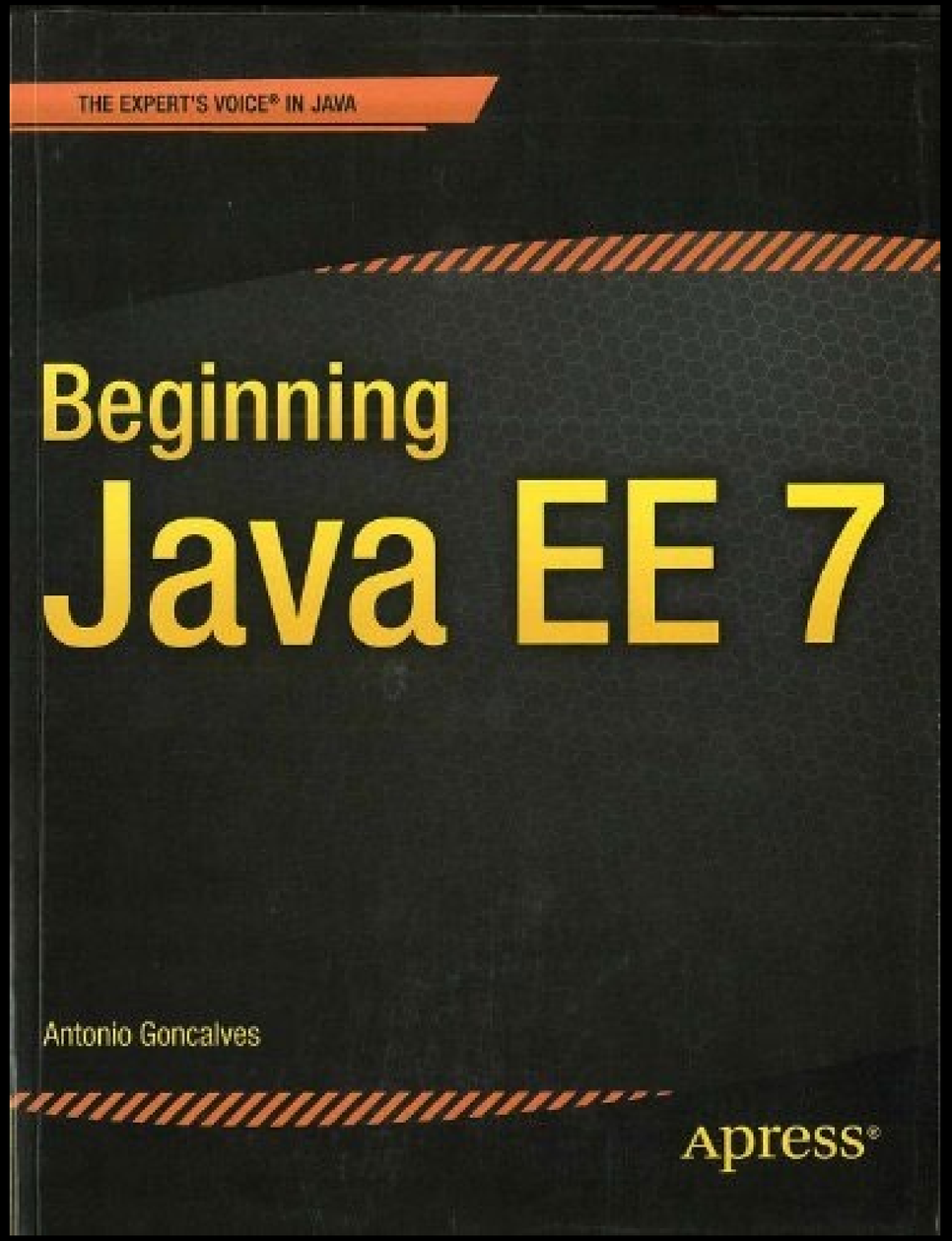
2. beans.xml

Arc ignora el contenido del descriptor en el archivo beans.xml, y no soporta configurar interceptors, decorators, o alternatives usando el archivo beans.xml

3. BeanManager

Esta clase implementa los siguientes métodos: getBeans(), createCreationalContext(), getReference(), getInjectableReference(), resolve(), getContext(), fireEvent(), getEvent(), y createInstance().

4. Quarkus no soporta @ConversationalScoped y @Interceptors, extensiones portables, especialización, passivization y passivizing scopes, y métodos interceptors en super clases.


THE EXPERT'S VOICE® IN JAVA
Beginning
Java EE 7
Antonio Goncalves
Apress®

Implementando REST APIs con JAX-RS

JAX-RS Class and Method-level Annotations Summary

Annotation	Description
`@Path`	A relative request path for a class or method.
`@GET`, `@PUT`, `@POST`, `@DELETE`, `@HEAD`	An endpoint HTTP verb.
`@Produces`, `@Consumes`	The Internet media types for the content consumed as the request parameters or produced as the response. The `javax.ws.rs.core.MediaType` class specifies the allowed values.

JAX-RS	Spring MVC / Spring Web	Descripción
`@Path("/resource")`	`@RequestMapping("/resource")` (a nivel de clase o método)	Define la ruta base o relativa del endpoint.
`@GET`	`@GetMapping` ó `@RequestMapping(method = GET)`	Endpoint HTTP GET.
`@POST`	`@PostMapping` ó `@RequestMapping(method = POST)`	Endpoint HTTP POST.
`@PUT`	`@PutMapping` ó `@RequestMapping(method = PUT)`	Endpoint HTTP PUT.
`@DELETE`	`@DeleteMapping` ó `@RequestMapping(method = DELETE)`	Endpoint HTTP DELETE.
`@HEAD`	`@RequestMapping(method = HEAD)`	Endpoint HTTP HEAD.
`@Produces("application/json")`	`@GetMapping(produces = "application/json")` ó `@RequestMapping(produces = "application/json")`	Indica el tipo de respuesta (content-type).
`@Consumes("application/json")`	`@PostMapping(consumes = "application/json")` ó `@RequestMapping(consumes = "application/json")`	Indica el tipo de request body aceptado.

JAX-RS Method Parameter-level Annotations Summary

Annotation	Description
`@PathParam`	Retrieves a method parameter from a segment of the URI.
`@QueryParam`	Retrieves a method parameter from an HTTP query parameter by using the query parameter name.
`@HeaderParam`	Retrieves a method parameter from a header in the HTTP request by using the header name.
`@FormParam`	Retrieves a method parameter from a form field by using the form field name.

Annotation	Description
`@Context`	Returns the context of the `HttpServletRequest` object or the `SecurityContext` object for the incoming HTTP request. Additionally, you can use this annotation to inject class-level variables and the `UriInfo` for incoming requests.

JAX-RS	Spring Framework	Descripción
`@PathParam("id")`	`@PathVariable("id")`	Obtiene un valor desde un segmento de la URI.
`@QueryParam("q")`	`@RequestParam("q")`	Obtiene un parámetro de query (`?q=valor`).
`@HeaderParam("h")`	`@RequestHeader("h")`	Obtiene un valor desde un header HTTP.
`@FormParam("f")`	`@RequestParam("f")` (para `application/x-www-form-urlencoded`) o `@ModelAttribute`	Obtiene un campo enviado en un formulario.
`@Context`	Inyección de objetos como `HttpServletRequest` , `HttpServletResponse` , `Principal` , etc., directamente en el método o con `@Autowired` .	Permite acceder al contexto de la petición.

Gracias

www.joedayz.pe