

Desarrollo de micro servicios cloud-native con Quarkus

Administrando la configuración de la aplicación en Microservicios

MicroProfile 7.0

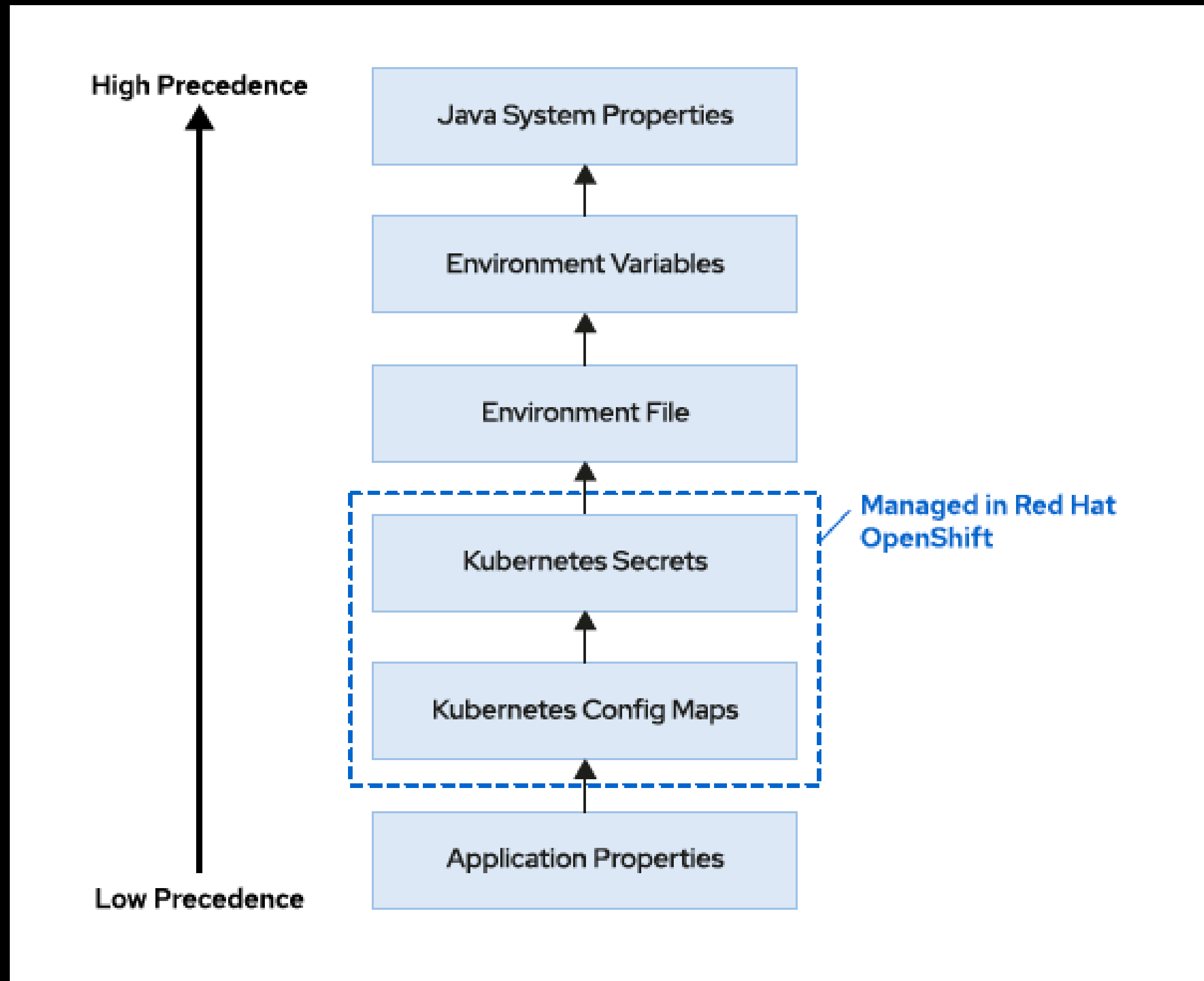
MicroProfile 7.0 is a major release that introduces significant changes, notably replacing MicroProfile Metrics with the more comprehensive MicroProfile Telemetry 2.0. Additionally, it specifies Jakarta EE 10 Core Profile as a dependency, simplifying integration. This release also includes updates to 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



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 <code>javax.servlet.http.HttpSession</code> 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.	


```
// 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 índice `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 <code>javax.ws.rs.core.MediaType</code> class specifies the allowed values.

JAX-RS	Spring MVC / Spring Web	Descripción
<code>@Path("/resource")</code>	<code>@RequestMapping("/resource")</code> (a nivel de clase o método)	Define la ruta base o relativa del endpoint.
<code>@GET</code>	<code>@GetMapping</code> ó <code>@RequestMapping(method = GET)</code>	Endpoint HTTP GET.
<code>@POST</code>	<code>@PostMapping</code> ó <code>@RequestMapping(method = POST)</code>	Endpoint HTTP POST.
<code>@PUT</code>	<code>@PutMapping</code> ó <code>@RequestMapping(method = PUT)</code>	Endpoint HTTP PUT.
<code>@DELETE</code>	<code>@DeleteMapping</code> ó <code>@RequestMapping(method = DELETE)</code>	Endpoint HTTP DELETE.
<code>@HEAD</code>	<code>@RequestMapping(method = HEAD)</code>	Endpoint HTTP HEAD.
<code>@Produces("application/json")</code>	<code>@GetMapping(produces = "application/json")</code> ó <code>@RequestMapping(produces = "application/json")</code>	Indica el tipo de respuesta (content-type).
<code>@Consumes("application/json")</code>	<code>@PostMapping(consumes = "application/json")</code> ó <code>@RequestMapping(consumes = "application/json")</code>	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	<p>Returns the context of the <code>HttpServletRequest</code> object or the <code>SecurityContext</code> object for the incoming HTTP request.</p> <p>Additionally, you can use this annotation to inject class-level variables and the <code>UriInfo</code> for incoming requests.</p>

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

Gracias

www.joedayz.pe