Java Spring Boot and Quarkus are popular frameworks for building Java-based microservices and applications. Both use annotations to simplify configuration and development. Here's a comparison of commonly used annotations in **Spring Boot** and **Quarkus**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Functionality** | **Spring Boot Annotation** | **Quarkus Annotation** | **Explanation** |
| **Application Entry Point** | @SpringBootApplication | @QuarkusMain (optional) | Marks the main class of the application in Spring Boot and Quarkus. Quarkus uses main() less often. |
| **REST Controller** | @RestController | @Path | In Spring Boot, @RestController handles HTTP requests; in Quarkus, @Path is used for resource paths. |
| **Mapping HTTP Requests** | @RequestMapping, @GetMapping, @PostMapping, etc. | @GET, @POST, @PUT, etc. | Spring Boot offers specific annotations for HTTP verbs, while Quarkus uses JAX-RS annotations for mapping. |
| **Dependency Injection** | @Autowired, @Qualifier | @Inject | Spring Boot uses @Autowired, and Quarkus uses the standard CDI @Inject for dependency injection. |
| **Request Parameters** | @RequestParam, @PathVariable | @QueryParam, @PathParam | Spring Boot and Quarkus use different annotations for passing query or path parameters in HTTP requests. |
| **Entity** | @Entity | @Entity | Both use JPA's @Entity for marking a class as a JPA entity. |
| **Transactional** | @Transactional | @Transactional | Both frameworks use the standard JPA @Transactional annotation to manage transactions. |
| **Configuration** | @Configuration | @ConfigProperties | Spring Boot's @Configuration is for defining beans and properties; Quarkus uses @ConfigProperties. |
| **Property Injection** | @Value("${property}") | @ConfigProperty(name="property") | Both are used for injecting configuration properties from files like application.properties. |
| **Scheduled Tasks** | @Scheduled | @Scheduled | Both frameworks use @Scheduled to define scheduled tasks. |
| **Exception Handling** | @ExceptionHandler | @ServerExceptionMapper | In Spring Boot, @ExceptionHandler is used to handle exceptions, while Quarkus uses @ServerExceptionMapper. |
| **Validation** | @Valid, @Validated | @Valid | Both use @Valid for bean validation, with Spring Boot adding @Validated for grouping validation. |
| **Cross-Origin Requests (CORS)** | @CrossOrigin | No equivalent | Spring Boot provides @CrossOrigin to enable CORS handling, which Quarkus manages via configuration. |
| **Bean Scope** | @Scope | @ApplicationScoped, @RequestScoped, etc. | Spring Boot uses @Scope, while Quarkus uses CDI annotations for defining bean scope. |
| **Component Scanning** | @ComponentScan | No equivalent | Spring Boot uses @ComponentScan for scanning components, while Quarkus discovers them automatically. |
| **Asynchronous Execution** | @Async | @Blocking, @NonBlocking | Spring Boot provides @Async for async methods, while Quarkus uses @Blocking and @NonBlocking. |

### **Key Differences:**

* **Dependency Injection:** Quarkus is built around the **CDI (Context and Dependency Injection)** specification and uses standard annotations like @Inject. Spring Boot provides more custom dependency injection mechanisms, such as @Autowired.
* **REST API:** Quarkus uses **JAX-RS** for handling RESTful APIs, whereas Spring Boot uses its own annotations.
* **Configuration:** Spring Boot relies heavily on annotation-based configurations, while Quarkus focuses on minimizing configuration and using sensible defaults, relying on build-time optimizations.This comparison highlights how both frameworks offer similar functionalities with some key differences in approach and usage of annotations.