Micronaut Cheat Sheet

AOT · DI · HTTP Client · GraalVM · Cloud

SETUP & CLI

```
mn create-app com.example.demo
--features=data-jpa,security-jwt,graalvm
./mvnw mn:run  (or gradle mnRun)
mn create-controller Hello --lang kotlin
```

All compile-time DI no Spring-like runtime scanning

MAIN APPLICATION

```
@MicronautApplication
public class Application{
public static void main(String[] a){
Micronaut.run(Application.class, a);
}
}
```

Starts in <100ms; tiny memory footprint

CONTROLLERS

```
@Controller("/api/users")
public class UserController {
@Get("/{id}")
@Produces(APPLICATION_JSON)
public HttpResponse<User>
get(@PathVariable Long id){...}
@Post @Consumes(APPLICATION_JSON)
public HttpResponse<User> create(@Body
User u){...}
}
```

DEPENDENCY INJECTION

```
@Singleton / @Prototype / @RequestScope
@Inject / constructor injection
@Requires(property="feature.x")
conditional bean
@Primary / @Secondary for disambiguation
```

DI processed at compile time by annotation processor
No reflection AOT proxy generation

CONFIGURATION

```
application.yml  /  application-prod.yml
@ConfigurationProperties("app")
public class AppConfig {
private String name;  // maps to
app.name
}
@Value("${greeting:Hello}") String
greeting;
```

Env vars override properties automatically

DATA REPOSITORIES

```
@Repository
interface UserRepo extends
CrudRepository<User,Long>{
List<User> findByEmail(String email);
@Query("SELECT * FROM users WHERE
active=true")
List<User> findActive();
}
```

Micronaut Data generates SQL at compile time

AOP & INTERCEPTORS

```
@Around
@Type(LogInterceptor.class)
public @interface Logged {}
@Singleton
@InterceptorBean(Logged.class)
class LogInterceptor implements
MethodInterceptor{
Object intercept(MethodInvocationContext
ctx){...}
}
```

DECLARATIVE HTTP CLIENT

```
@Client("http://user-service")
interface UserClient {
@Get("/users/{id}")
User getUser(@PathVariable Long id);
}
@Inject UserClient client;
```

Compile-time generated no reflection, native-compatible

COMPILE-TIME DI / AOT

Annotation processors run during compilation
DI metadata baked into bytecode (not reflection)
No runtime classpath scanning
Much faster startup than Spring
GraalVM native: mn:nativeCompile

```
./target/myapp  starts in <20ms
```

TESTING

```
@MicronautTest
class UserControllerTest {
@Inject @Client("/") HttpClient client;
@Test void testGet(){
var resp =
client.toBlocking().exchange("/users/1");
assertEquals(200,
resp.status().getCode());
}
}
```

COMMON ANNOTATIONS

```
@MicronautApplication @Controller @Get
@Post @Put @Delete
@Singleton @Inject @Value
@ConfigurationProperties
@Client @Body @PathVariable @QueryValue
@Introspected  // for reflection-less
serialization
@Requires  // conditional beans
@Validated @NotNull @Min @Max  bean
validation
```

COMMON GOTCHAS

@Introspected needed on data classes for native image
Circular deps: Micronaut fails at compile time (good!)
AOP only works on injected beans, not new'd instances
HTTP client must match server's content-type exactly
Config refresh requires restart no @RefreshScope