自定义 Spring Boot Starter 示例（Greeting）

本模块演示如何编写并消费一个自定义的 Spring Boot Starter，包括自动配置、条件装配、配置元数据、Web 端点、健康检查与 Micrometer 指标集成。项目基于 Spring Boot 2.6.7 与 JDK 8。

模块结构

- `greeting-spring-boot-starter`：Starter 模块，提供自动配置与功能实现
- `greeting-app`：演示应用，依赖 Starter 并演示其用法

关键能力

- 自动配置：`GreetingAutoConfiguration` 按条件装配所需 Bean
- 配置属性：`GreetingProperties` 绑定 `greeting.*` 前缀的配置
- 核心服务：`GreetingService.greet(name)` 生成问候语，内置错误与指标记录
- Web 端点：`/api/greeting` 与 `/api/greeting/{name}` 返回问候语
- 健康检查：`GreetingHealthIndicator` 提供基础健康检测逻辑
- 指标采集：`GreetingMetrics` 采集请求/错误计数（依赖 Micrometer）
- 配置元数据：`META-INF/spring-configuration-metadata.json` 提供 IDE 智能提示
- 自动装配注册：`META-INF/spring.factories`（Spring Boot 2.x 格式）

依赖与安装

在父 POM 下包含两个子模块：

```
<modules>
    <module>greeting-spring-boot-starter</module>
    <module>greeting-app</module>
</modules>
```

应用使用 Starter 的依赖方式：

```
<dependency>
    <groupId>com.example</groupId>
    <artifactId>greeting-spring-boot-starter</artifactId>
    <version>1.0.0-SNAPSHOT</version>
</dependency>
```

如需 Web 与 Actuator、Micrometer 支持（Starter 已将其标记为 optional），请在应用侧按需引入：

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

可用配置（greeting.*）

- `greeting.enabled`：是否启用 Starter（默认：true）
- `greeting.web-enabled`：是否启用 Web 端点（默认：true）
- `greeting.health-enabled`：是否启用健康检查（默认：true）
- `greeting.prefix`：问候语前缀（默认："Hello, "）
- `greeting.suffix`：问候语后缀（默认："!"）
- `greeting.language`：语言（默认："en"，示例值：en/zh/ja/ko）
- `greeting.cache-enabled`：是否启用缓存（默认：false）
- `greeting.cache-expire-seconds`：缓存过期秒数（默认：300）

示例（`greeting-app/src/main/resources/application.properties`）：

```
# Greeting properties
greeting.prefix=hi,
greeting.suffix=!
greeting.enabled=true
greeting.web-enabled=true
greeting.health-enabled=true
greeting.language=en
greeting.cache-enabled=false
greeting.cache-expire-seconds=300

# Actuator properties
management.endpoints.web.exposure.include=health,info
management.endpoint.health.show-details=always
management.info.env.enabled=true
```

自动配置说明

`com.example.greeting.GreetingAutoConfiguration`

- 条件：
  - `greeting.enabled=true`（默认启用）
  - `@ConditionalOnMissingBean` 允许用户自定义覆盖
- 装配的 Bean：
  - `GreetingService`：核心服务
  - `GreetingHealthIndicator`：当 `greeting.health-enabled=true`
  - `GreetingController`：在 Servlet Web 环境且 `greeting.web-enabled=true`
  - `GreetingMetrics`：当存在 `MeterRegistry`（Micrometer）

运行演示应用

```

```

```
cd greeting-app
mvn spring-boot:run
```

启动后控制台会输出当前配置、健康检查与示例问候语。默认端口 `8080`。

可用端点

- HTTP：
  - `GET /api/greeting`：默认问候
  - `GET /api/greeting/{name}`：按名称问候
- Actuator：
  - `GET /actuator/health`：健康检查
  - `GET /actuator/info`：应用信息

指标（可选）

当引入 Micrometer 并存在 `MeterRegistry` 时，自动采集：

- `greeting.requests.total`：问候请求次数
- `greeting.errors.total`：问候错误次数

自定义与扩展

- 覆盖默认实现：在应用中自行声明 `GreetingService`、`GreetingController`、`GreetingHealthIndicator` 或 `GreetingMetrics` Bean，可覆盖 Starter 默认 Bean（受 `@ConditionalOnMissingBean` 影响）
- 关闭某些能力：通过 `greeting.enabled/web-enabled/health-enabled=false` 逐项关闭
- 添加更多属性：扩展 `GreetingProperties` 并更新 `spring-configuration-metadata.json`

常见问题

- 端点未生效：确认应用已引入 `spring-boot-starter-web` 且 `greeting.web-enabled=true`
- 健康检查无输出：确认已引入 `spring-boot-starter-actuator` 且 `greeting.health-enabled=true`
- 指标无数据：确认存在 `MeterRegistry`（例如通过 Actuator 或 Micrometer 适配器）

许可证

本项目遵循仓库根目录的 `LICENSE`。