# ⚗ Propiedades comunes de la aplicación

JHipster genera una aplicación Spring Boot y puede configurarse utilizando el mecanismo de propiedades estándar de Spring Boot.

JHipster configura esas propiedades en tiempo de generación y, a menudo, tienen valores diferentes en los modos de desarrollo y producción: obtenga más información sobre esto en nuestra documentación de Perfiles (https://www.jhipster.tech/profiles/) .

En una aplicación JHipster, hay tres tipos de propiedades:

1. Propiedades de aplicación estándar de Spring Boot
2. Propiedades de la aplicación JHipster
3. Propiedades específicas de la aplicación

## Propiedades de aplicación estándar de Spring Boot

Al igual que cualquier aplicación Spring Boot, JHipster le permite configurar cualquier propiedad (https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html) estándar de la aplicación Spring Boot (https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html) .

## Propiedades de la aplicación JHipster

JHipster proporciona propiedades de aplicación específicas, que provienen de la biblioteca del lado del servidor de JHipster (https://github.com/jhipster/jhipster) . Esas propiedades son estándar para todos los proyectos de JHipster, pero algunos de ellos solo funcionan según lo que seleccionó cuando creó su aplicación: por ejemplo, la `jhipster.cache.hazelcast` clave solo funciona si seleccionó Hazelcast como su caché de Hibernate de segundo nivel.

Esas propiedades se configuran utilizando la `io.github.jhipster.config.JHipsterProperties` clase.

Aquí hay una documentación para esas propiedades:

```
jhipster:

    # Thread pool that will be used for asynchronous method calls in JHipster
    async:
        core-pool-size: 2 # Initial pool size
        max-pool-size: 50 # Maximum pool size
        queue-capacity: 10000 # Queue capacity of the pool

    # Specific configuration for JHipster gateways
    # See https://www.jhipster.tech/api-gateway/ for more information on JHipster gateway
s
    gateway:
        rate-limiting:
            enabled: false # Rate limiting is disabled by default
            limit: 100_000L # By default we allow 100,000 API calls
            duration-in-seconds: 3_600 # By default the rate limiting is reinitialized ev
ery hour
        authorized-microservices-endpoints: # Access Control Policy, if left empty for a
 route, all endpoints will be accessible
            app1: /api # recommended prod configuration, it allows the access to all API
 calls from the "app1" microservice

    # HTTP configuration
    http:
        # V_1_1 for HTTP/1.1 or V_2_0 for HTTP/2.
        # To use HTTP/2 you will need SSL support (see the Spring Boot "server.ssl" confi
guration)
        version: V_1_1
        #Force the server cipher suite to follow the exact order specifying in server.ss
l.ciphers (For perfect forward secrecy)
        useUndertowUserCipherSuitesOrder: true
        cache: # Used by io.github.jhipster.web.filter.CachingHttpHeadersFilter
            timeToLiveInDays: 1461 # Static assets are cached for 4 years by default

    # Hibernate 2nd level cache, used by CacheConfiguration
    cache:
        hazelcast: # Hazelcast configuration
            time-to-live-seconds: 3600 # By default objects stay 1 hour in the cache
            backup-count: 1 # Number of objects backups
            # Configure the Hazelcast management center
            # Full reference is available at: http://docs.hazelcast.org/docs/management-c
enter/3.9/manual/html/Deploying_and_Starting.html
            management-center:
                enabled: false # Hazelcast management center is disabled by default
                update-interval: 3 # Updates are sent to the Hazelcast management center
 every 3 seconds by default
                # Default URL for Hazelcast management center when using JHipster's Docke
r Compose configuration
                # See src/main/docker/hazelcast-management-center.yml
                # Warning, the default port is 8180 as port 8080 is already used by JHips
ter
                url: http://localhost:8180/mancenter
        ehcache: # Ehcache configuration
            time-to-live-seconds: 3600 # By default objects stay 1 hour in the cache
            max-entries: 100 # Number of objects in each cache entry
        caffeine: # Caffeine configuration
            time-to-live-seconds: 3600 # By default objects stay 1 hour in the cache
```

```
            max-entries: 100 # Number of objects in each cache entry
        infinispan: #Infinispan configuration
            config-file: default-configs/default-jgroups-tcp.xml
            # local app cache
            local:
                time-to-live-seconds: 60 # By default objects stay 1 hour (in minutes) in
the cache
                max-entries: 100 # Number of objects in each cache entry
            #distributed app cache
            distributed:
                time-to-live-seconds: 60 # By default objects stay 1 hour (in minutes) in
the cache
                max-entries: 100 # Number of objects in each cache entry
                instance-count: 1
            #replicated app cache
            replicated:
                time-to-live-seconds: 60 # By default objects stay 1 hour (in minutes) in
the cache
                max-entries: 100 # Number of objects in each cache entry
        # Memcached configuration
        # Uses the Xmemcached library, see https://github.com/killme2008/xmemcached
        memcached:
            # Disabled by default in dev mode, as it does not work with Spring Boot devto
ols
            enabled: true
            servers: localhost:11211 # Comma or whitespace separated list of servers' add
resses
            expiration: 300 # Expiration time (in seconds) for the cache
            use-binary-protocol: true # Binary protocol is recommended for performance (a
nd security)
        redis: # Redis configuration
            expiration: 3600 # By default objects stay 1 hour (in seconds) in the cache
            server: redis://localhost:6379 # Server address

    # E-mail properties
    mail:
        enabled: false # If e-mail sending is enabled. The standard `spring.mail` keys wi
ll need to be configured
        from: jhipster@localhost # The default "from" address for e-mails
        base-url: http://127.0.0.1:8080 # URL to the application, used inside e-mails

    # Spring Security specific configuration
    security:
        remember-me: # JHipster secure implementation of the remember-me mechanism, for s
ession-based authentication
            # security key (this key should be unique for your application, and kept secr
et)
            key: 0b32a651e6a65d5731e869dc136fb301b0a8c0e4
        client-authorization: # Used with JHipster UAA authentication
            access-token-uri: # URL of the JHipster UAA server OAuth tokens
            token-service-id: # ID of the current application
            client-id: # OAuth client ID
            client-secret: # OAuth client secret
        authentication:
            jwt: # JHipster specific JWT implementation
                # The secret token should be encoded using Base64 (you can type `echo 'se
cret-key'|base64` on your command line).
                # If both properties are configured, the `secret` property has a higher p
```

```
riority than the `base64-secret` property.
                secret: # JWT secret key in clear text (not recommended)
                base64-secret:  # JWT secret key encoded in Base64 (recommended)
                token-validity-in-seconds: 86400 # Token is valid 24 hours
                token-validity-in-seconds-for-remember-me: 2592000 # Remember me token is
valid 30 days

    # Swagger configuration
    swagger:
        default-include-pattern: /api/.*
        title: JHipster API
        description: JHipster API documentation
        version: 0.0.1
        terms-of-service-url:
        contact-name:
        contact-url:
        contact-email:
        license:
        license-url:
        host:
        protocols:

    # DropWizard Metrics configuration, used by MetricsConfiguration
    metrics:
        jmx: # Export metrics as JMX beans
            enabled: true # JMX is enabled by default
        # Send metrics to a Graphite server
        # Use the "graphite" Maven profile to have the Graphite dependencies
        graphite:
            enabled: false # Graphite is disabled by default
            host: localhost
            port: 2003
            prefix: jhipster
        # Send metrics to a Prometheus server
        prometheus:
            enabled: false # Prometheus is disabled by default
            endpoint: /prometheusMetrics
        logs: # Reports Dropwizard metrics in the logs
            enabled: false
            reportFrequency: 60 # frequency of reports in seconds

    # Logging configuration, used by LoggingConfiguration
    logging:
        logstash: # Forward logs to Logstash over a socket
            enabled: false # Logstash is disabled by default
            host: localhost # Logstash server URL
            port: 5000 # Logstash server port
            queue-size: 512 # Queue for buffering logs
        spectator-metrics: # Reports Netflix Spectator metrics in the logs
            enabled: false # Spectator is disabled by default

    # By default cross-origin resource sharing (CORS) is enabled in "dev" mode for
    # monoliths and gateways.
    # It is disabled by default in "prod" mode for security reasons, and for microservice
s
    # (as you are supposed to use a gateway to access them).
    # This configures a standard org.springframework.web.cors.CorsConfiguration
    # Note that "exposed-headers" is mandatory for JWT-based security, which uses
```

```
    # the "Authorization" header, and which is not a default exposed header.
    cors:
        allowed-origins: "*"
        allowed-methods: "*"
        allowed-headers: "*"
        exposed-headers: "Authorization"
        allow-credentials: true
        max-age: 1800

    # Ribbon displayed on the top left-hand side of JHipster applications
    ribbon:
        # Comma-separated list of profiles that display a ribbon
        display-on-active-profiles: dev
```

# Propiedades específicas de la aplicación

Su aplicación generada también puede tener sus propias propiedades Spring Boot. Esto es muy recomendable, ya que permite la configuración segura de tipo de la aplicación, así como la finalización automática y la documentación dentro de un IDE.

JHipster ha generado una `ApplicationProperties` clase en el `config` paquete, que ya está preconfigurado, y ya está documentado en la parte inferior de la `application.yml` , `application-dev.yml` y `application-prod.yml` archivos. Todo lo que necesita hacer es codificar sus propias propiedades específicas.

---

(http://www.octoconsulting.com/)

Octo Consulting Group (http://www.octoconsulting.com/)

- Equipo (https://www.jhipster.tech/team/)
- Ilustraciones (https://www.jhipster.tech/artwork/)
- Políticas (https://www.jhipster.tech/policies/)
- Meetups (https://www.jhipster.tech/meetups/)
- Archivo (https://www.jhipster.tech/documentation-archive/)

(https://github.com/jhipster/generator-jhipster)

(https://twitter.com/java_hipster)