Introduction to JHipster

Hackathon evening, September 2019

Orestis Palampougioukis

JHipster

Problem

- A lot of modern web apps have high complexity and require:
  - *Beautiful design*
  - *No page reloads*
  - *Ease and speed of deployment*
  - *Extensive testing*
  - *Robustness and scalability of high-performance servers and deployment process*
  - *Monitoring*
  - *….*

Large amount of technologies working in sync to achieve all that => huge amount of effort into configurations / setting up

JHipster

- Open source platform using Yeoman to generate / develop / deploy Spring Boot + front-end web apps

- CLI for initial app generation + subsequent additions of:
  - Entities (frontend + backend)
  - Relationships
  - Spring controllers
  - Spring services
  - Internationalization
  - ...

Goal

- A beautiful front-end, with the latest HTML5/CSS3/JavaScript frameworks
- A robust and high-quality back-end, with the latest Java/Caching/Data access technologies
- All automatically wired up, with security and performance in mind
- Great developer tooling, for maximum productivity

Client side

- NPM dependency management to install and run client-side tools
- Webpack
  - *Compile, optimize, minimize*
  - *Efficient production builds*
- BrowserSync
  - Hot reload
- Testing
  - Jest, Gatling, Cucumber, Protractor
- Bootstrap
- Angular / React

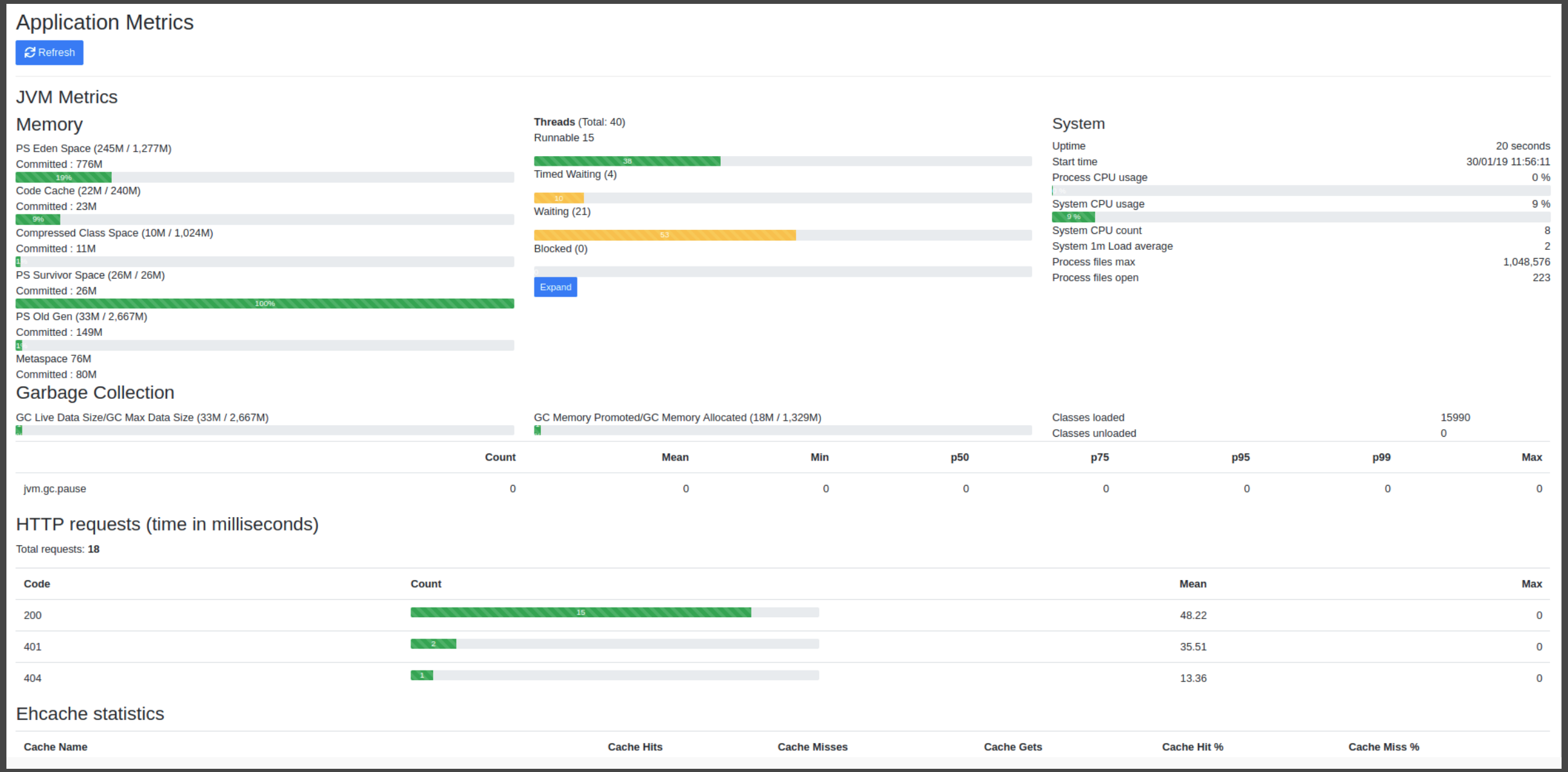
Server side

- Spring Boot
  - Configured out of the box
  - Live reload
- Maven / Gradle
- Netflix OSS
  - Eureka - load balancing & failover
  - Zuul – Proxy for dynamic routing, monitoring, security
  - Ribbon – Software load balancing for services
- Liquibase
  - DB source control

Server side

- JPA (Java Persistence API), Spring Data JPA
- MongoDB, Couchbase, Cassandra
- Elasticsearch
- Spring Security
- Thymeleaf (Java templating)
- Monitoring (JVM, app server, Spring Beans, Cache…)
- Docker / Docker-compose fully pre-configured

Monitoring


Application Metrics
Refresh
JVM Metrics
Memory
PS Eden Space (245M / 1,277M)
Committed : 776M
19%
Code Cache (22M / 240M)
Committed : 23M
9%
Compressed Class Space (10M / 1,024M)
Committed : 11M
1
PS Survivor Space (26M / 26M)
Committed : 26M
100%
PS Old Gen (33M / 2,667M)
Committed : 149M
1
Metaspace 76M
Committed : 80M
Threads (Total: 40)
Runnable 15
38
Timed Waiting (4)
10
Waiting (21)
53
Blocked (0)
Expand
System
Uptime 20 seconds
Start time 30/01/19 11:56:11
Process CPU usage 0 %
System CPU usage 9 %
9 %
System CPU count 8
System 1m Load average 2
Process files max 1,048,576
Process files open 223
Garbage Collection
GC Live Data Size/GC Max Data Size (33M / 2,667M)
GC Memory Promoted/GC Memory Allocated (18M / 1,329M)
Classes loaded 15990
Classes unloaded 0
Count Mean Min p50 p75 p95 p99 Max
jvm.gc.pause 0 0 0 0 0 0 0 0
HTTP requests (time in milliseconds)
Total requests: 18
Code Count Mean Max
200 15 48.22 0
401 2 35.51 0
404 1 13.36 0
Ehcache statistics
Cache Name Cache Hits Cache Misses Cache Gets Cache Hit % Cache Miss %

Monitoring

Endpoints requests (time in millisecond)

Method	Endpoint url	Count	Mean
GET	/management/audits	1	71.006
GET	/management/info	1	73.312
POST	/management/loggers/{name}	9	4.424
POST	/api/authenticate	1	546.659
GET	root	1	92.757
GET	/management/health	1	20.766
GET	/management/loggers	10	13.617
GET	/api/account	1	42.089
GET	/**	3	47.853

Cache statistics

Cache name	Cache Hits	Cache Misses	Cache Gets	Cache Puts	Cache Removals	Cache Evictions	Cache Hit %	Cache Miss %
com.mycompany.myapp.domain.User	0	0	0	1	0	0	0	0
usersByEmail	0	0	0	0	0	0	0	0
usersByLogin	1	1	2	1	0	0	50	50
com.mycompany.myapp.domain.Authority	0	0	0	2	0	0	0	0
com.mycompany.myapp.domain.User.authorities	0	0	0	1	0	0	0	0
com.mycompany.myapp.domain.Teacher	0	0	0	0	0	0	0	0

DataSource statistics (time in millisecond)

Connection Pool Usage (active: 0, min: 10, max: 10, idle: 10)	Count	Mean	Min	p50	p75	p95	p99	Max
Acquire	27	0.79	0	0	0	0	0	0
Creation	0	0	0	0	0	0	0	0
Usage	27	6.19	0	0	0	0	0	0

Deployment / Cloud

- Kubernetes
- Heroku
- AWS
- Boxfuse
- Google cloud
- OpenShift
- CloudFoundry

Sub-generators

- jhipster kubernetes
- Answer a few questions
- Done

Marketplace

- Modules
- Blueprints

Blueprints

- Enhance JHipster with new features such as supproting different languages / frameworks
- Demonstrate how the main generator behavior can be modified to fit anyone's needs
- Kotlin
  - Replaces most Java backend with Kotlin
- Vue.js
  - Replaces frontend logic with Vue.js
- .Net
- Node.js
  - Replaces Java side with Nest.js framework

Opinion

- Amazingly efficient for greenfield projects
  - Adhering to the generated structure matters
- Cumbersome for projects that need to adhere to pre-existing structure
  - Can still be very beneficial to setup the initial configuration

Thank you

After dinner :)

- Install Jhipster
  - https://www.jhipster.tech/installation/
- Generate a JHipster project with your preferred initial set-up
  - https://www.jhipster.tech/creating-an-app/
- Use the generator to create entities
  - https://www.jhipster.tech/creating-an-entity/
- Create a Spring service
  - https://www.jhipster.tech/creating-a-spring-service/
- ...