Microprofile.io
не спрингом единым

Микросервисы можно писать и не на буте

T · · Systems ·

Дмитрий Александров

@bercut2000

Старший проектировщик информационных систем

Представьте себе невероятное:
Нам надо написать веб приложение!

Естественно надо разбить все на микросервисы

Мы все ОЧЕНЬ МНОГО слышали
про микросервисы

Но их надо как-то писать!

Прориворугатель:

На этом докладе мы НЕ будем рассуждать:

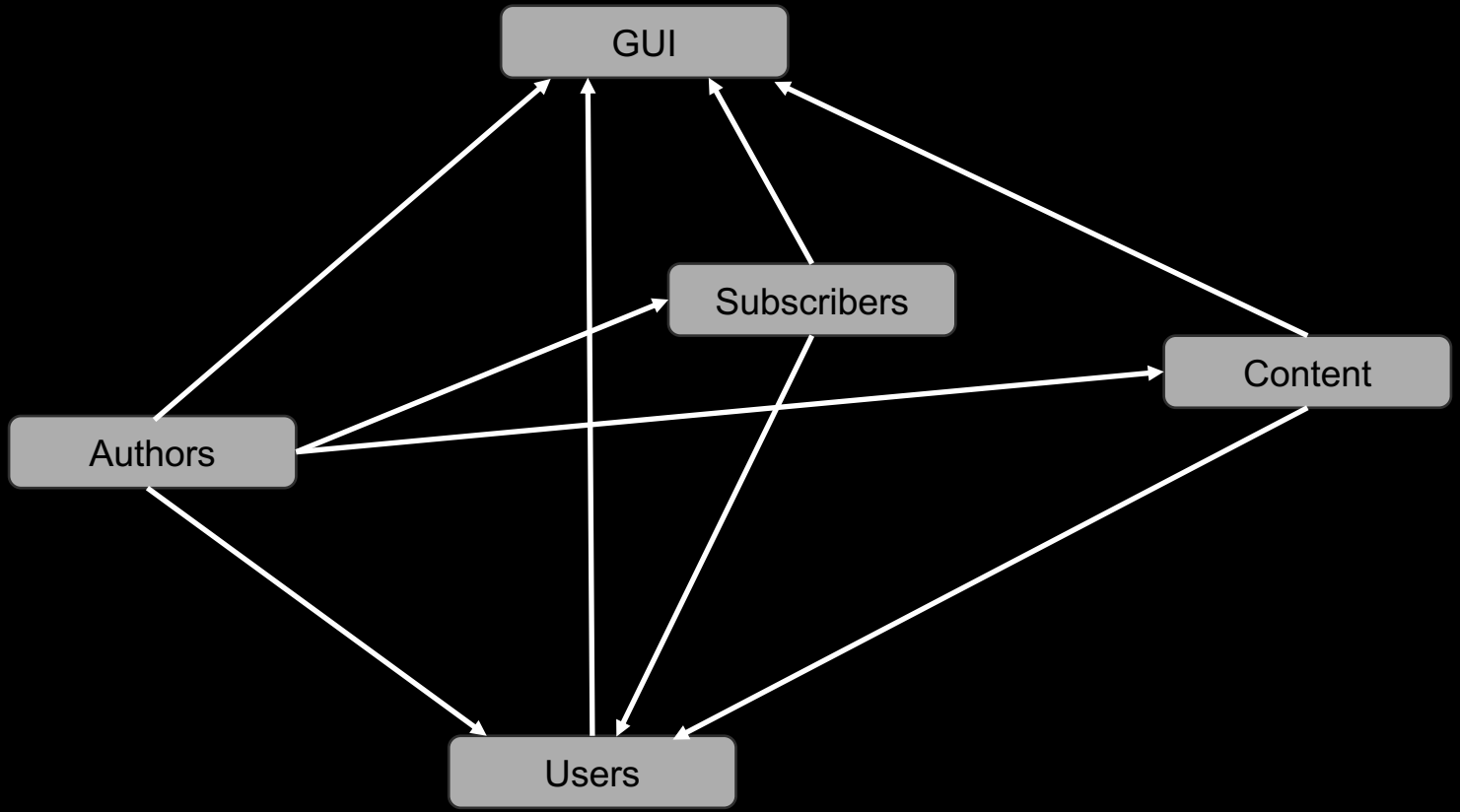
- что такое микросервис
- чем он лучше или хуже других
- зачем та или иная технология

Наш основной вопрос: как?

,а не зачем?

Демо 1

Magazine Manager



Это надо сделать модно и красиво!

Node? … Go?

.. нее, у нас Java стек!

.. ну тогда естественно!

А что еще? Не Вебсферу же прикручивать?!

А что еще? Не Вебсферу же прикручивать?!
(хотя об этом позже...)

Собственно, а что такое..

Спринг буут

Spring Boot это Spring-овое **convention-over-configuration** решение для создания stand-alone, production-grade Spring-овых приложений, которые можно "просто запускать". Оно уже преконфигурировано под “оптимальное” исполнение наиболее стандартных приложений. Почти не нужно конфигурировать..

Спринг буут

Крутое, потому что:

- Stand-alone

Спринг буут

Крутое, потому что:

- Stand-alone
- Embed Tomcat или Jetty (не нужно деплоить WAR)

Спринг буут

Крутое, потому что:

- Stand-alone
- Embed Tomcat или Jetty (не нужно деплоить WAR)
- Предоставляет 'starter' POMs, чтобы не путаться

Спринг буут

Крутое, потому что:

- Stand-alone
- Embed Tomcat или Jetty (не нужно деплоить WAR)
- Предоставляет 'starter' POMs, чтобы не путаться
- Автоматическая конфигурация Spring

Спринг буут

Крутое, потому что:

- Stand-alone
- Embed Tomcat или Jetty (не нужно деплоить WAR)
- Предоставляет 'starter' POMs, чтобы не путаться
- Автоматическая конфигурация Spring
- Полезные плюшки metrics, health checks и externalized configuration

Спринг буут

Крутое, потому что:

- Stand-alone
- Embed Tomcat или Jetty (не нужно деплоить WAR)
- Предоставляет 'starter' POMs, чтобы не путаться
- Автоматическая конфигурация Spring
- Полезные плюшки metrics, health checks и externalized configuration
- Вообще нет кодогенерации и не нужна XML конфигурация

По идее, Спринг прекрасен!

По идее, Спринг прекрасен!
И так думают все!

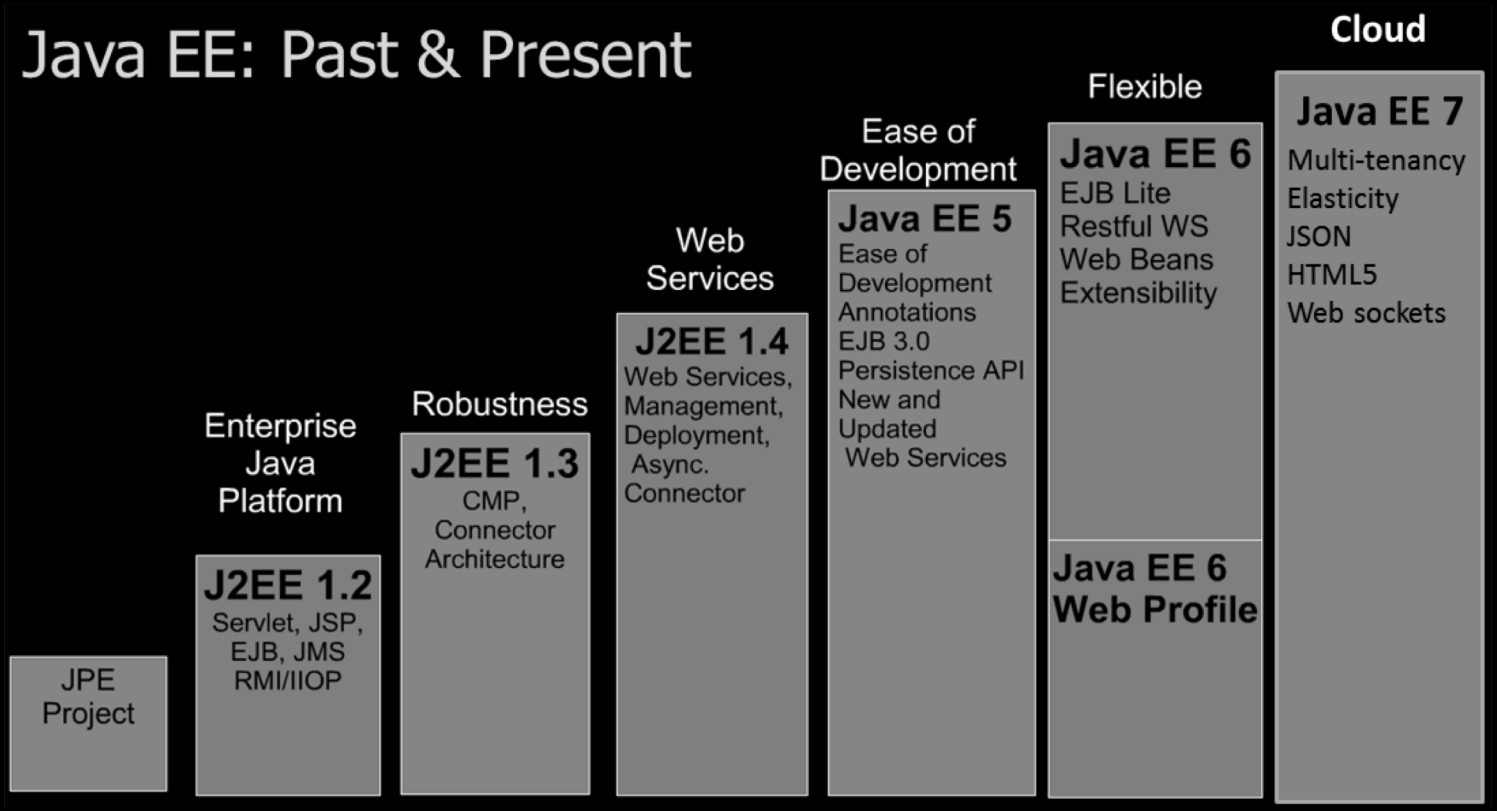
По идее, Спринг прекрасен!
И так думают все!
И не без почвы!

Демо 2

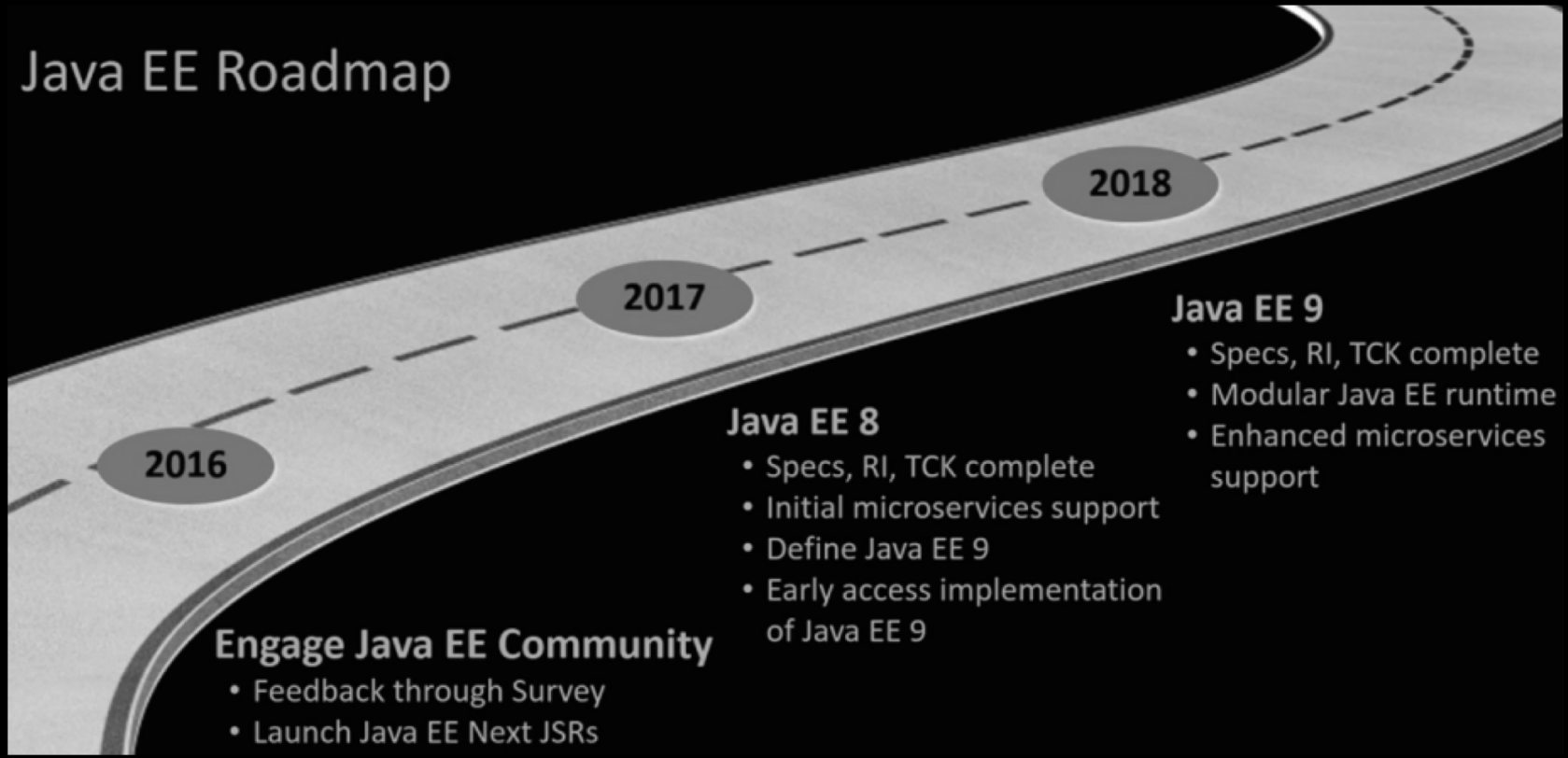
Но .. он один такой?

Как я перестал бояться и начал любить ЕЕ

EE:


Java EE: Past & Present
JPE Project
Enterprise Java Platform
J2EE 1.2
Servlet, JSP, EJB, JMS RMI/IIOP
Robustness
J2EE 1.3
CMP, Connector Architecture
Web Services
J2EE 1.4
Web Services, Management, Deployment, Async. Connector
Ease of Development
Java EE 5
Ease of Development Annotations EJB 3.0 Persistence API New and Updated Web Services
Flexible
Java EE 6
EJB Lite Restful WS Web Beans Extensibility
Java EE 6 Web Profile
Cloud
Java EE 7
Multi-tenancy Elasticity JSON HTML5 Web sockets

EE:

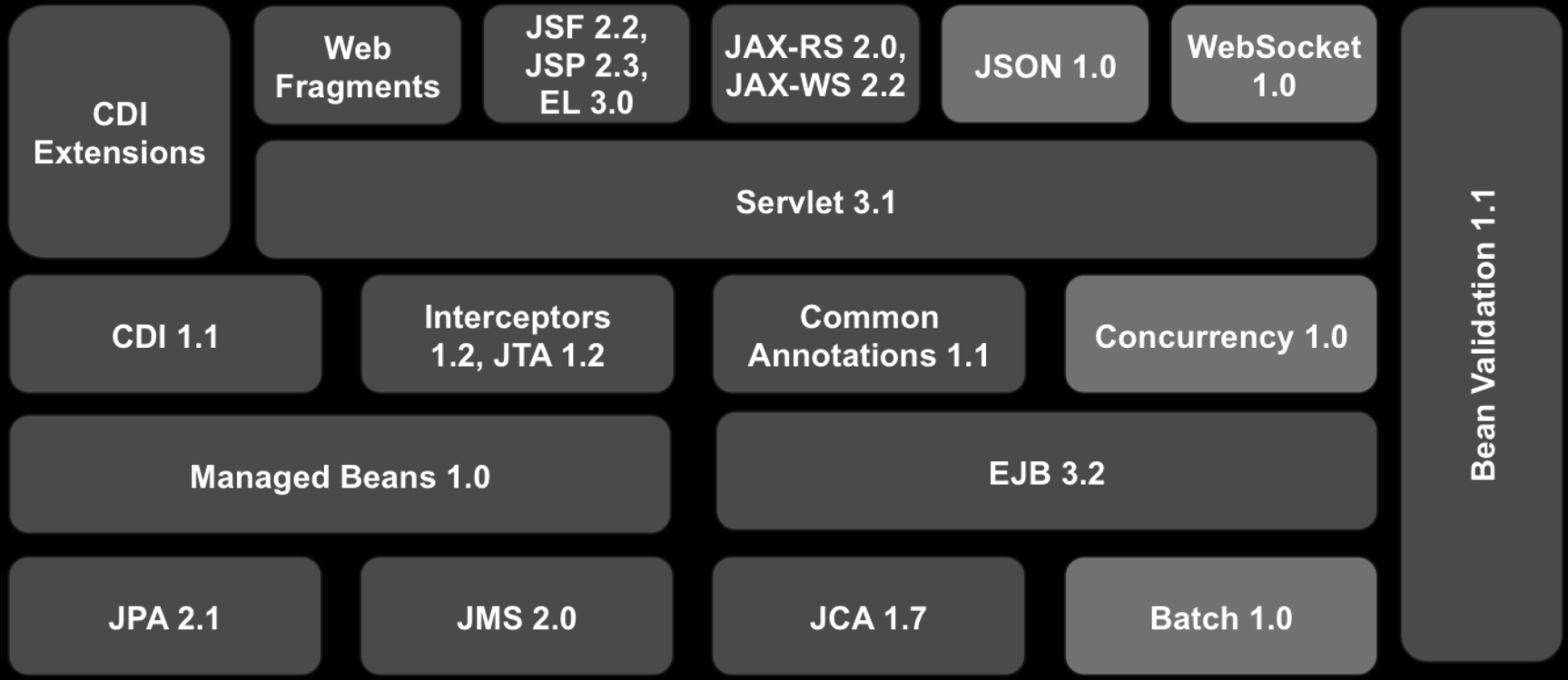

Java EE Roadmap
2016
2017
2018
Engage Java EE Community
• Feedback through Survey
• Launch Java EE Next JSRs
Java EE 8
• Specs, RI, TCK complete
• Initial microservices support
• Define Java EE 9
• Early access implementation of Java EE 9
Java EE 9
• Specs, RI, TCK complete
• Modular Java EE runtime
• Enhanced microservices support

Потом Вы сами знаете, что случилось!


EE4J

JAKARTA EE

EE:


CDI Extensions
Web Fragments
JSF 2.2, JSP 2.3, EL 3.0
JAX-RS 2.0, JAX-WS 2.2
JSON 1.0
WebSocket 1.0
Servlet 3.1
Bean Validation 1.1
CDI 1.1
Interceptors 1.2, JTA 1.2
Common Annotations 1.1
Concurrency 1.0
Managed Beans 1.0
EJB 3.2
JPA 2.1
JMS 2.0
JCA 1.7
Batch 1.0

А что там у спринга:

У них тоже много чего:

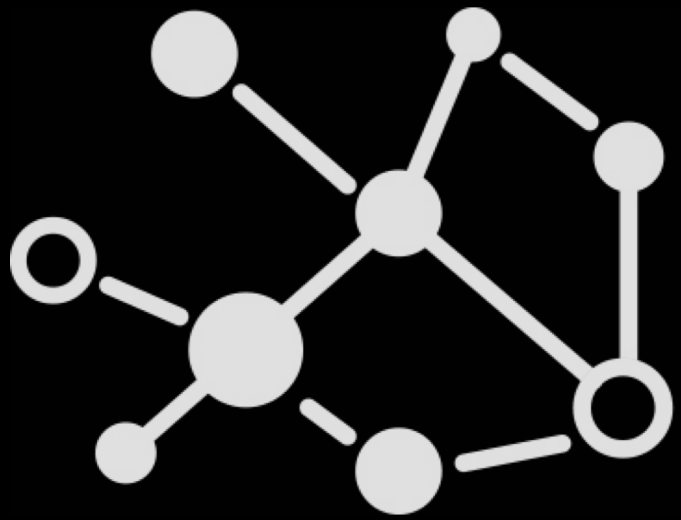
Кстати, мы тут немного не об этом..

Нам нужно написать несколько микросервисов и заставить их работать вместе!

Зачем поднимать все это безумие,
если нужно совсем немного..

Так подумали несколько кантор:

Так подумали несколько кантор:



MicroProfile

Ну это же все тоже самое ЕЕ!


ЭТО ВООБЩЕ
РАЗНЫЕ ВЕЩИ!
Opening
Mon
Tue-Thur
Fri-Sat
Sunday
imgflip.com

об чем:

- Eclipse MicroProfile это open-source community specification для Enterprise Java Microservices

об чем:

- Eclipse MicroProfile это open-source community specification для Enterprise Java Microservices
- Сообщество частных лиц, организаций и разработчиков, работающих в рамках проекта с открытым исходным кодом (Eclipse), для Enterprise Java Microservices

Основные концепции:

- Минималистичный Fat jar

Основные концепции:

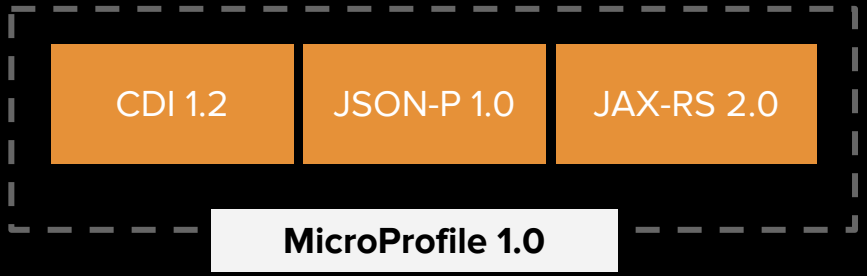
- Минималистичный Fat jar
- Минимальная (или полностью отсутствующая) конфигурация

Основные концепции:

- Минималистичный Fat jar
- Минимальная (или полностью отсутствующая) конфигурация
- Портабильность!

Сели и подумали:
что из этого всего нужно для написания микросервиса?

MicroProfile 1.0 (Sep, 2016)



MicroProfile 1.0 (Sep, 2016)

```
<dependency>
    <groupId>io.microprofile</groupId>
    <artifactId>microprofile</artifactId>
    <version>1.0.0</version>
    <scope>provided</scope>
    <type>pom</type>
</dependency>
```

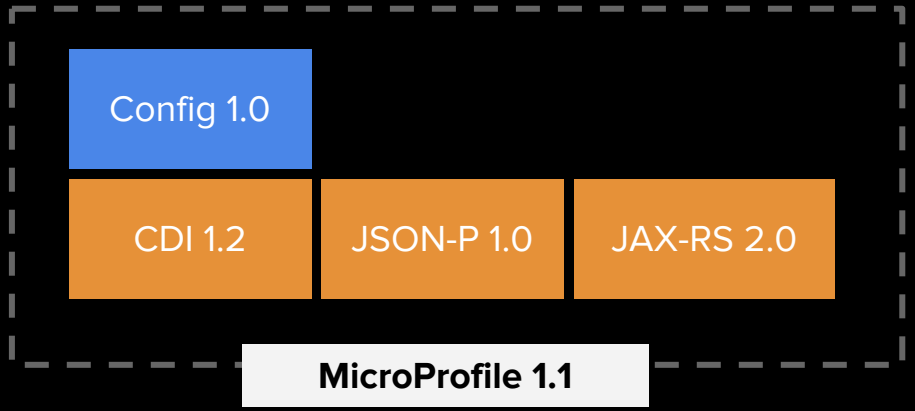
```
compile group: 'io.microprofile', name: 'microprofile', version:
'1.0.0', ext: 'pom'
```

Если сервер сертифицирован по (или пишет, что поддерживает) Microprofile 1.0 – просто запускаем приложение!

Демо 3

Портабильный код… но не конфигурация..

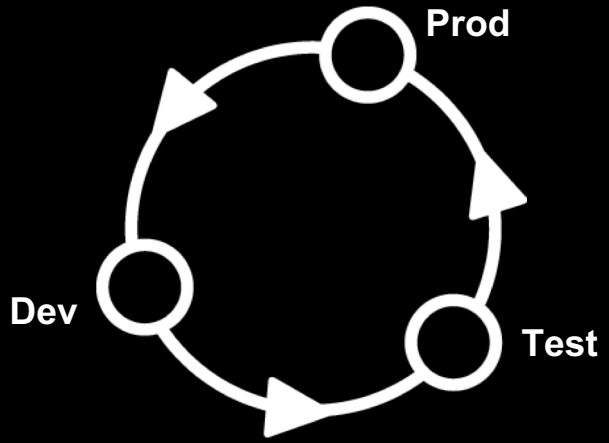
Eclipse MicroProfile 1.1 (Aug, 2017)



= New
= No change from last release

Конфигурация

логично выносить конфигурацию из программы, но каждый сервер конфигурится по-своему..



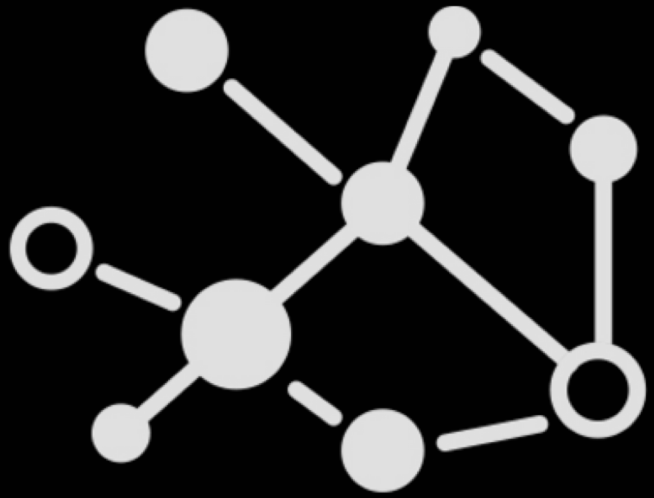
Конфигурация

- DeltaSpike Config (http://deltaspike.apache.org/documentation/configuration.html)
- Extracted parts of DeltaSpike Config (https://github.com/struberg/javaConfig/)
- Apache Tamaya (http://tamaya.incubator.apache.org/)
- Sabot, merged into Tamaya (https://tomitribe.io/p/sabot)

Eclipse MicroProfile 1.1 (Aug, 2017)

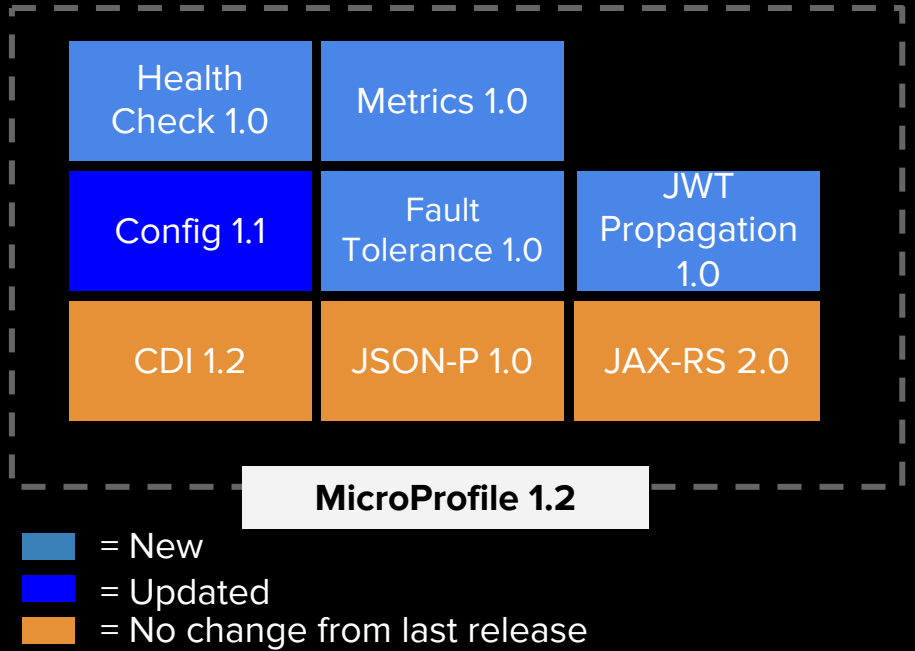
```
<dependency>
    <groupId>org.eclipse.microprofile.bom</groupId>
    <artifactId>microprofile-bom-spec</artifactId>
    <scope>provided</scope>
    <version>1.1.0</version>
    <type>pom</type>
</dependency>
```

```
compile group: 'org.eclipse.microprofile.bom', name:
'microprofile-bom-spec', version: '1.1.0', ext: 'pom'
```



Eclipse MicroProfile

Eclipse MicroProfile 1.2 (Sep, 2017)



Конфигурация 1.1

- API/SPI Изменения
  a. ConfigSource SPI был расширен с помощью метода по умолчанию, который возвращает имена свойств для данного ConfigSource
- Функциональные изменения
  a. Реализации теперь должны включать конвертер URL-адресов @Priority (1)
  b. Формат имени свойства по умолчанию для точки injection с использованием @ConfigProperty был изменен первой буквы класса. Реализации могут поддерживать это поведение. Вместо этого в MicroProfile Config 1.1 требуется использовать имя класса.
  c. Реализации теперь должны поддерживать примитивные типы, в дополнение к уже указанным примитивным типам враперов
- Изменения спецификации
  a. Уточнения по значениям параметров

Health Check 1.0

- Необходима совместимость с контейнерами (i.e. http://kubernetes.io/docs/user-guide/liveness/)

Health Check 1.0

- Необходима совместимость с контейнерами (i.e. http://kubernetes.io/docs/user-guide/liveness/)
- хорошо бы чтобы machine-to-machine communication как-то работало

Health Check 1.0

- Необходима совместимость с контейнерами (i.e. http://kubernetes.io/docs/user-guide/liveness/)
- хорошо бы чтобы machine-to-machine communication как-то работало

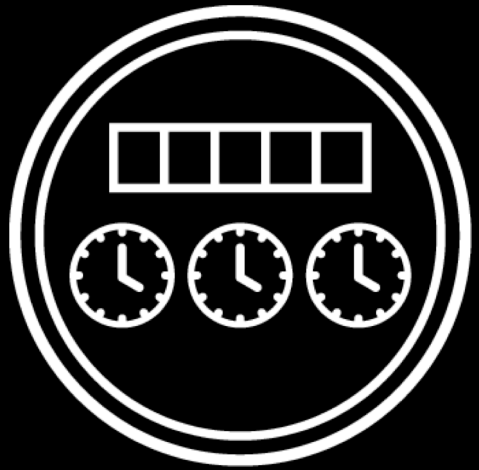
- … но чтобы и человек мог понять

Метрики

Нужно хорошо и правильно мониторить сервисы,

и это нужно делать из коробочки

Metric Registry



Required Base метрики
Application метрики
Vendor-specific метрики

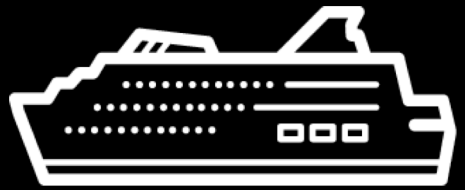
Metrics

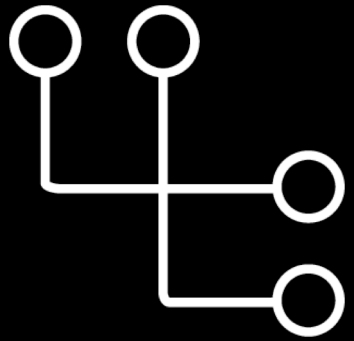
- @Counted
- @Gauge
- @Metered
- @Timed

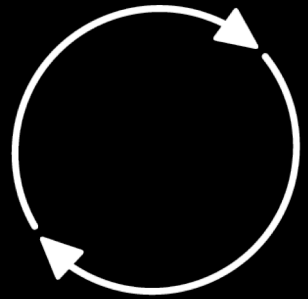
- Histogram
- MetricsRegistry

Fault Tolerance

Все ломается! Особенно микросервисы!



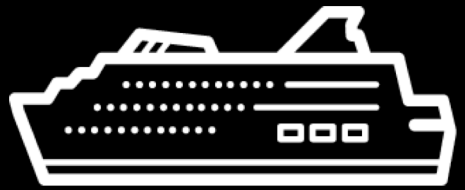


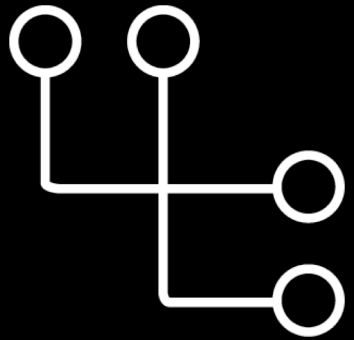


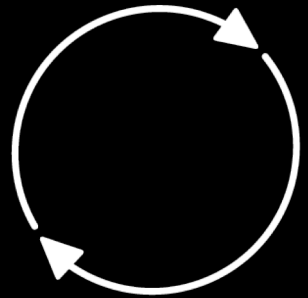
Fault Tolerance

Все ломается! Особенно микросервисы!

С этим надо жить! И хендлить их грациозно!







Fault Tolerance 1.0

В основном повлияли:

- Hystrix
- Failsafe

Цели:

- Отделить ответственность от логики (Runnables/Callables/etc) через retry policies, bulkheads, circuit breakers

Fault Tolerance 1.0 - что делает

- **Timeout**: Ну собственно все ясно

Fault Tolerance 1.0 - что делает

- **Timeout**: Ну собственно все ясно
- **RetryPolicy**: Определить критерии сколько еще пробовать

Fault Tolerance 1.0 - что делает

- **Timeout**: Ну собственно все ясно
- **RetryPolicy**: Определить критерии сколько еще пробовать
- **Fallback**: если чтото пойдет не так, как действовать дальше!

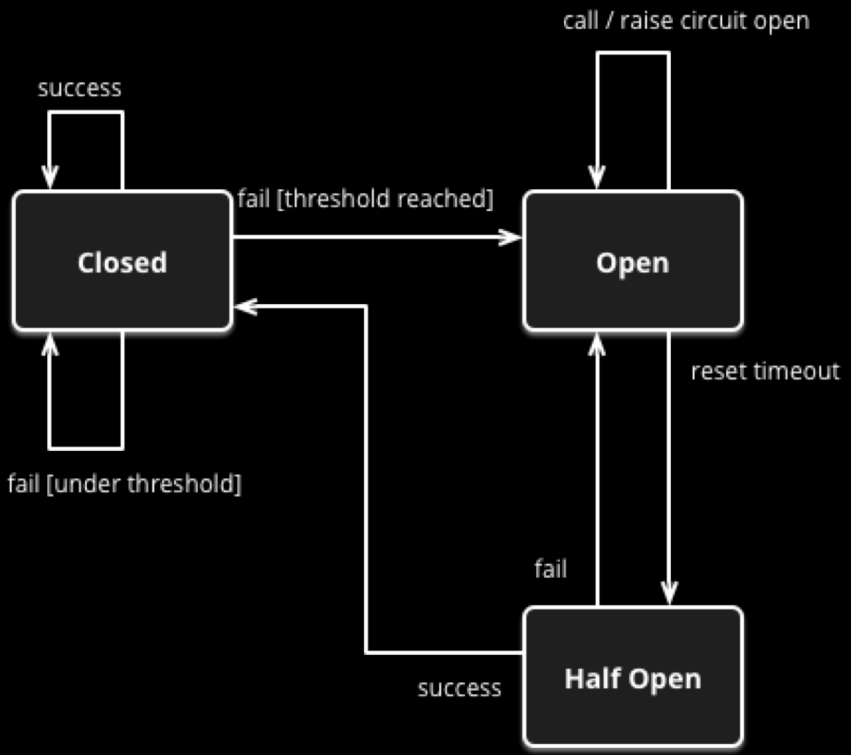
Fault Tolerance 1.0 - что делает

- **Timeout**: Ну собственно все ясно
- **RetryPolicy**: Определить критерии сколько еще пробовать
- **Fallback**: если чтото пойдет не так, как действовать дальше!
- **CircuitBreaker**: это сложно, предлагает способы предотвращать перегрузку системы или ненужные таймауты

Fault Tolerance 1.0 - что делает

- **Timeout**: Ну собственно все ясно
- **RetryPolicy**: Определить критерии сколько еще пробовать
- **Fallback**: если чтото пойдет не так, как действовать дальше!
- **CircuitBreaker**: это сложно, предлагает способы предотвращать перегрузку системы или ненужные таймауты
- **Bulkhead**: способ изоляции частей системы, в случае сбоев которых, остальные остаются работоспособными.

Circuit Breaker



Circuit Breaker

https://martinfowler.com/bliki/CircuitBreaker.html

JWT

Требования безопасности, связанные с архитектурой микросервиса, тесно связаны с безопасностью RESTful сервисов. В стиле архитектуры RESTful службы обычно не имеют состояния и любое состояние безопасности, связанное с клиентом, отправляется в целевую службу по каждому запросу, чтобы позволить службам повторно создавать контекст безопасности для вызывающего и выполнять проверку подлинности и авторизации

JWT

Повлияли в основном:

- OAuth2
- OpenID Connect(OIDC), and
- JSON Web Tokens(JWT)

Цель:

- Одна из основных стратегий распространения состояния безопасности от клиентов к услугам или даже от служб к услугам связана с использованием токенов безопасности.
- Для микросервисов на основе RESTful маркеры безопасности предлагают очень легкий и совместимый способ распространения идентификаторов между различными службами.

JWT

Инфраструктура:

Аннотации:

- @Clame (String or ClaimValue)
- @RolesAllowed
- @LoginConfig
- @DeclareRoles

JWT

Инфраструктура:

Классы:

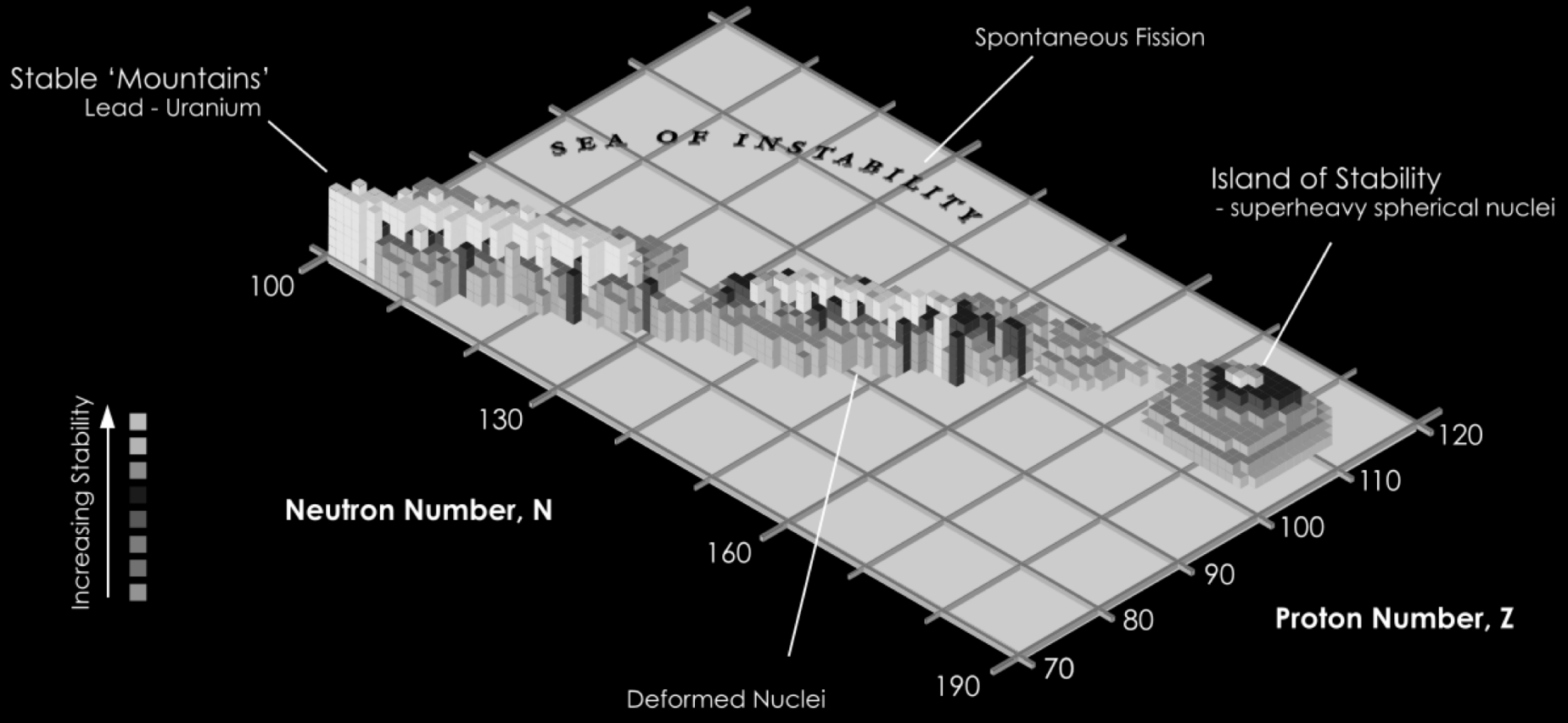
- @Inject JsonWebToken jwt;
- @Inject Principal principal;
- @Inject JsonNumber jsonNumber;
-
- Various JWT tools

Eclipse MicroProfile 1.2 (Sep, 2017)

```
<dependency>
    <groupId>org.eclipse.microprofile</groupId>
    <artifactId>microprofile</artifactId>
    <scope>provided</scope>
    <version>1.2</version>
    <type>pom</type>
</dependency>
```

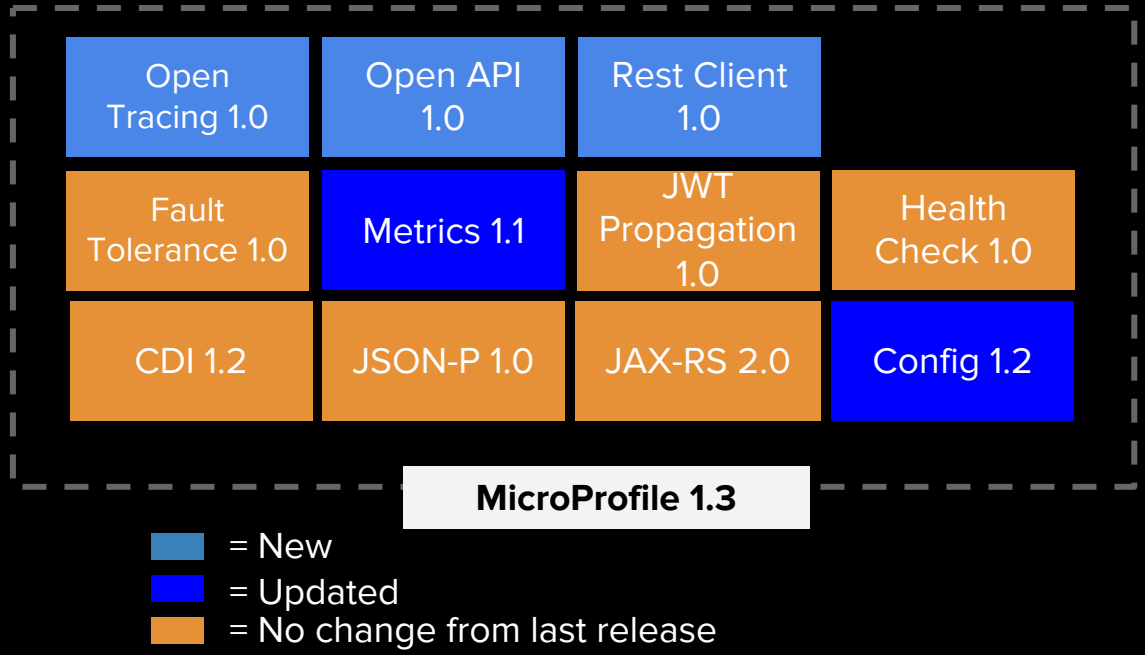
```
compile group: 'org.eclipse.microprofile', name: 'microprofile',
version: '1.2', ext: 'pom'
```

Демо 5


Stable 'Mountains'
Lead - Uranium
Spontaneous Fission
SEA OF INSTABILITY
Island of Stability
- superheavy spherical nuclei
100
130
160
190
120
110
100
90
80
70
Neutron Number, N
Proton Number, Z
Increasing Stability
Deformed Nuclei

Собственно, на этой версии заканчивается стабильность!

Eclipse MicroProfile 1.3 (Q1 CY2018)



Метрики 1.1

Что нового:

- Улучшенное TCK.
- org.eclipse.microprofile.metrics.MetricRegistry.register(String name, Metric, Metadata) депрекейтнуто.
- Use org.eclipse.microprofile.metrics.MetricRegistry.register(Metadata, Metric) instead, where Metadata already has a field for the name.
- Глобальные таги теперь доступны через MicroProfile Config (env переменная все еще валидна).
- Аннотации и метаданные теперь могут иметь флаг для многократного использования. Имя метрики может быть зарегистрировано более одного раза. Значение по умолчанию равно false.

Configuration 1.2

Новые фичи:

- SPI ConfigBuilder был расширен с помощью метода, который позволяет зарегистрировать конвертер с указанным типом класса. Это изменение устраняет ограничение, при котором не возможно добавить лямбда-конвертоты.

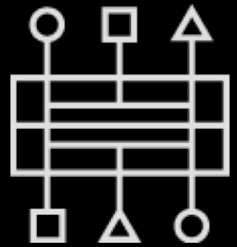
- Реализации должны теперь поддерживать преобразователь массива. Для преобразователя массива программный луукап свойства (например, config.getValue (myProp, String []. Class)) должен поддерживать возвращаемый тип массива. Для поиска injection необходимо также поддерживать массив, список или набор (например, @Inject @ConfigProperty (name = "myProp") список <String> propValue;).

- Реализации также должны поддерживать преобразователи общего плана, если для данного класса нет соответствующих типов преобразователей. Реализация должна использовать конструктор класса с одним строковым параметром, а затем попробовать valueOf (String), за которым следует CharSequence.

- Имплементации должны поддерживать Class конверторы

OpenAPI

Управление микросервисами в MSA может стать громоздким по мере увеличения количества микросервисов. Управление микросервисами осуществляется через их API. Управление, безопасность, балансировка нагрузки и дросселирование - это политики, которые могут применяться к API-интерфейсам с микросервисами. OpenAPI предоставляет Java-интерфейсы и модели программирования, которые позволяют разработчикам Java изначально создавать документы OpenAPI v3 из своих приложений JAX-RS.



OpenAPI 1.0

- Enterprise Java Binding of the OpenAPI v3 specification
- Основан на Swagger Core
- OpenAPI
  - Определяет стандартны, программное language-agnostic описание языка для REST APIs
  - Понимание и машин и человека

OpenTracing

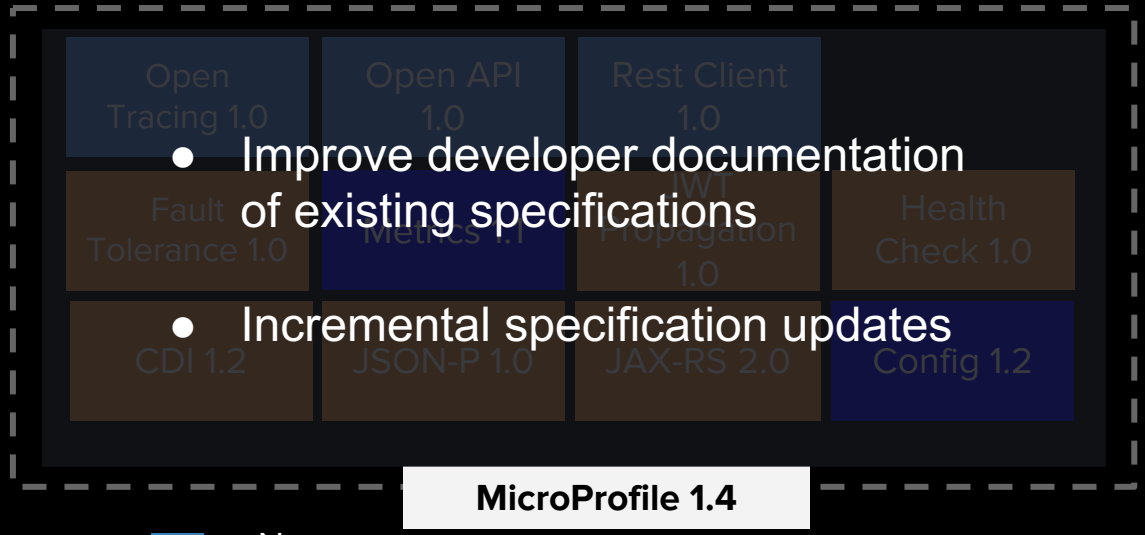
- Назначает каждому внешнему запросу уникальный идентификатор внешнего запроса
- Пропускает идентификатор внешнего запроса ко всем службам, связанным с обработкой запроса
- Включает внешний идентификатор запроса во всех сообщениях журнала
- Записывает информацию (например, время начала, время окончания) о запросах и операциях, выполняемых при обработке внешнего запроса в централизованной службе

Eclipse MicroProfile 1.3 (Q1 CY2018)

```
<dependency>
    <groupId>org.eclipse.microprofile</groupId>
    <artifactId>microprofile</artifactId>
    <version>1.3</version>
    <scope>provided</scope>
    <type>pom</type>
</dependency>
```

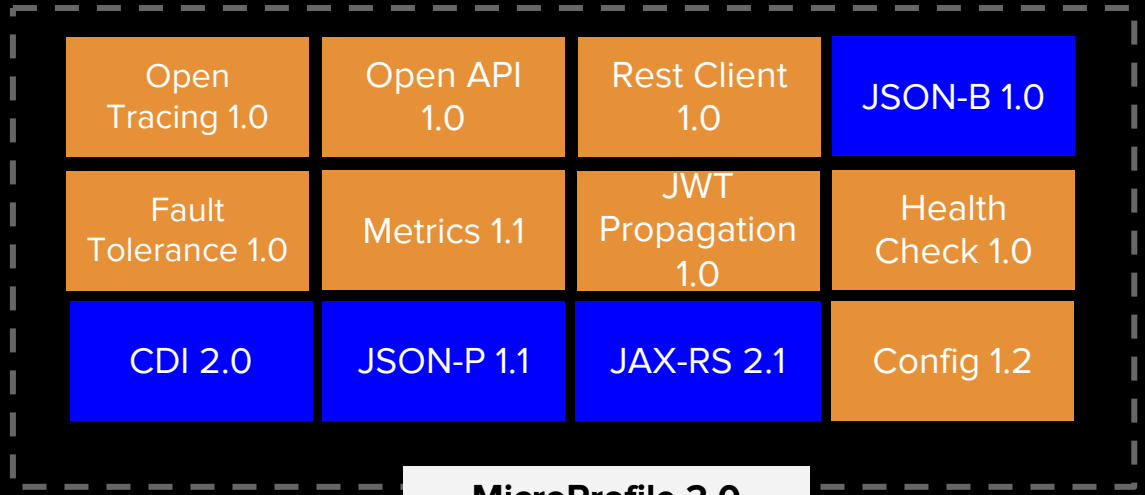
```
compile group: 'org.eclipse.microprofile', name: 'microprofile',
version: '1.3', ext: 'pom'
```

Eclipse MicroProfile 1.4 (Q2 CY2018)



Eclipse MicroProfile 2.0 (2H 2018?)

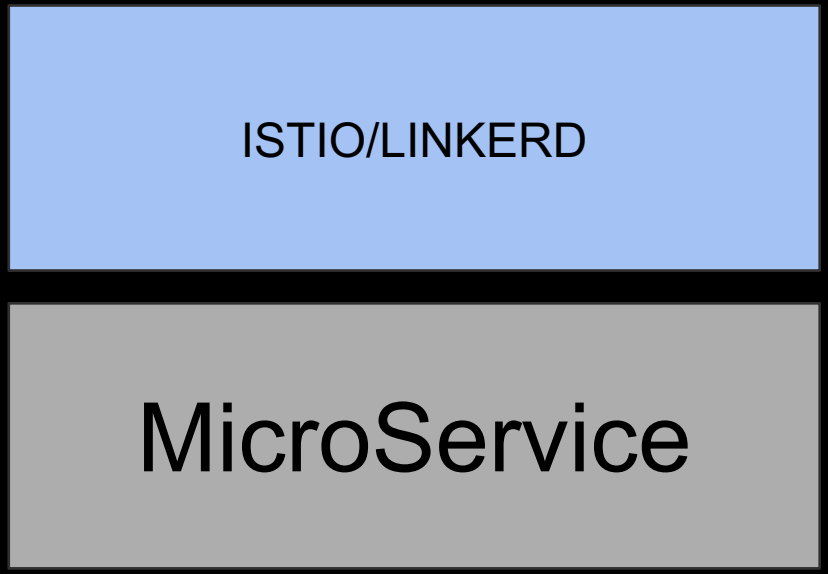
Roadmap



Ну, по идее можно и ISTIO прикрутить...

Service Mesh

Service Mesh



Service Mesh

https://github.com/eclipse/microprofile-service-mesh

Как там с тестированием?

Тестирование

Многие считают Arquillian тяжелым и трудным в настройке

Тестирование

https://antoniogoncalves.org/2018/01/16/java-ee-vs-spring-testing/

Ну так вот

Почему стоит задуматься применить Microprofile

- Оно стандартное

Ну так вот

Почему стоит задуматься применить Microprofile

- Оно стандартное
- Оно opensource

Ну так вот

Почему стоит задуматься применить Microprofile

- Оно стандартное
- Оно opensource
- Оно сапортится крутыми вендорами и комюнити

Ну так вот

Почему стоит задуматься применить Microprofile

- Оно стандартное
- Оно opensource
- Оно сапортится крутыми вендорами и комюнити
- Всего один POM

Ну так вот

Почему стоит задуматься применить Microprofile

- Оно стандартное
- Оно opensource
- Оно сапортится крутыми вендорами и комюнити
- Всего один POM
- Автоматическая конфигурация

Ну так вот

Почему стоит задуматься применить Microprofile

- Оно стандартное
- Оно opensource
- Оно сапортится крутыми вендорами и комюнити
- Всего один POM
- Автоматическая конфигурация
- Полезные плюшки metrics, health checks и externalized configuration

Ну так вот

Почему стоит задуматься применить Microprofile

- Оно стандартное
- Оно opensource
- Оно сапортится крутыми вендорами и комюнити
- Всего один POM
- Автоматическая конфигурация
- Полезные плюшки metrics, health checks и externalized configuration
- Вообще нет кодогенерации и не нужна XML конфигурация

Спринг буут *<- Вспомним это слайд*

Крутое, потому что:

- Stand-alone
- Embed Tomcat или Jetty (не нужно деплоить WAR)
- Предоставляет 'starter' POMs, чтобы не путаться
- Автоматическая конфигурация Spring
- Полезные плюшки metrics, health checks и externalized configuration
- Вообще нет кодогенерации и не нужна XML конфигурация

Если не разницы, зачем платить больше!

Именно платить:

Для разработчика оно может быть и бесплатное..

Именно платить:

Для разработчика оно может быть и бесплатное..

Но для конторы оно может быть очень даже платное

Именно платить:

Для разработчика оно может быть и бесплатное..

Но для конторы оно может быть очень даже платное

- А как там с лицензиями?

Именно платить:

Для разработчика оно может быть и бесплатное..

Но для конторы оно может быть очень даже платное

- А как там с лицензиями?
- А как там с саппортом?

Именно платить:

Для разработчика оно может быть и бесплатное..

Но для конторы оно может быть очень даже платное

- А как там с лицензиями?
- А как там с саппортом?
- А как там допусками и сертификацией?

Именно платить:

Для разработчика оно может быть и бесплатное..

Но для конторы оно может быть очень даже платное

- А как там с лицензиями?
- А как там с саппортом?
- А как там допусками и сертификацией?
- … А как там с законодательством?

А может плюсов больше?

Почему стоит задуматься применить Microprofile

- Практически переписывается заново

А может плюсов больше?

Почему стоит задуматься применить Microprofile

- Практически переписывается заново
- Опенсорс

А может плюсов больше?

Почему стоит задуматься применить Microprofile

- Практически переписывается заново
- Опенсорс
- Выбрасывается наследие EE

(например все делается с CDI)

А может плюсов больше?

Почему стоит задуматься применить Microprofile

- Практически переписывается заново
- Опенсорс
- Выбрасывается наследие EE

  (например все делается с CDI)

- Oracle не отстает (точнее немного отстает): Helidon

Helidon:

MicroFramework:

```
WebServer.create(
  Routing.builder()
    .get("/greet", (req, res)
      -> res.send("Hello World!"))
    .build())
  .start();
```

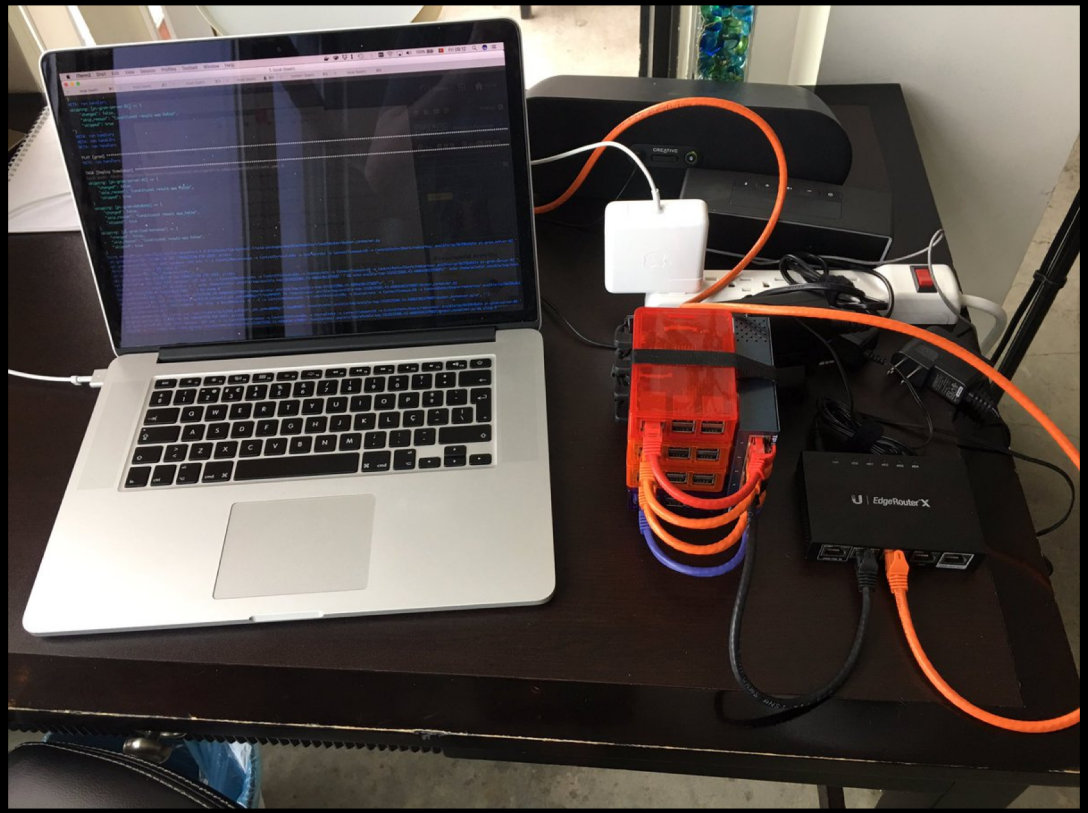
MicroProfile:

```
public class GreetService {
  @GET
  @Path("/greet")
  public String getMsg() {
    return "Hello World!";
  }
}
```

Повод задуматься:

Ну так вот, ЕЕ уже не тяжел!

Оно еще и легкое



Спасибо!!!