GIAC | BEIJING Dec.12.16-17

架構 ARCHNOTES 高可用架构

技术架构未来

thegiac.com

Java生态圈与微服务

丁雪丰

Java老矣，尚能饭否?

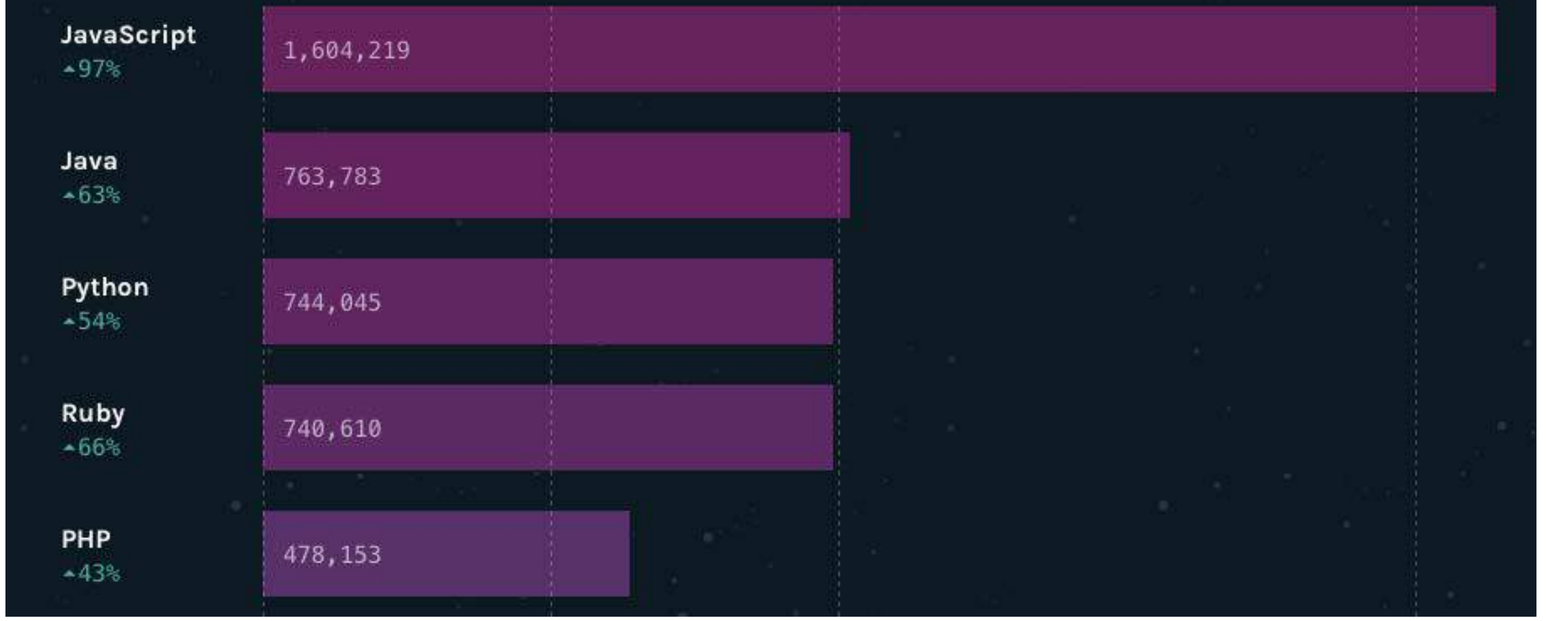
Java[redacted]，尚能饭[redacted]

Java在发展

- 发展历程
  - 诞生于1995年，1996年发布1.0
  - 2004年，发布重大更新1.5，并改为5.0
  - 2014年，发布Java 8
  - 2017年7月，发布Java 9（不再跳票的话）

In total, GitHub is home to open source projects written in **316 unique programming languages**. Here are the most popular by number of pull requests in the last twelve months.

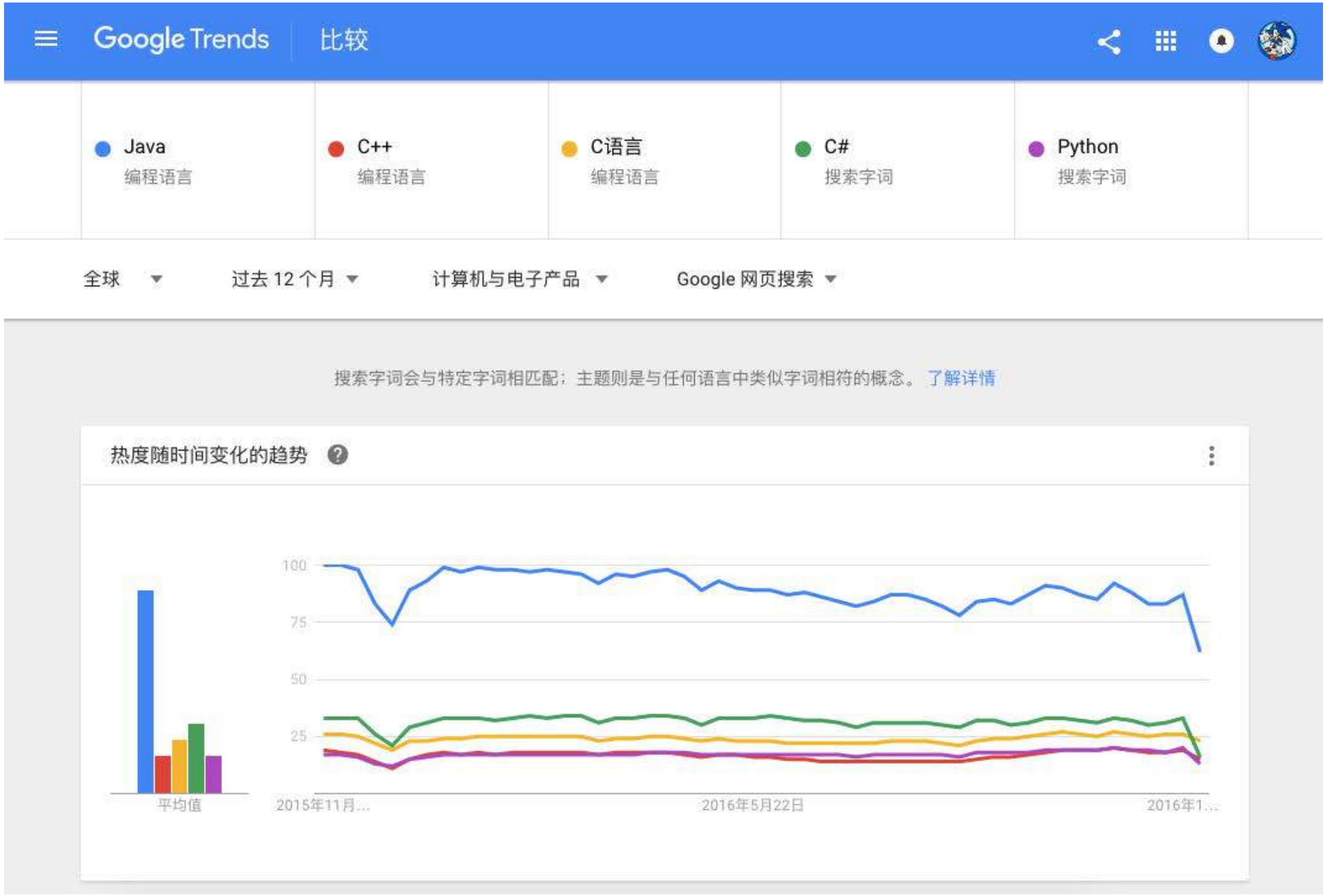
15 most popular languages used on GitHub by opened Pull Request and percentage change from previous period



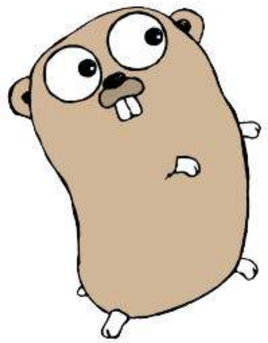
Java依然是利刃

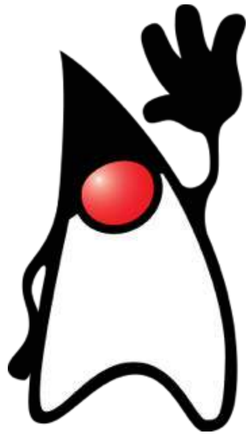
- TIOBE编程语言排行榜
  - Java——2015年度编程语言
  - 长期位居TIOBE榜首
  - 2016年11月TIOBE排行

Nov 2016	Nov 2015	Change	Programming Language	Ratings	Change
1	1		Java	18.755%	-1.65%
2	2		C	9.203%	-7.94%
3	3		C++	5.415%	-0.78%
4	4		C#	3.659%	-0.66%
5	5		Python	3.567%	-0.20%


Google Trends
比较
Java
编程语言
C++
编程语言
C语言
编程语言
C#
搜索字词
Python
搜索字词
全球
过去 12 个月
计算机与电子产品
Google 网页搜索
搜索字词会与特定字词相匹配；主题则是与任何语言中类似字词相符的概念。 了解详情
热度随时间变化的趋势
100
75
50
25
平均值
2015年11月...
2016年5月22日
2016年1...

```go
package main

import (
    "github.com/astaxie/beego"
)

type MainController struct {
    beego.Controller
}

func (this *MainController) Get() {
    this.Ctx.WriteString("hello world!")
}

func main() {
    beego.Router("/", &MainController{})
    beego.Run()
}
```





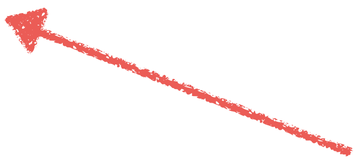
```java
package demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class DemoApplication {
    @RequestMapping("/")
    public String sayHello() {
        return "hello world!";
    }

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

Java依然是利刃

- 但……谁会没事去写Hello World呢?
- 业务逻辑、中间件、框架……总有一款适合你

Java依然是利刃

- 但……谁会没事去写Hello World呢?
- 业务逻辑、中间件、框架……总有一款适合你



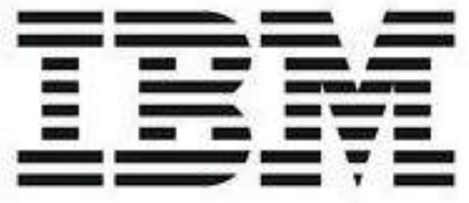
估计大多数人每天都在写业务逻辑吧
语言优势 vs. 工具链 vs. 环境……

生态同样重要

- Java不仅仅是语言
  - Java Programming Language
  - JVM Platform
    - Groovy / Scala / JRuby / Jython / Clojure / Ceylon / Kotlin …

Clojure


ORACLE


IBM

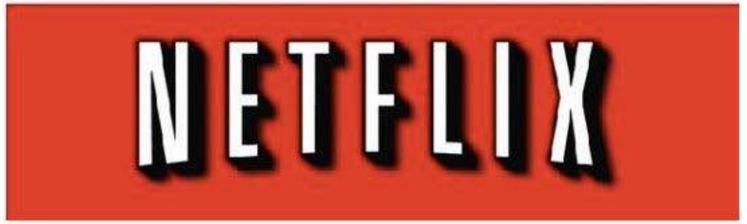

SAP


EMC²


Google


THE
APACHE
SOFTWARE FOUNDATION


Alibaba Group
阿里巴巴集团


NETFLIX


spring by Pivotal


GIAC
BEIJING
Dec.12.16-17

thegiac.com

微服务与语言

微服务与语言

有关系么?

微服务

- 微服务就是一些协同工作的小而自治的服务。
  - 一些
  - 小而自治

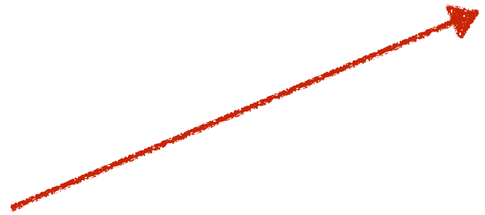
与语言的关系

与语言的关系

- 两者**没有**必然的关系
- 用什么语言都能写出合格的微服务

与语言的关系

- 两者**没有**必然的关系
- 用什么语言都能写出合格的微服务




回复
回复全部
转发
删除
Go为什么是微服务的基石
ArchSummit 北京2016 于2016年9月22日 星期四 上午06:58 发送给

设问句?
反问句?
肯定句?

Go为什么是微服务的基石?

Go为什么是微服务的基石!

设问句?
反问句?
肯定句?

Go为什么是微服务的基石?

Go为什么是微服务的基石!

老生常谈的话题，**没有最好的语言，只有最合适的场景……**

So，大家老拿微服务和语言说来说去是不厚道的

我也得自我检讨:-(

设问句?
反问句?
肯定句?

Go为什么是微服务的基石?

Go为什么是微服务的基石!

老生常谈的话题，**没有最好的语言，只有最合适的场景……**

So，大家老拿微服务和语言说来说去是不厚道的

我也得自我检讨:-(

BTW，我也喜欢Go

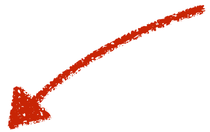
微服务的关注点

微服务的关注点

- 如何设计微服务

微服务的关注点

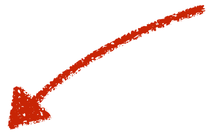
设计，不是开发，不是运维



- 如何设计微服务

微服务的关注点

设计，不是开发，不是运维



- 如何设计微服务
  - Domain-Driven Design
  - Bounded Context
  - 业务上的设计远比代码上的实现来的重要

微服务的关注点

设计，不是开发，不是运维

- 如何设计微服务
  - Domain-Driven Design
  - Bounded Context
  - 业务上的设计远比代码上的实现来的重要

实现上的选择

实现上的选择

实现上的选择

- 只有RESTful服务才是微服务吗?

实现上的选择

- 只有RESTful服务才是微服务吗?
- 只有同步的服务才是微服务吗?

实现上的选择

- 只有RESTful服务才是微服务吗?
- 只有同步的服务才是微服务吗?
- 只有很小的服务才是微服务吗?

实现上的选择

- 只有RESTful服务才是微服务吗?
- 只有同步的服务才是微服务吗?
- 只有很小的服务才是微服务吗?
- 只有用了容器的服务才是微服务吗?

实现上的选择

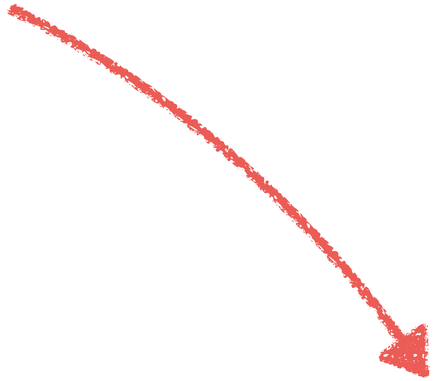
- 只有RESTful服务才是微服务吗?
- 只有同步的服务才是微服务吗?
- 只有很小的服务才是微服务吗?
- 只有用了容器的服务才是微服务吗?
- 只有……

实现上的选择

- 只有RESTful服务才是微服务吗?
- 只有同步的服务才是微服务吗?
- 只有很小的服务才是微服务吗?
- 只有用了容器的服务才是微服务吗?
- 只有……

如果……这个主题只能让你记住一句话

如果……这个主题只能让你记住一句话



如果……这个主题只能让你记住一句话

先从领域模型入手，设计合理的服务，然后再来谈微服务落地。

如果……这个主题只能让你记住一句话

先从领域模型入手，设计合理的服务，然后再来谈微服务落地。

微服务的落地

微服务的落地

- 微服务的落地需要考虑很多问题，比如：

微服务的落地

- 微服务的落地需要考虑很多问题，比如：
  - 服务的实现

微服务的落地

- 微服务的落地需要考虑很多问题，比如：
  - 服务的实现
    - 如何开发

微服务的落地

- 微服务的落地需要考虑很多问题，比如：
  - 服务的实现
    - 如何开发
    - 如何测试

微服务的落地

- 微服务的落地需要考虑很多问题，比如：
  - 服务的实现
    - 如何开发
    - 如何测试

微服务的落地

- 微服务的落地需要考虑很多问题，比如：
  - 服务的实现
    - 如何开发
    - 如何测试
  - 服务的运维

微服务的落地

- 微服务的落地需要考虑很多问题，比如：
  - 服务的实现
    - 如何开发
    - 如何测试
  - 服务的运维
    - 可运维

微服务的落地

- 微服务的落地需要考虑很多问题，比如：
  - 服务的实现
    - 如何开发
    - 如何测试
  - 服务的运维
    - 可运维
    - 高可用

微服务与Java

微服务与Java

进入落地环节

我们不聊什么

- 为什么要选Java
- 细节的技术选型
- 如何用Java编写微服务

我们聊什么

- 写好的服务该怎么测
- 给服务增加些安全感
- 看得见的服务

我们不聊什么

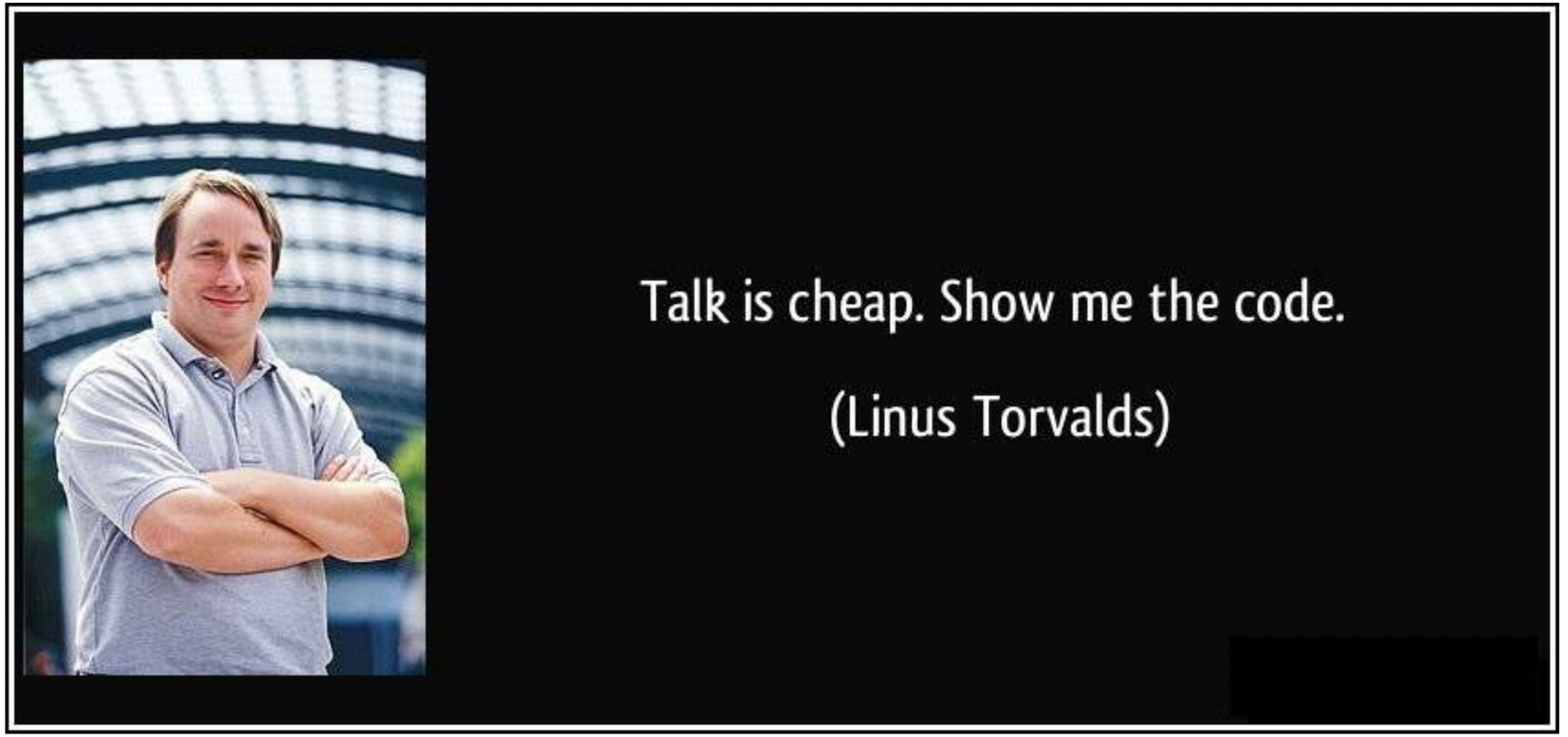
- 为什么要选Java
- 细节的技术选型
- 如何用Java编写微服务

我们聊什么

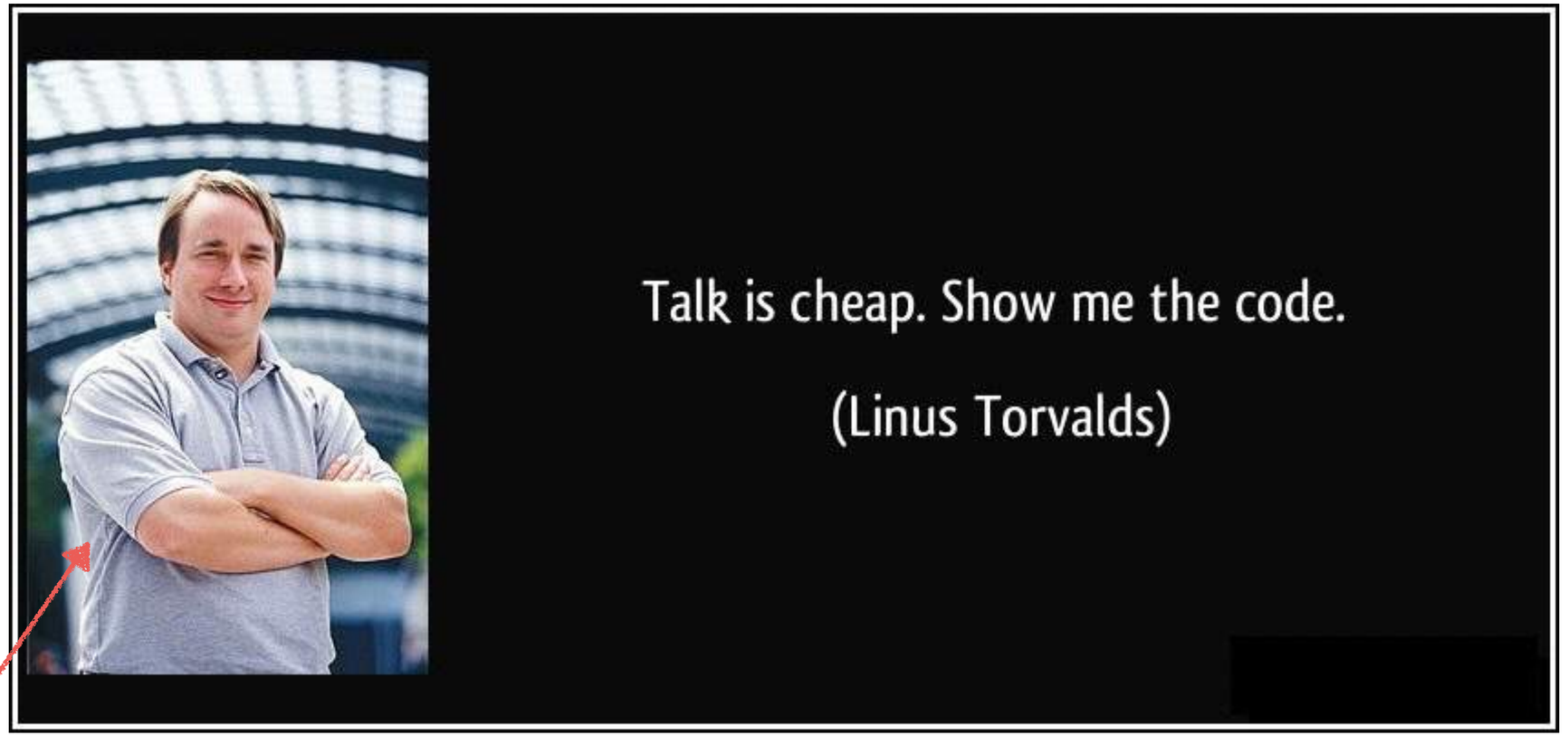
- 写好的服务该怎么测
- 给服务增加些安全感
- 看得见的服务

微服务的测试

- 我们只谈自动化测试，手工测试说多了都是泪
- ~~单元测试，这就不多说了，JUnit人人都会~~
- 集成测试
  - 服务提供方——如何优雅地调用自己
  - 服务调用方——如何模拟被调用方



现场演示时间



我错了
这位大神不写Java:-(

现场演示时间

e.g. 测试“/hi/{name}”服务

以SpringBoot的方式运行测试

```java
@RunWith(SpringRunner.class)
@SpringBootTest
public class SpringCloudDemoServerApplicationTest {
    @Autowired
    private WebApplicationContext wac;

    private MockMvc mockMvc;

    @Before
    public void setUp() throws Exception {
        mockMvc = MockMvcBuilders.webAppContextSetup(wac).build();
    }

    @Test
    public void testHi() throws Exception {
        mockMvc.perform(get("/hi/digitalsonic"))
                .andExpect(status().isOk())
                .andExpect(content().string("Hi digitalsonic"))
                .andExpect(content().contentTypeCompatibleWith(MediaType.TEXT_PLAIN));
    }
}
```

以DSL的形式调用并判断结果

e.g. 测试“/”访问远程“/hi/{name}”服务

不要再Mock接口了
通过Moco来个“真”的服务

偶尔还是需要动点手脚

```java
@RunWith(SpringRunner.class)
@SpringBootTest("remote.server=localhost:12345")
public class SpringCloudDemoSimpleClientApplicationTest {

    @Autowired
    private WebApplicationContext wac;
    @Autowired
    private SpringCloudDemoSimpleClientApplication app;

    private Runner runner; // Moco Runner
    private MockMvc mockMvc;

    @Before
    public void setUp() throws Exception {
        HttpServer server = httpServer(12345);
        server.response("Hi digitalsonic");
        runner = runner(server);
        runner.start();

        mockMvc = MockMvcBuilders.webAppContextSetup(wac).build();
        app.setRestTemplate(new RestTemplate()); // Bypass the @LoadBalanced one
    }

    @After
    public void tearDown() throws Exception {
        runner.stop();
    }

    @Test
    public void testRemoteGreeting() throws Exception {
        mockMvc.perform(get("/"))
                .andExpect(status().isOk())
                .andExpect(content().string("Hi digitalsonic"));
    }
}
```

微服务的测试

- 你可能还会想到
  - 测试覆盖——EclEmma、Cobertura
  - Mock工具——Mockito、EasyMock、PowerMock
  - 性能测试——JMeter
  - 持续集成——Jenkins、CruiseControl
  - 代码扫描——FindBugs、PMD

微服务的防护

- ~~加密与签名，传统的手段，就不提子~~
- ~~DDoS等攻击，属于网络安全范畴，也不提子~~
- 服务的认证与授权