KotlinConf

Kotlin EE
Boost Your Productivity

Marcus Fihlon
@McPringle

Agenda

- Introduction
- Why Kotlin
- Why Java EE
- Kotlin EE
- **Live Coding**
- Wrap-Up

Introduction

Disclaimer

The following presentation has been approved for open audiences only. Hypersensitivity to occasional profanity requires covering ears.

All logos, photos etc. used in this presentation are the property of their respective copyright owners and are used here for educational purposes only. Any and all marks used throughout this presentation are trademarks of their respective owners.

The presenter is not acting on behalf of CSS Insurance, neither as an official agent nor representative. The views expressed are those solely of the presenter.

Marcus Fihlon disclaims all responsibility for any loss or damage which any person may suffer from reliance on this information or any opinion, conclusion or recommendation in this presentation whether the loss or damage is caused by any fault or negligence on the part of presenter or otherwise.

About me

- Software Engineer
  CSS Insurance, Open Source Software
- Agile Coach
  CSS Insurance
- Lecturer
  TEKO Swiss Technical College
- Speaker
  Conferences, User Groups, Meetups
- Author
  Articles, Books
- Community Leader
  Hackergarten, Java User Group Switzerland, Kotlin Swiss User Group, Voxxed Days Zürich

Coming soon*

Kotlin Web Development

Develop full stack web applications with Kotlin and React.js

Packt Publishing
ISBN 978-1-78862-031-4

* around Q2/2018

Buzzword Bingo

- **Monolith**
  Bad by default!
- **Microservices**
  Solve every problem!
- **Nanoservices**
  Solve all other Problems!
- **Docker**
  Just because it's cool!
- **Java EE**
  Uncool and heavy framework for bloated monoliths!

Why Kotlin

About Kotlin

- Statically typed
- Object oriented
- Java compatible
- Easy to learn

- Compiles to Java 6 Bytecode
- Compiles to Java 8 Bytecode
- Transpiles to JavaScript
- Compiles to Native



Kotlin Syntax

```
fun main(args: Array<String>) {
    val event = "KotlinConf"
    println("Hi $event!")
}
```

Classes and Properties

```
class Person(val name: String,
             val age: Int,
             val company: String?) {
    …
}
```

Classes final by default

```
open class Person(val name: String,
                  val age: Int,
                  val company: String?) {
    …
}
```

Data classes

```
data class Person(val name: String,
                  val age: Int,
                  val company: String?)

val marcus = Person("Marcus", 29, null)
println(marcus)     // Person(name=Marcus, age=43, company=null)

val marcusCSS = marcus.copy(company = "CSS Insurance")
println(marcusCSS) // Person(name=Marcus, age=43, company=CSS Insurance)
```

Singletons

```
object PersonController {
    val persons = mutableListOf<Person>()

    fun save(person: Person) {
        persons.add(person)
    }

    …
}
```

Coroutines

```
fun main(args: Array<String>): Unit = runBlocking {
    val jobs = List(10) {
        async(CommonPool) {
            PriceService().price
        }
    }
    println(jobs.sumBy { it.await() } / 10)
}
```

Typesafe Builder

```
val page = html {
    head {
        title { "My Website" }
    }
    body {
        p  { "Hello KotlinConf!" }
    }
}
```

Why Java EE

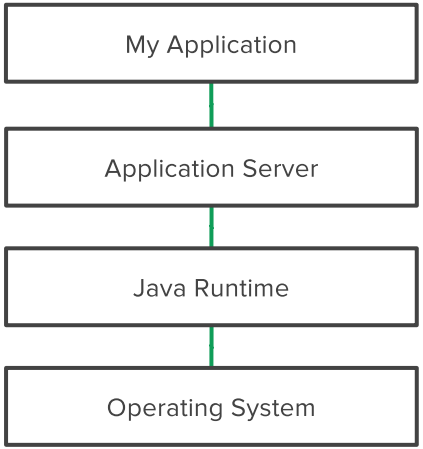
Library Management as a Service

- Servlets, JSTL, EL and JSPs
- **WebSockets**
- JSF
- **JAX-RS**
- EJB lite
- JTA
- **JPA**
- **Bean Validation**
- **CDI**
- **Interceptors**
- JBatch
- **Concurrency**
- JCache
- …

```
dependencies {
   providedCompile "javax:javaee-api:$javaee_version"
}
```

Ready to use containers

```
FROM payara/micro
COPY myapplication.war /opt/payara/deployments
```



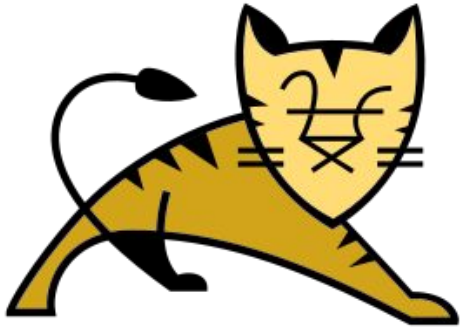
changes on every build

changes frequently

changes sometimes, if not delayed

changes very infrequently

It's a Standard!



Kotlin EE

Classes final by default

```
buildscript {
    dependencies {
        classpath "org.jetbrains.kotlin:kotlin-allopen:$kotlin_version"
    }
}

allOpen {
    annotation('javax.ejb.Stateless')
    annotation('javax.ws.rs.Path')
}
```

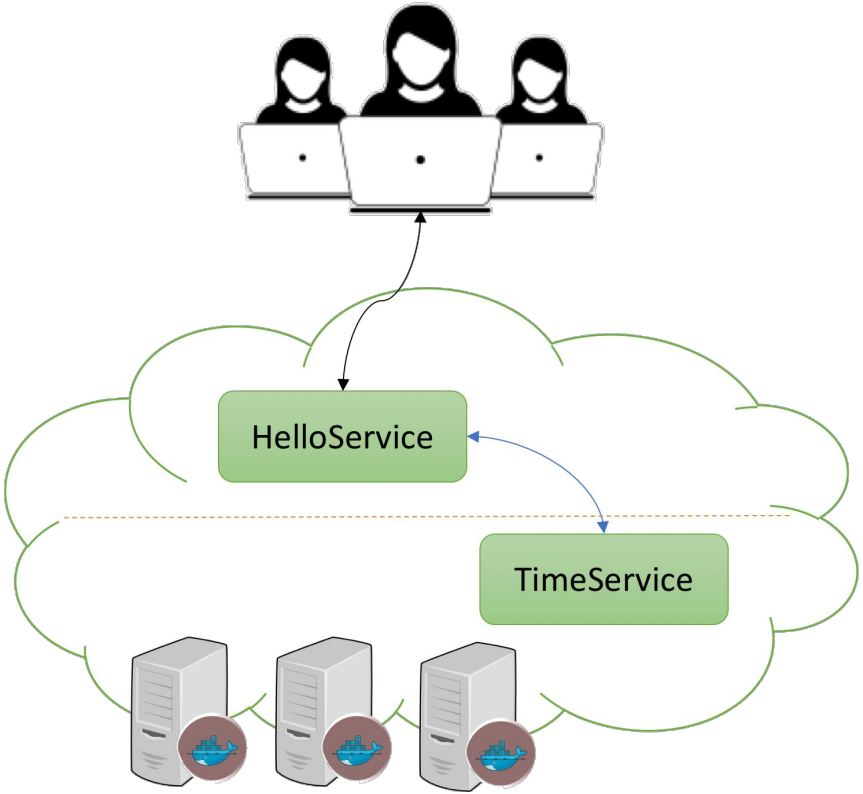
Zero-Argument Constructor

```
buildscript {
    dependencies {
        classpath "org.jetbrains.kotlin:kotlin-noarg:$kotlin_version"
    }
}

noArg {
   annotation('javax.ws.rs.Path')
}
```

Live Coding

Architecture



Prepare the Swarm

- **List all available machines**
  `docker-machine ls`
- **Create a new machine to be used as a manager node**
  `docker-machine create -d virtualbox mgr`
- **Create a new machine to be used as a worker node**
  `docker-machine create -d virtualbox node01`
- **Login to the manager**
  `docker-machine ssh mgr`
- **Initialize the swarm**
  `docker swarm init --advertise-addr 192.168.99.100`

Prepare the Visualizer

- **Start the visualizer service**

```
docker run -it -d -p 8080:8080 -e HOST=192.168.99.100 \
-v /var/run/docker.sock:/var/run/docker.sock \
--name visualizer dockersamples/visualizer
```

- **Open the visualizer service in a web browser**

  http://192.168.99.100:8080/

Join the Swarm

- **Ask the manager node for the token to join the swarm**
  `docker swarm join-token worker`
- **Login to the worker node**
  `docker-machine ssh node01`
- **Join the worker to the swarm**
  `docker swarm join --token <token> 192.168.99.100:2377`

Verify the Swarm

- **Login to the manager**
  `docker-machine ssh mgr`
- **List all nodes**
  `docker node ls`
- **Show more detailed information**
  `docker info`

Create a Network

- **Login to the manager**

  ```
  docker-machine ssh mgr
  ```
- **Create a new overlay network**

  ```
  docker network create -d overlay kotlinconf
  ```
- **List all networks**

  ```
  docker network ls
  ```

Deploy Services

- **Login to the manager**

```
docker-machine ssh mgr
```

- **Deploy the time service**

```
docker service create --network kotlinconf \
--name timeservice mcpringle/timeservice
```

- **Deploy the hello service**

```
docker service create --network kotlinconf -p 8181:8080 \
--name helloservice mcpringle/helloservice
```

- **List all services**

```
docker service ls
```

Testing, Scaling and Maintenance

- **Testing our services**
  ```
  http http://192.168.99.100:8181/api/hello/kotlinconf
  ```
- **Scaling services up and down**
  ```
  docker service update --replicas 3 helloservice
  ```
- **Take a node out of service**
  ```
  docker node update --availability drain mgr
  ```
- **Take a note back into service**
  ```
  docker node update --availability active mgr
  ```

Wrap-up

Conclusion

With Kotlin, Java EE and Docker you get:

- Easy to understand code
- Fast build times
- Reproducible deployments
- Easy scaling of your services

You are fast and efficient because you have your focus on your business code.

You create business value!

You ever dreamed of living in Switzerland?

Code Warriors wanted!

About you

- You are frighteningly awesome at what you do.
- You can perform quick and deadly tactical strikes, as well as feats of epic badassery. Sometimes both at the same time.
- You would rather refactor existing, mostly working, ugly code instead of rewriting it from scratch. OK nobody would really rather do that, but you know it's the right and honorable thing to do most of the time.
- You know where your towel is.

About us

- Biggest Basic Health Insurance Company
- 2700 Employees
- 300 IT Professionals
- 24 Scrum Teams
- 130 Software Developers
- Flexible working times
- Personal development support
- Very good social benefits
- 6 weeks vacation
- Located in Lucerne, Switzerland

Contact me: marcus.fihlon@css.ch

Thank you!

http://fihlon.ch/kotlinconf17