Spring Boot – framework for micro services

Jakub Kubryński
jk@devskiller.com
@jkubrynski

whoami

DevSkiller

bottega IT SOLUTIONS

confitura'14

History

- 1999 J2EE 1.2
- 2001 xDoclet 1.0
- 2004 Spring Framework 1.0
  - Injection
  - POJO oriented
  - AOP & transactions
- 2006 Java EE 5

'Classic' way

- hundreds of thousands LOC
- thousands of tests ... or not
- hundreds of issues in jira
- and a lot of design patterns

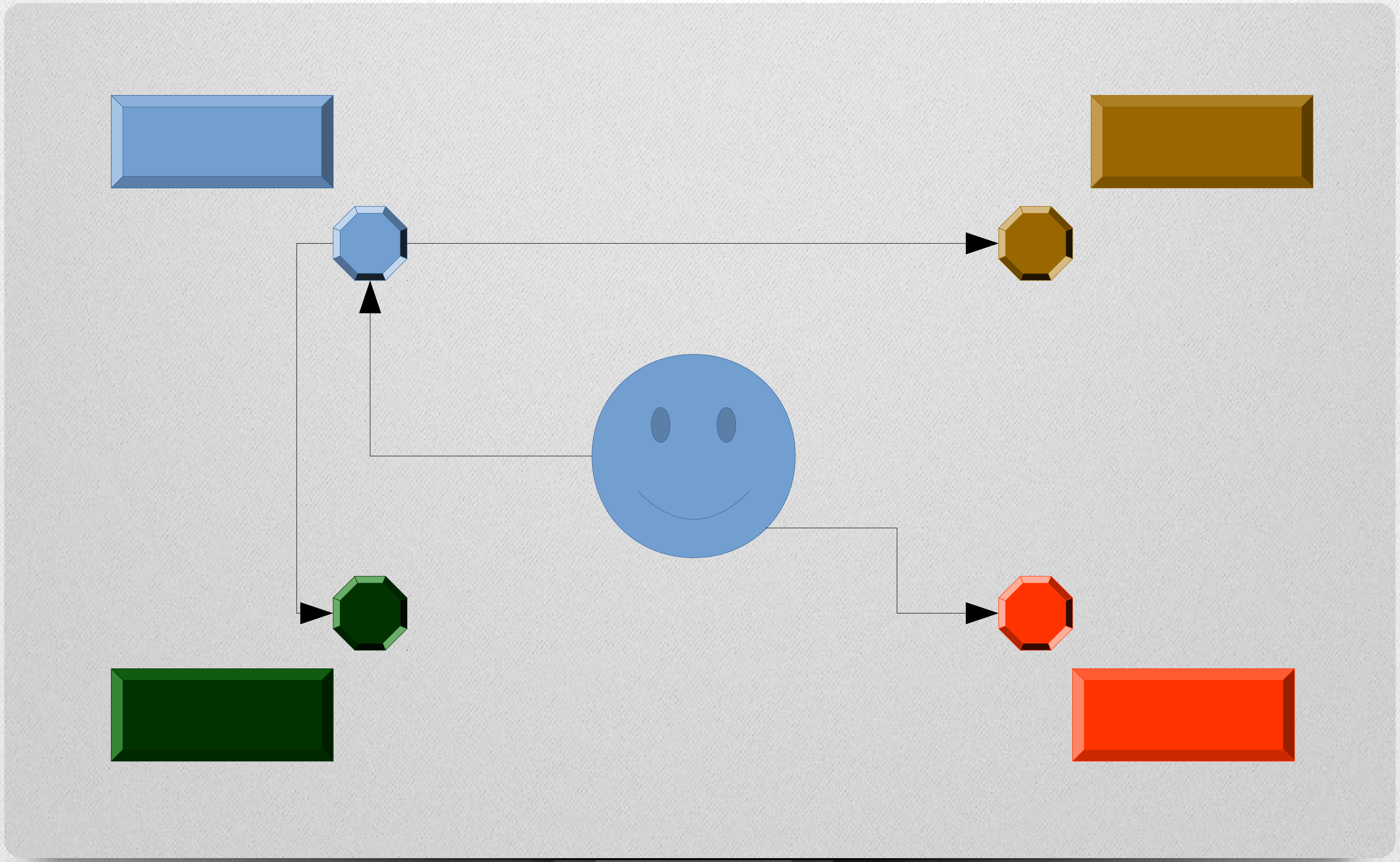
... lava flow, big ball of mud, orgy, yo-yo

'Classic' way

Micro way

- Single responsibility
- Loosely coupled
- Reliable
- Small, light

Micro way



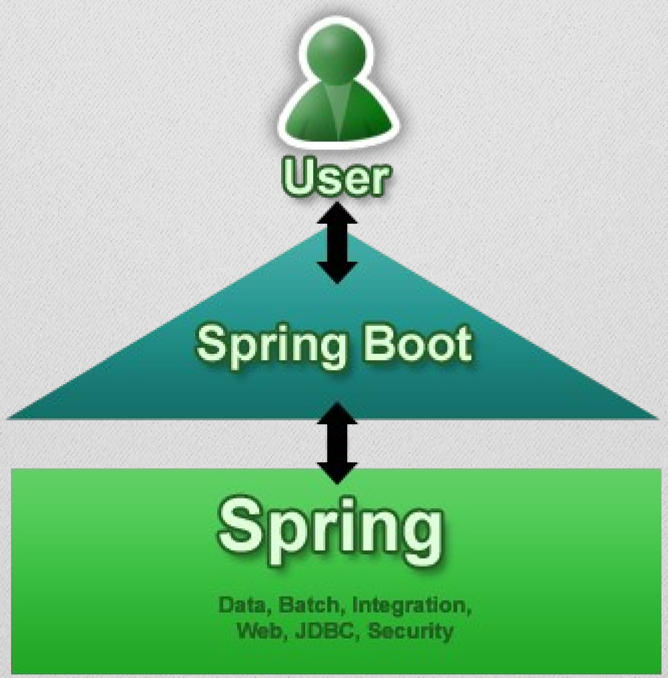
Hard?

- Versioning
- Integration testing
- Boilerplate bootstrap code

History

- 1999 J2EE 1.2
- 2001 xDoclet 1.0
- 2004 Spring Framework 1.0
  - Injection
  - POJO oriented
  - AOP & transactions
- 2006 Java EE 5
- 2013 Spring Boot!

Focus



source: spring.io

Revolution

```
@RestController
@EnableAutoConfiguration
public class HelloWorld {

  @RequestMapping("/")
  public String helloWorld() {
    return "Hello World!";
  }

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

}
```

Key features

- Stand-alone Spring applications
- Embedded Tomcat or Jetty
- Starter dependencies
- Automatic configuration
- Production-ready environment
- No code generation / no XML config

Blocks

- SpringApplication
- @EnableAutoConfiguration
- @ConditionalOnClass
- @ConditionalOnBean
- @ConditionalOnExpression

Sample auto-configuration

```
@Configuration

@ConditionalOnClass({ MBeanExporter.class })

@ConditionalOnMissingBean({ MBeanExporter.class })

@ConditionalOnExpression("${spring.jmx.enabled:true}")

public class JmxAutoConfiguration {

  ...

}
```

Starters

- spring-boot-starter
- spring-boot-starter-web
- spring-boot-starter-test
- spring-boot-starter-actuator

Starters

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
</dependency>

<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
```

Production ready

- Monitoring endpoints
  - /health
  - /info
  - /metrics
  - /mappings
- JMX / SSH
- Auditing

Properties

```
@ConfigurationProperties(prefix=”mail”)
public class MailProperties {
  private InetAddress serverAddress;
  private Resource template;
}


mail.serverAddress : 84.123.456.32
mail.template : classpath:mail.vm
```

Profiles

- spring.profiles.active = production,mysql
- configuration per profile:
  - application-production.properties
  - conference-test.properties

Logging

- log4j
- logback
- java.util.Logging

Security

- spring-boot-starter-security
- @SecurityAutoConfiguration
- @SecurityProperties
  - security.requireSsl = true

WAR

```
public class WebInit extends SpringBootServletInitializer {


    @Override
    protected SpringApplicationBuilder
         configure(SpringApplicationBuilder application) {
        return application.sources(SampleApplication.class);
    }


}
```

Tests

@SpringApplicationConfiguration(classes =
Application.class)

@ContextConfiguration(classes = Application,
loader = SpringApplicationContextLoader)

@IntegrationTest

It's Spring

How it helps ?

- Dramatically reduces boilerplate code
- Enables polyglot
- Simplifies integration testing
- Simplifies environment maintenance

You have questions

I (probably) have answers

END! THANK YOU