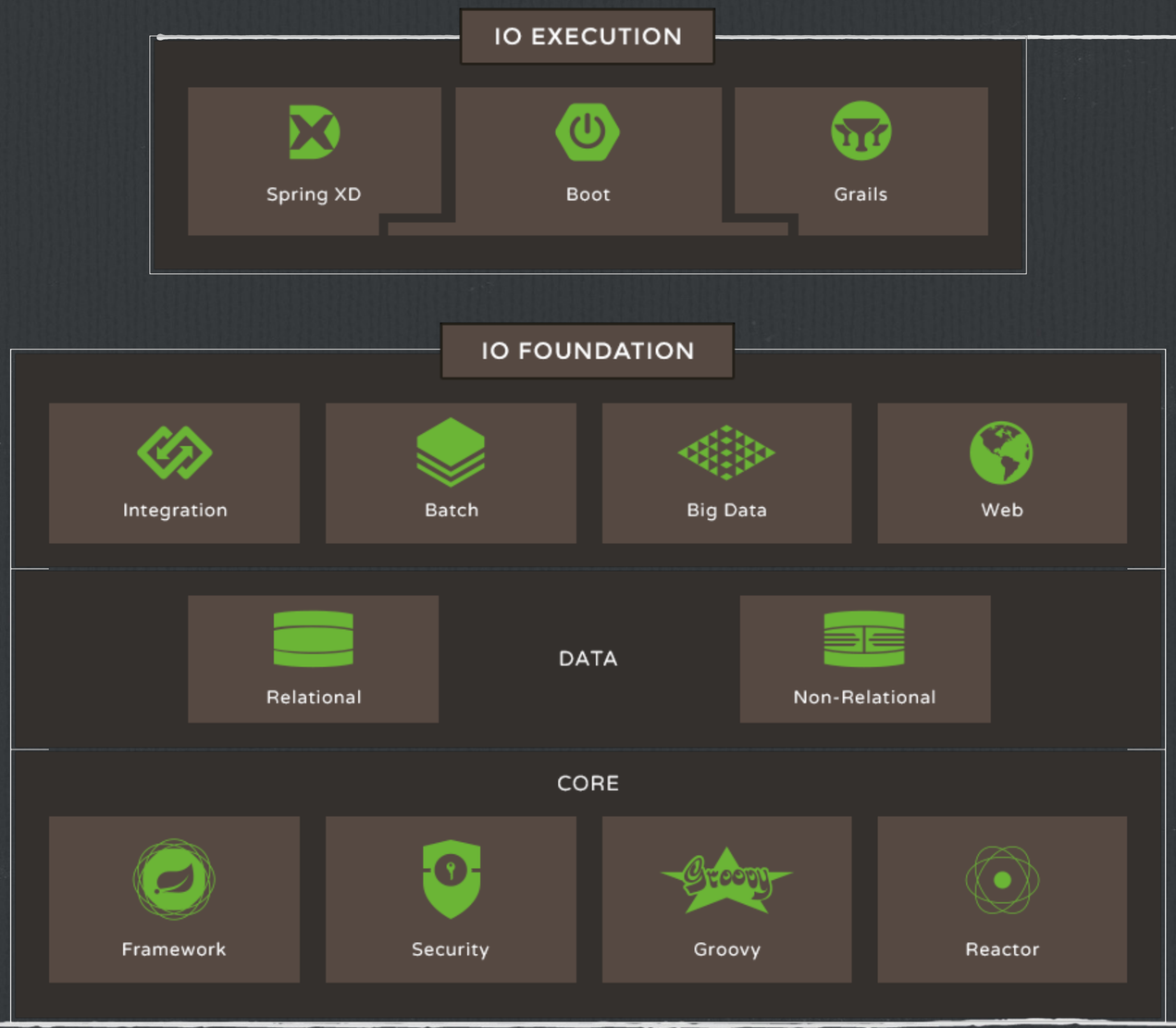
Spring Boot

Spring news

SpringOne 2GX 2013-4 - Spring Platform reinvented

1. Spring 4.x - java8 , generic/ordered injection, @Conditional, AsyncRestTemplate, webSocket
2. Spring security Oauth 2.x.x - OAuth (1a) & OAuth2 by standard Spring Security
3. Spring XD - runtime environment for managing Big Data (by DSL - stream, sinks, producers)
4. Springs IO - dependency management solution
5. Spring Boot
6. Spring IO platform - Big Things Come in Small (Java) Packages

Spring IO platform



Execution tier

DSR - Domain Specific Runtimes

for apps build on IO foundation

Goal

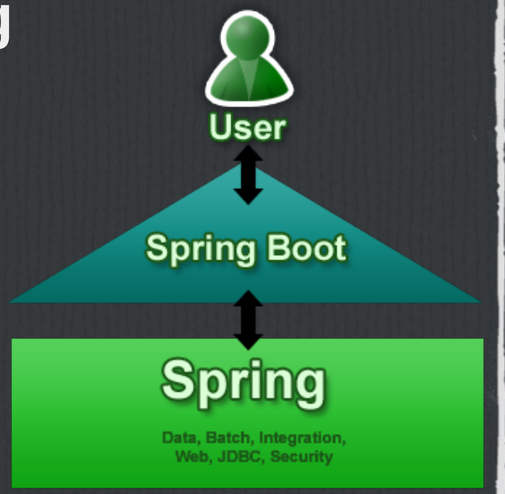
- Eliminate boilerplate configuration

Result

- Manages things that runs as processes on OS

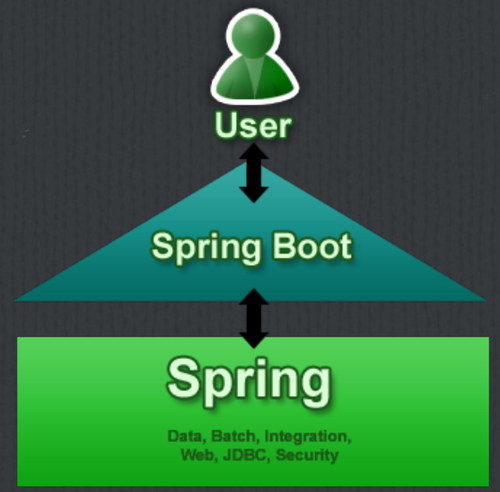
Spring Boot

- A tool for getting started (Bootstrap and Run) with the Spring based application as fast as possible (and opinionated)
- harmonizing (well tested) dependencies
- Non-functional features for free
- If you not ok with defaults you can change it quickly



Spring Boot

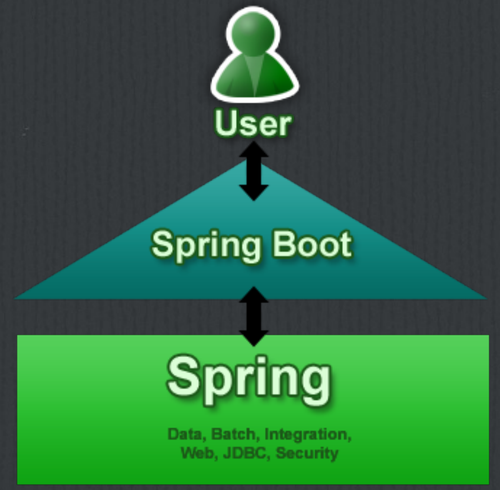
- no code generation
- no XML
- no aspects
- can run like java -jar (can be build as deployable war if necessary)
- convetion over configuration : @autoconfig + starters
- needs java 1.6+, Maven 3 or Gradle 1,6+



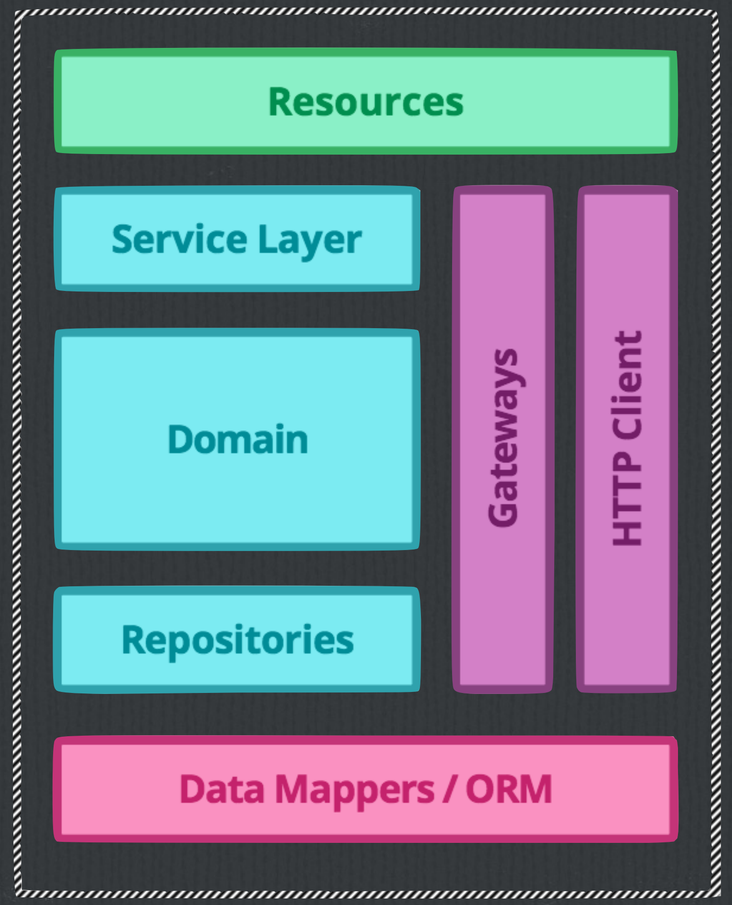
Spring Boot

Final configurations

	Pom.xml	Configuration	Properties
PetClinic	377 lines 77 imported artifacts	9 XML files 350 lines	2 files 17 lines
Spring Boot	117 lines 79 imported artifacts	6 JavaConfig classes 130 lines	1 file 10 lines



Tons of micro services like …



Network boundary

Protocol (message to domain objects)

Domain

External

Persitence

Dropwizard

- Create production ready (micro)service easily
- Exposed by HTTP (REST)
- Modules with “all-you-need” features
  - Jetty, Jersey Rest client, Authentication, metrics, single-jar deployment, slf4j, … all wrapped in !
- Configuration of (almost) everything in the single YAML properties file

Spring Boot

Provides easy way to :

- Bootstrap
  - Starters, AutoConfiguration
- Configuration
  - Properties
- Runtime
  - Actuator: Alerting, Logging, Monitoring, Metrics
- Test

Bootstrap - autoconfig - example

Bootstrap - autoconfig

- Pre-Configured beans
  - spring-boot-autoconfigure library
  - Applied on condition
  - starters to enable autoconfiguration
- @EnableAutoConfiguration
  - On main class
  - Excluding autoconfig
  - processing spring.factories file

```
@Configuration
@ConditionalOnClass(Mongo.class)
@EnableConfigurationProperties(MongoProperties.class)
@ConditionalOnMissingBean(MongoDbFactory.class)
public class MongoAutoConfiguration {

    @Autowired
    private MongoProperties properties;

    @Autowired(required = false)
    private MongoClientOptions options;

    private Mongo mongo;

    @PreDestroy
    public void close() {
        if (this.mongo != null) {
            this.mongo.close();
        }
    }

    @Bean
    @ConditionalOnMissingBean
    public Mongo mongo() throws UnknownHostException {
        this.mongo = this.properties.createMongoClient(this.options);
        return this.mongo;
    }

}
```

Bootstrap - custom autoconfig

Bootstrap - starters & @autoconfig

Configuration

- Properties
  - Relaxed binding
  - @ConfigurationProperties classes
  - Order (Command line, Jndi, Sys, Env, configFile)
  - profiles
    - YAML
    - application-${profile}.properties
  - configprops actuator

Runtime - How to Start

SpringApplication.run(ConfigClass.class, ‘args’)

SpringApplicationBuilder(ConfigClass.class).run(args)

```
new SpringApplicationBuilder(ParentConfiguration.class)
    .profiles("adminServer", "single")
    .child(AdminServerApplication.class)
    .web(true)
    .registerShutdownHook(false)
   .run(args);
```

- producing ConfigurableApplicationContext

Runtime - What really happened ?

`new SpringApplication(ConfigClass.class)`

1. Deduce that java.server.Servlet is on CLASSPATH
   - produce different context than
2. Set initializers
   - programmatic init of the context (own listeners) before context.refresh()
3. Set listeners
   - listen to the events (OnApplicationEvent)
4. Deduce Main class
   - with help of stack trace :) ; mainly for logging purposes (starting/started)

Runtime - What really happened ?

new SpringApplication(ConfigClass.class).run(args):

1. Prepare Environment
    - system, args, propertySources
2. Initialize context
3. Register shutdown hook
4. Refresh context
    - refresh all singleton beans
    - before run initializers, after run commandLineRunners

Runtime - Jar compression

- its not shaded jar
  - hard to see which jar dependency is used

„myapp.jar


```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <version>1.2.0.BUILD-SNAPSHOT</version>
            <executions>
                <execution>
                    <goals>
                        <goal>repackage</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

Runtime - Actuator

- Rest Endpoints
- JMX endpoints (can be accessed over HTTP with Jolokia)
- Remote Shell (SSH, Telnet) - CrasH

Runtime - Actuator

- Audit
  - AuditEventRepository
- Trace endpoint
- Trace repository

Runtime - Remote Shell

- CraSH console (over SSH/telnet)

Runtime - Extract PID

- [ ] run as a service
- [ ] shell script

Runtime - Command Line Runner

- ☐ Starts right after context.refresh
- ☐ have access to args

Testing

- @IntegrationTest
- @WebAppConfiguration
- Integration tests:
  - @IntegrationTest(“server.port:0”)
  - @Value("${local.server.port}")

Testing

- EnvironmentTestUtils.addEnvironment(env, "org=Spring", “name=Boot");
- TestRestTemplate
  - ignoring cookies
  - not following redirects
  - server-side exception throwing

Spring Cloud

IO Coordination- Spring Cloud

- Event bus
- Service discovery
- Global locks
- Leader election
- Distributed config
- Intelligent routing
- Cluster state

Spring Cloud

- ☐ Distributed/versioned configuration
- ☐ Service registration and discovery
- ☐ Routing
- ☐ Service-to-service calls
- ☐ Load balancing
- ☐ Circuit Breaker
- ☐ Asynchronous
- ☐ Distributed messaging

Examples

- Github :
  - https://github.com/To-da/spring-boot-examples
  - https://github.com/To-da/spring-example-conditional
  - https://github.com/spring-projects/spring-boot/tree/master/spring-boot-samples
- Youtube :
  - http://youtu.be/lSEnslwhoFU
  - http://youtu.be/EUJ8OO6zPvM
  - http://youtu.be/fEuXdeJPY7U
  - http://youtu.be/i-vSIkMHskI