Pivotal

Cloud Native Workshop

Spring Boot and Actuator

Pivotal Cloud Foundry

Pivotal

Agenda

1. Challenges building non-Boot applications
2. What is Spring Boot?
3. Capabilities

Pivotal

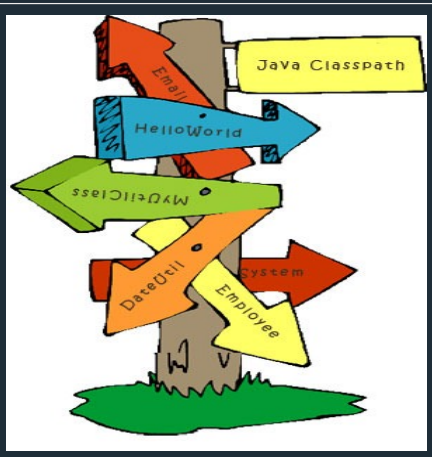
Agenda

1. Challenges building non-Boot applications
2. What is Spring Boot?
3. Capabilities

Pivotal

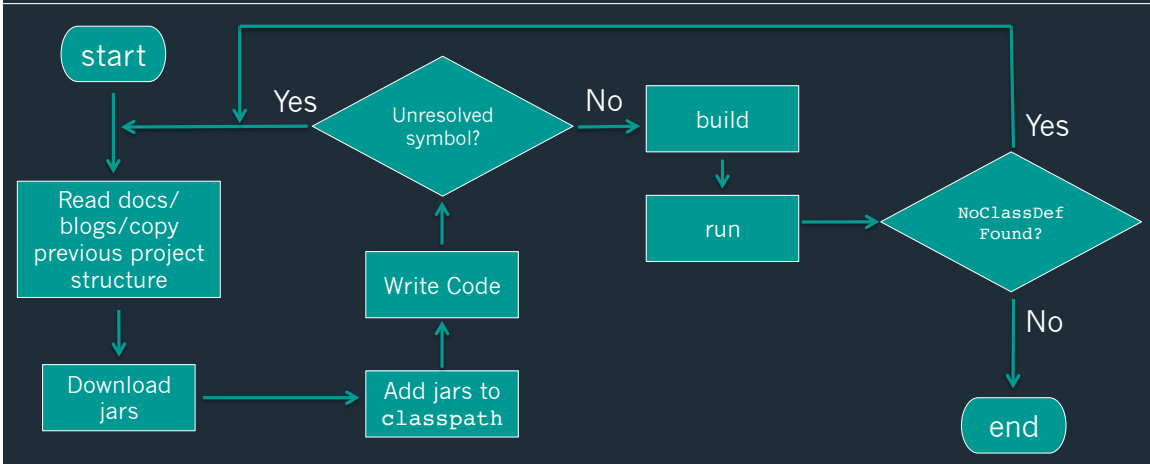
`Classpath` Hell

- A Jar is missing
- There is one Jar too many
- A class is not visible where it should be



Pivotal

Resolving `Classpath` Hell



Pivotal

Boilerplate code

Every application that accesses a relational database with `JDBC` needs to configure a `JdbcTemplate` and a `DataSource`

```
@Bean
public JdbcTemplate jdbcTemplate(
        DataSource dataSource) {
    return new JdbcTemplate(dataSource);
}
```

```
@Bean
public DataSource dataSource() {
    return new EmbeddedDatabaseBuilder()
            .setType(EmbeddedDatabaseType.H2)
            .addScripts("schema.sql", "data.sql")
            .build();
}
```

Pivotal

Boilerplate code

To get a db connection

- Set Driver
- Set Connection URL
- Set Username
- Set Password
- Catch Exceptions

```
public static Connection getConnection() {
    if (dbConnection != null) {
        return dbConnection;
    } else {
        try {
            InputStream inputStream =
                    DbUtil.class.getClassLoader().
                    getResourceAsStream("db.properties");
            Properties properties = new Properties();
            if (properties != null) {
                properties.load(inputStream);

                String dbDriver = properties.getProperty("dbDriver");
                String connectionUrl = properties.getProperty("connectionUrl");
                String userName = properties.getProperty("userName");
                String password = properties.getProperty("password");

                Class.forName(dbDriver).newInstance();
                dbConnection = DriverManager.
                        getConnection(connectionUrl, userName, password);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return dbConnection;
    }
}
```


Pivotal

Boilerplate code

For CRUD Operations

- Prepare Statement
- Set Parameters
- Execute Statement
- Catch Exceptions

```
public void save(String userName, String password, String firstName,
        String lastName, String dateOfBirth, String emailAddress) {
    try {
        PreparedStatement prepStatement = dbConnection.prepareStatement(
                "insert into student(userName, password, " +
        "firstName, lastName, dateOfBirth, emailAddress) " +
                        "values (?, ?, ?, ?, ?, ?)");
        prepStatement.setString(1, userName);
        prepStatement.setString(2, password);
        prepStatement.setString(3, firstName);
        prepStatement.setString(4, lastName);
        prepStatement.setDate(5, new java.sql.Date(
                new SimpleDateFormat("MM/dd/yyyy").
                parse(dateOfBirth.substring(0, 10)).getTime()));
        prepStatement.setString(6, emailAddress);

        prepStatement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (ParseException e) {
        e.printStackTrace();
    }
}
```


Pivotal

Agenda

1. Challenges building non-Boot applications
2. What is Spring Boot?
3. Capabilities

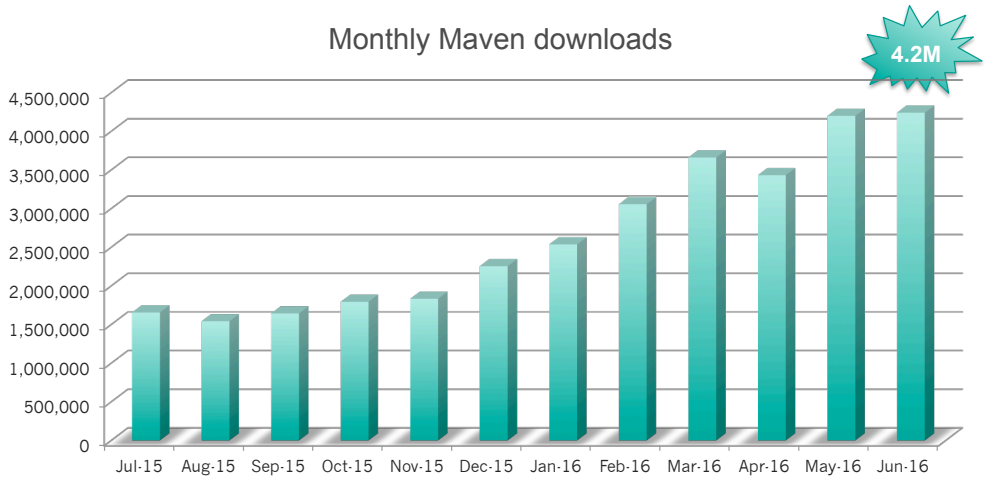
Pivotal

Spring Boot

Spring Boot is an opinionated framework to simplify bootstrapping and development of new Spring Applications

Pivotal

Spring Boot Adoption



Source: oss.sonatype.org

Agenda

1. Challenges building non-Boot applications
2. What is Spring Boot?
3. Capabilities

Pivotal

Capabilities

- Quick start project generation
- Automatic project dependency management
- Configuration drift prevention
- Conditional configuration

Pivotal

Capabilities

- Developer Productivity Tooling
- Auto-configuration
- Monitoring and management endpoints
- Microservices-friendliness

Pivotal

Spring Initializr

(Quick start project generation)

Pivotal

Spring Initializer

Spring Initializr is a configurable service to consistently and easily generate a quick start project

Pivotal

Spring Initializer

- Generates a Spring Boot project structure
- Provides a Maven/Gradle build specification
- Doesn't generate application code
- You can customize the Spring Initialzr
  - `https://github.com/spring-io/initializr/`

Pivotal

Spring Boot starters

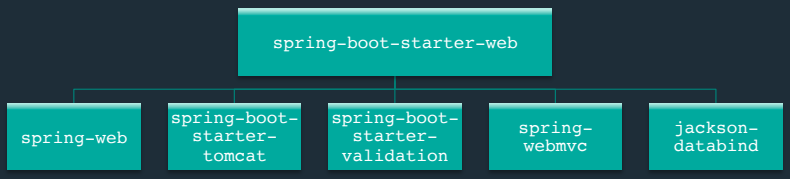
(Automatic project dependency management)

Pivotal

© Copyright 2015 Pivotal. All rights reserved.

Spring Boot starters

- Are virtual packages deployed to Maven central
- They pull in other dependencies while containing no code of their own



Pivotal

Profiles

(Conditional configuration)

Pivotal

Profiles

Segregate parts of the application configuration and make it available in certain environments via

- Annotations
- Properties file

Pivotal

Profiles

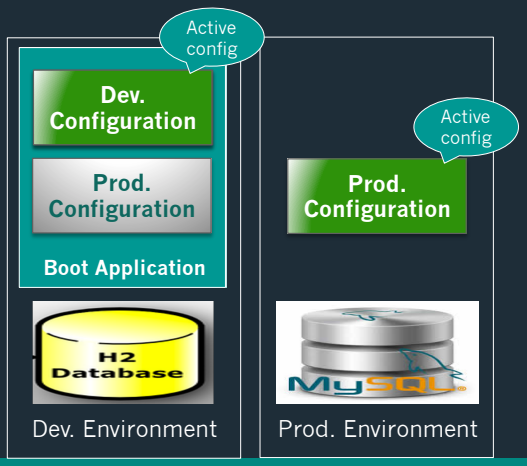
Mark `@Component` or `@Configuration` with `@Profile` to limit when it is loaded.

```
@Configuration
@Profile("development")
public class DevelopmentConfiguration {
    // The @Profile requires that the "development"
    // profile be active at runtime
    // for this configuration to be applied
    // Activate this profile in the properties
    // file with spring.profile.active=development
    // command line with --spring.profiles.active=development
}
```

```
@Configuration
@Profile("production")
public class ProductionConfiguration {
    // The @Profile requires that the "production"
    // profile be active at runtime
    // for this configuration to be applied
    // Activate this profile in the properties
    // file with spring.profile.active=production
    // command line with --spring.profiles.active=production
}
```

Pivotal

Profiles



Checkout code from source control

Build an executable jar

Configuration activated based on active profile

Deploy anywhere

No Code Changes

No Configuration Changes

It Just Works!

Pivotal

Precedence of externalized configuration

1. Command line arguments.
2. Properties from `SPRING_APPLICATION_JSON` (inline JSON embedded in an environment variable or system property)
3. JNDI attributes from `java:comp/env`.
4. Java System properties (`System.getProperties()`).
5. OS environment variables.
6. A `RandomValuePropertySource` that only has properties in `random.*`.
7. Profile-specific application properties outside of your packaged jar (`application-{profile}.properties` and YAML variants)
8. Profile-specific application properties packaged inside your jar (`application-{profile}.properties` and YAML variants)
9. Application properties outside of your packaged jar (`application.properties` and YAML variants).
10. Application properties packaged inside your jar (`application.properties` and YAML variants).
11. `@PropertySource` annotations on your `@Configuration` classes.
12. Default properties (specified using `SpringApplication.setDefaultProperties`).

Pivotal

Actuator

(Monitoring & Management Endpoints)

Pivotal

Actuator

Offers production-ready features such as monitoring and metrics to improve operator efficiency

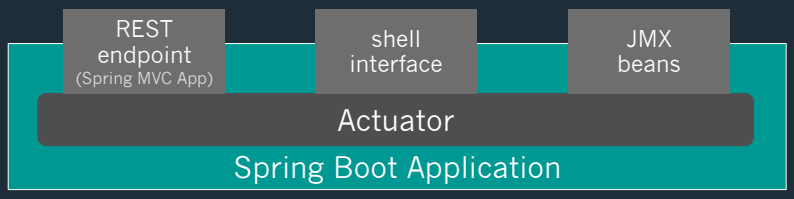
```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

Pivotal

Actuator

Actuator Services are exposed through

- HTTP endpoints
- shell interface
- JMX Beans



Pivotal

Types of Actuator Endpoints

Configuration
- `/beans`
- `/autoconfig`
- `/env`
- `/configprops`
- `/controller`

Metrics
- `/metrics`
- `/trace`
- `/dump`

Miscellaneous
- `/shutdown`
- `/info`

Pivotal

/health

Reports health metrics for the application

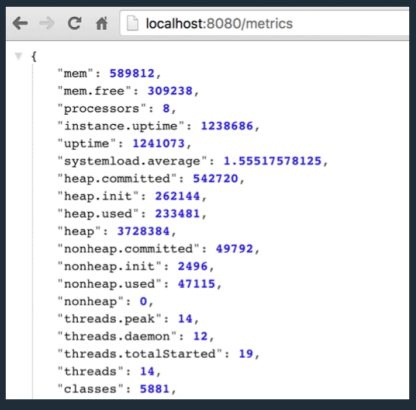
- Each check returns a Health object
  - status
    - UP
    - DOWN
    - UNKONWN
    - OUT_OF_SERVICE
- Plugin your own health definitions

Pivotal

/metrics

Reports application metrics such as

- Memory usage and
- HTTP request counters



Pivotal

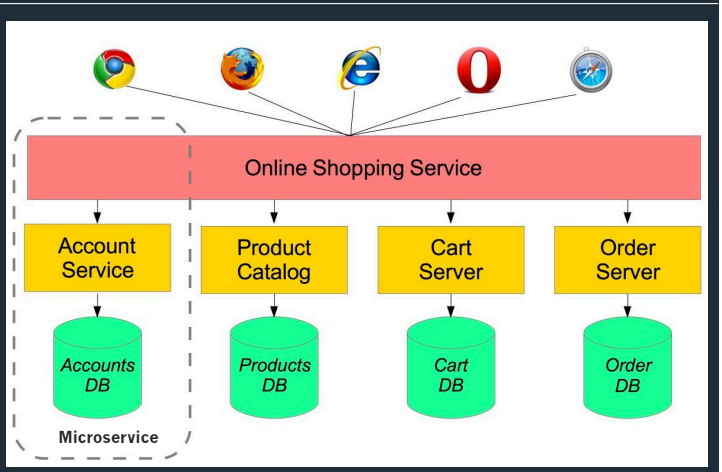
Microservices

(Microservices-friendliness)

Pivotal

Microservices

Each microservice is a Spring Boot fat jar



Pivotal

© Copyright 2015 Pivotal. All rights reserved.

Pivotal

Thank You