Configuration

Bryan Hansen
twitter: bh5k
http://www.linkedin.com/in/hansenbryan

Configuration

Database

- **JPA is an abstraction layer for our database**
- **We can use almost any Relational Database that we have a Dialect for**
  - More about dialects later
- **For our class we are going to use MySQL**
  - http://www.mysql.com/downloads/installer/5.6.html
- **For development:**
  - Username = root
  - Password = password

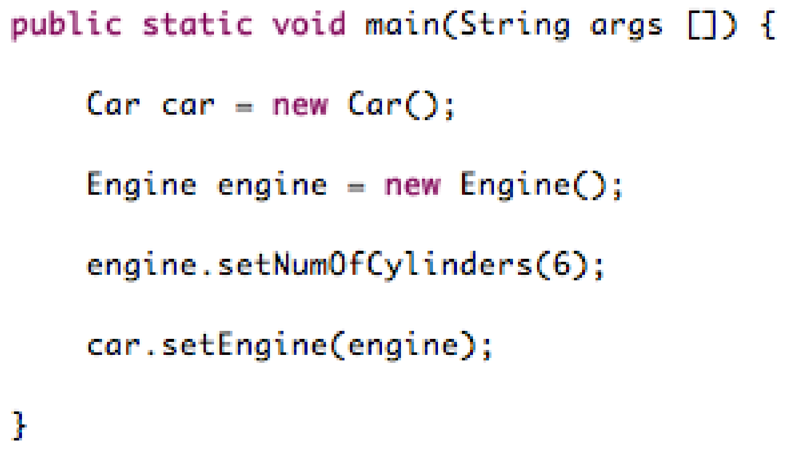
Maven

- Jars required for JPA

```xml
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.21</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>4.1.9.Final</version>
</dependency>
<dependency>
    <groupId>javax.transaction</groupId>
    <artifactId>jta</artifactId>
    <version>1.1</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>3.2.0.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-orm</artifactId>
    <version>3.2.0.RELEASE</version>
</dependency>
```

Plain Java Code

- **All of this code can be configured using Java:**



```
public static void main(String args []) {
    Car car = new Car();
    Engine engine = new Engine();
    engine.setNumOfCylinders(6);
    car.setEngine(engine);
}
```

Spring XML Config Setter Injection

- The same code configured using XML and Spring:

```xml
<bean name="car" class="com.pluralsight.model.Car">
    <property name="engine" ref="engine" />
</bean>

<bean name="engine" class="com.pluralsight.model.Engine">
    <property name="numOfCylinders" value="6" />
</bean>
```

Autowiring Setters

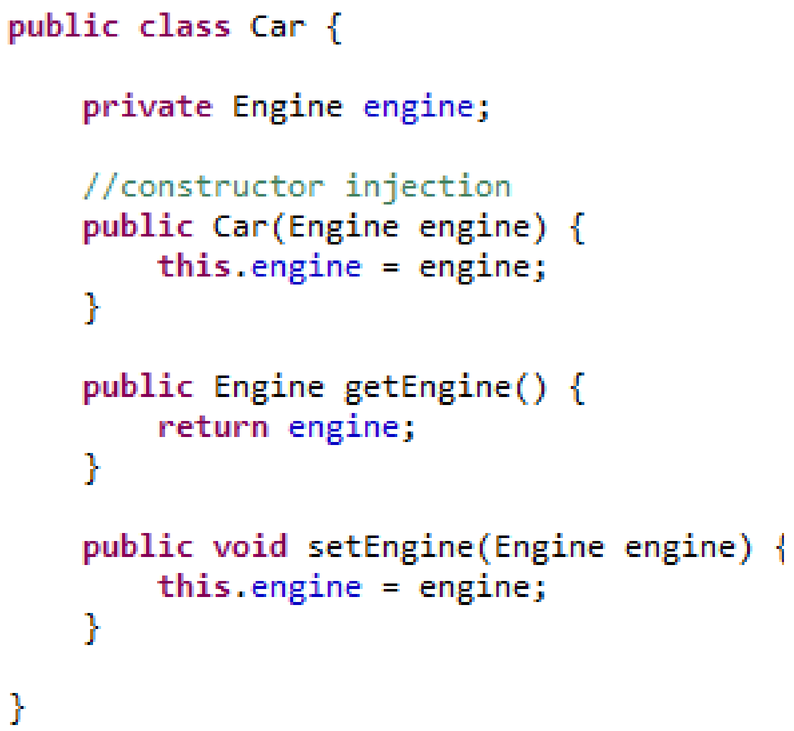
- Instead of using XML, we can also Autowire using Annotations:

```java
@Component
public class Car {

    @Autowired
    private Engine engine;

    //public no args constructor
    public Car() {

    }

    public Engine getEngine() {
        return engine;
    }

    //setter used for injection
    public void setEngine(Engine engine) {
        this.engine = engine;
    }

}
```

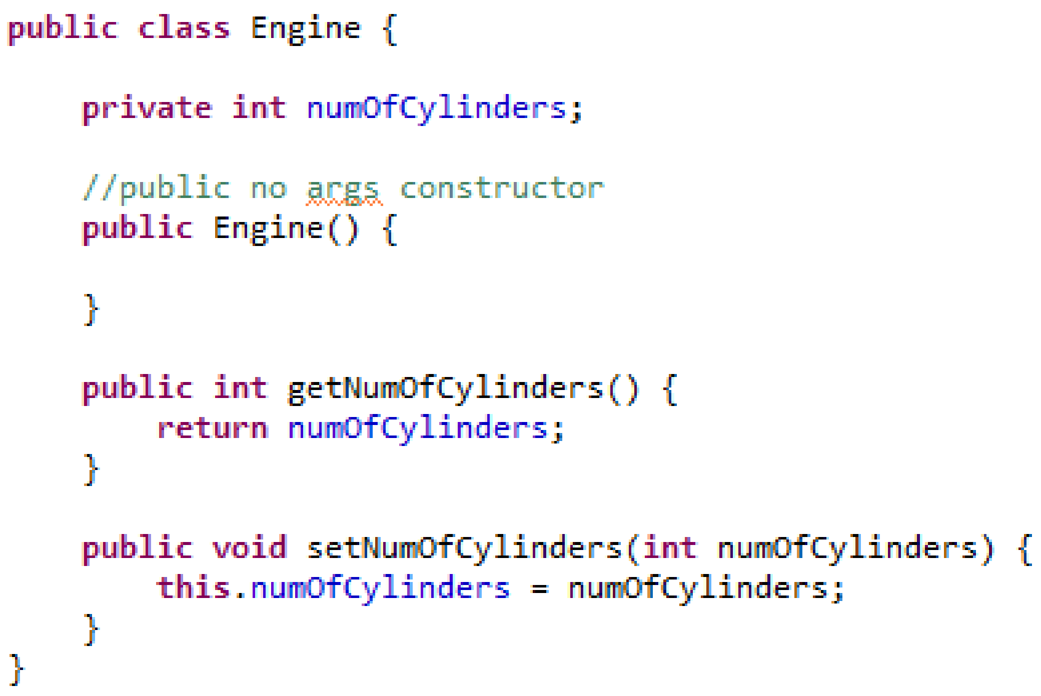
```java
@Component
public class Engine {

    private int numOfCylinders;

    //public no args constructor
    public Engine() {

    }

    public int getNumOfCylinders() {
        return numOfCylinders;
    }

    public void setNumOfCylinders(int numOfCylinders) {
        this.numOfCylinders = numOfCylinders;
    }
}
```

Constructor Injection

- Constructor with args in order that you want



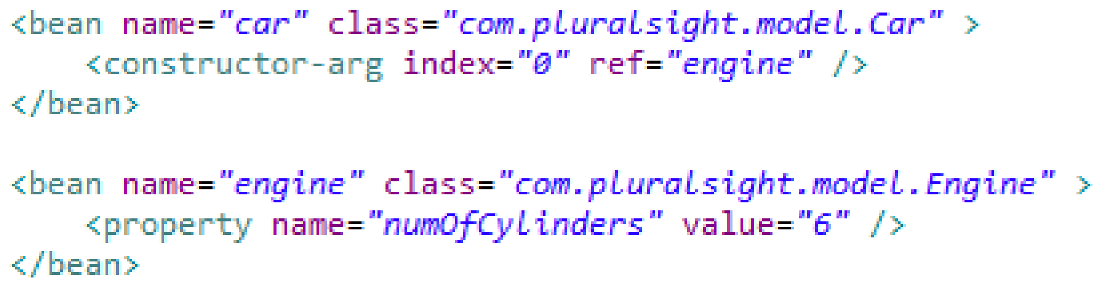
```java
public class Car {

    private Engine engine;

    //constructor injection
    public Car(Engine engine) {
        this.engine = engine;
    }

    public Engine getEngine() {
        return engine;
    }

    public void setEngine(Engine engine) {
        this.engine = engine;
    }

}
```



```java
public class Engine {

    private int numOfCylinders;

    //public no args constructor
    public Engine() {

    }

    public int getNumOfCylinders() {
        return numOfCylinders;
    }

    public void setNumOfCylinders(int numOfCylinders) {
        this.numOfCylinders = numOfCylinders;
    }
}
```

Spring XML Config Constructor Injection

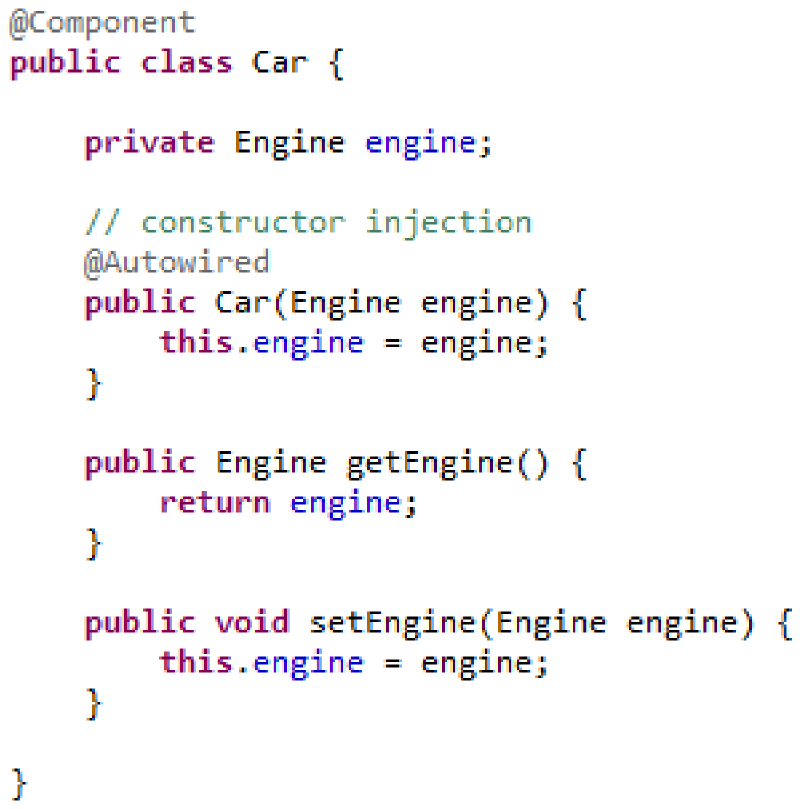
- The same code configured using XML and Spring:



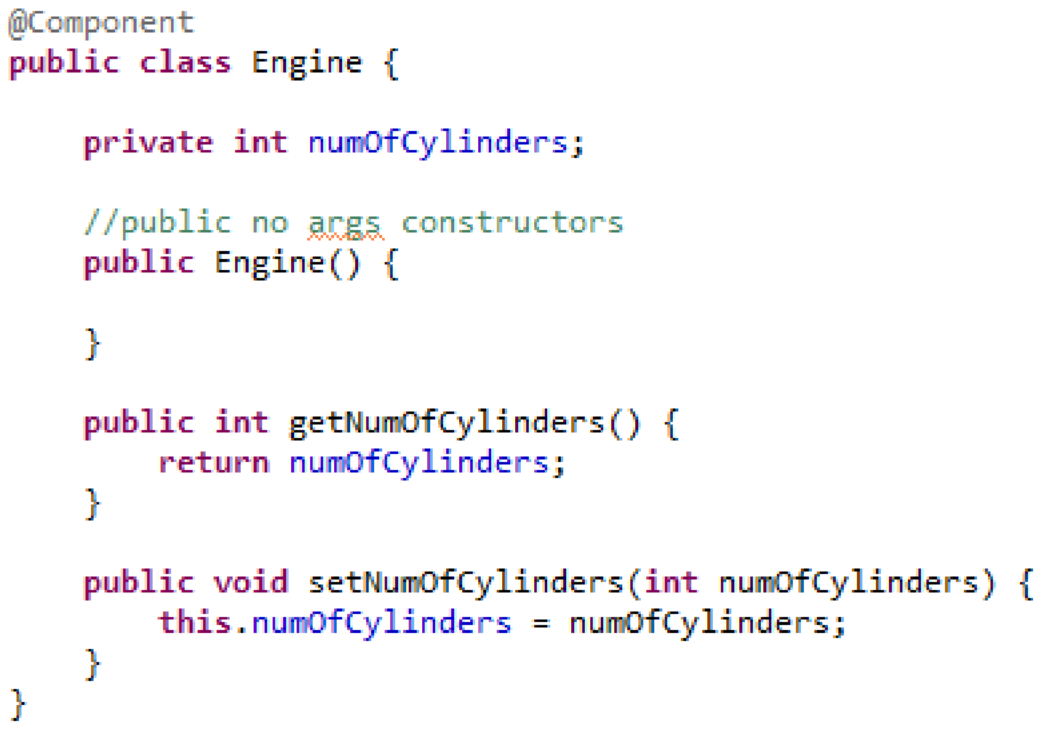
```xml
<bean name="car" class="com.pluralsight.model.Car" >
    <constructor-arg index="0" ref="engine" />
</bean>

<bean name="engine" class="com.pluralsight.model.Engine" >
    <property name="numOfCylinders" value="6" />
</bean>
```

Autowiring Constructors

- Instead of using XML, we can also Autowire using Annotations:



```java
@Component
public class Car {

    private Engine engine;

    // constructor injection
    @Autowired
    public Car(Engine engine) {
        this.engine = engine;
    }

    public Engine getEngine() {
        return engine;
    }

    public void setEngine(Engine engine) {
        this.engine = engine;
    }

}
```



```java
@Component
public class Engine {

    private int numOfCylinders;

    //public no args constructors
    public Engine() {

    }

    public int getNumOfCylinders() {
        return numOfCylinders;
    }

    public void setNumOfCylinders(int numOfCylinders) {
        this.numOfCylinders = numOfCylinders;
    }
}
```

Context Files

- **Spring configuration can be done multiple ways:**
  - XML
  - Annotations
  - Java Config
- **Using annotations there is still some XML config**
- **Context files are loaded multiple ways:**
  - Dispatcher Servlet
  - Loader Listener
  - Import

Summary

- Setter Injection
- Constructor Injection
- POJOs
- XML Configuration
- Autowiring
- Context Files