Spring/Spring Boot

Significant Terminology

Java Enterprise Edition

- Java Standard Edition extension
- Directed to enterprise solutions
- Java EE applications are run on application servers (Tomcat), which handle transactions, security, scalability, concurrency and management of the components it is deploying.

Boilerplate code

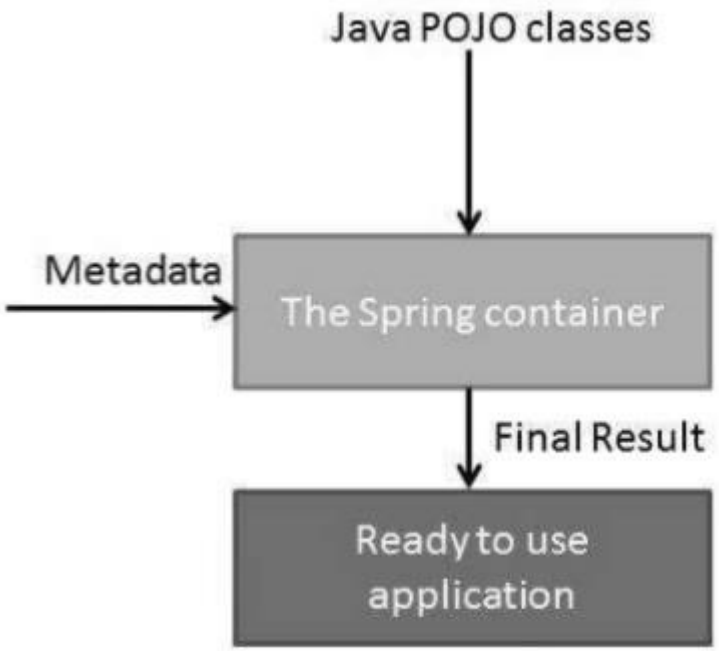
- Boilerplate refers to sections of code that have to be included in many places with little or no alteration. It is often used when referring to languages that are considered verbose, i.e. the programmer must write a lot of code to do minimal jobs

Dependecy Injection

- You do not have to create your objects, you have to describe how they should be created
- Don’t connect components and services directly, describe which services are needed by which component in config file
- How DI could be done:
  - By constructor
  - By seter
  - By Interface (not in spring)

Spring IoC Container

- Core of Spring Framework
- Creates, wires objects
- Configure and manage objects lifecycle



Spring Beans

- Objects that are backbone of application
- Managed by Spring IoC
- Configure via config metadata

IoC Container Types

- **Bean factory** – class that contains collection of beans, initializes the bean when client ask for

- **Application context** – is the central interface within a Spring application for providing configuration information to the application, provides extra functionality

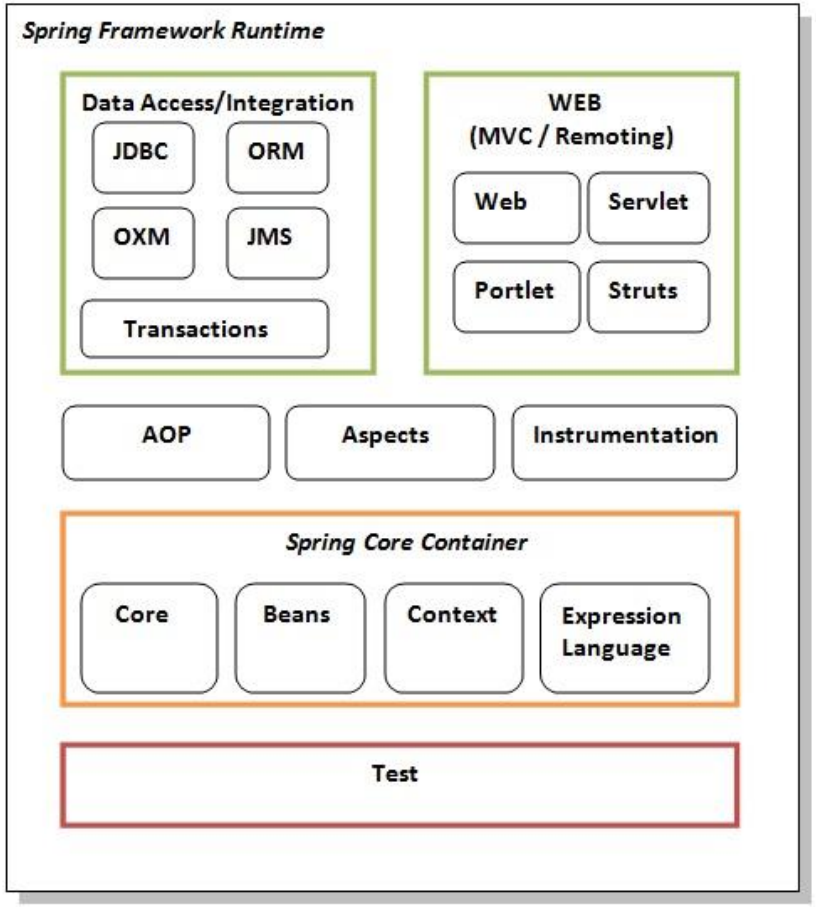
Spring Bean scopes

- Singleton
- Prototype
- Request
- Session
- Global session

ApplicationContext

- AnnotationConfigApplicationContext
- AnnotattionConfigWebApplicationContext
- ClassPathXmlApplicationContext
- FileSystemXmlApplicationContext
- XmlWebApplicationContext

Spring Modules



Spring’s configuration options

- Explicit configuration in XML
- Explicit configuration in Java
- Implicit bean discovery and automatic wiring

Automatically wiring beans

- *Component scanning* - Spring automatically discovers beans to be created in the application context.
- *Autowiring* - Spring automatically satisfies bean dependencies.

Component Scanning

```
@Configuration
@ComponentScan("soundsystem")
public class CDPlayerConfig {}
```

```
@Configuration
@ComponentScan(basePackages={"soundsystem", "video"})
public class CDPlayerConfig {}
```

```
@Configuration
@ComponentScan(basePackageClasses={CDPlayer.class, DVDPlayer.class})
public class CDPlayerConfig {}
```

```
@Component
public class CDPlayer implements MediaPlayer {
  private CompactDisc cd;
```

Autowiring

```
package soundsystem;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class CDPlayer implements MediaPlayer {
  private CompactDisc cd;

  @Autowired
  public CDPlayer(CompactDisc cd) {
    this.cd = cd;
  }

  public void play() {
    cd.play();
  }

}
```

@Autowired and @Qualifier

- *@Autowired* – use to wire bean at setter or constructor, provides more detailed control over how autowiring should be accomplished
- When more than bean of the same type is used *@Qualifier* is used along with *@Autowired* to remove confusion by specifying which ban will be wired.

@Component

- @Controller – presentation layer
- @Service – service (logic) layer
- @Repository – persistence layer