SPRING CORE CHEAT SHEET

Container, Dependency and IOC

Spring **5.x** - Date : **August 2018**

WHAT IS SPRING ?

- **Open source** (Apache 2 licence, sources on github)
- **Light**
  - Doesn't force to use an application server
  - Not invasive
- **Container**
  - Application objects doesn't have to look for their dependencies
  - Handles objects life cycle
- **Framework**
  - Ease integration and communication with third-party libraries

MINIMAL DEPENDENCIES

```
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-context</artifactId>
</dependency>
```

APPLICATION CONFIGURATION

- Java
- Annotations
- XML (still available but heavy)

```
@ComponentScan("fr.sii.cheatsheet.spring")
public class MyApp {

   public static void main(String[] args) {
      ApplicationContext app = new
AnnotationConfigApplicationContext(MyApp.class);

      DummyService helloWorld =
app.getBean(DummyService.class);
      helloWorld.getMessage();
   }
}
```

Prefer use of a configuration class

```
@Configuration
public class MyAppConfig {
  // @Bean, ...
}
```

DEPENDENCY INJECTION

Define a bean : @Bean

```
@Configuration
public class MyAppConfig {
  @Bean
  public DummyService dummyService(){
    return new DummyServiceImpl();
  }
}
```

Define a bean : @Component

@Component specializations :

- **@Service** : Heart of an app
- **@Repository** : Handles data persistence
- **@Controller** : Handles requests and reponses

```
@Component
public class DummyServiceImpl implements
DummyService {
}
```

Dependency injection : @Autowired

- Via a class field
- Via a setter
- Via the constructor *(prefer for easy test)*

```
@Component
public class FooServiceImpl implements FooService
{

  @Autowired
  private DummyService service;

  @Autowired
  public FooServiceImpl(DummyService
dummyservice) {
    this.service = dummyService;
  }

  @Autowired
  public DummyService
setDummyService(DummyService dummyService) {
    this.service = dummyService;
  }

}
```

SPRING CORE CHEAT SHEET

BEANS SCOPES

Scope	Description
singleton	(default) A single bean instance
prototype	A bean instance per usage
request	A bean instance per **HTTP request**
session	A bean instance per **HTTP session**
application	A bean instance per Servlet Context
websocket	A bean instance per WebSocket

CUSTOMIZING A BEAN

Initialization method

```
@PostConstruct
```

Destroy method

```
@PreDestroy
```

SPEL : SPRING EXPRESSION LANGUAGE

Access a system properties

```
@Value("#{ systemProperties['user.home'] }")
private String defaultHome;
```

Access a bean property

```
@Value("#{myBean.myValue}")
private String myValue;
```

Parse a string

```
@Value("#{myBean.myValue.substring(0,1)}")
private String myValue;
```

Operations

```
@Value("#{myBean.myValue.length() > 2}")
private String myValue;
```

PROPERTIES

```
/**
 * File 'foo.properties' loaded by Spring
 */
@Configuration
@PropertySource("classpath:foo.properties")
public class MyAppConfig {

  /**
   * Property max in file foo.properties.
   */
  private Integer max;

}
```

Use a property : **@Value("${message}")**

```
@Value("${message:Default message}")
private String message;
```

GO DEEPER

- https://spring.io/guides
- https://spring.io/projects/spring-framework
- Spring Core documentation
- Spring Boot cheat sheet