Spring 5

https://learning.oreilly.com/library/view/mastering-spring-5/9781789615692/09f950b4-7d56-453b-bc91-6cdf96c4a3b4.xhtml

https://github.com/PacktPublishing/Mastering-Spring-5.1

A alternative framework to Java EE, which consists of more that 20 modules. Its baseline version of java is java 8.

- Spring Boot
- Spring MVC
- Spring Beans
- Spring JDBC
- Spring AOP
- Spring Web services
- Spring Batch
- Spring Cloud
- Spring Web Flux
- Spring Transactions
- Spring Data
- Spring Security
- Spring HATEOAS (Hypermedia As The Engine of Application State)

Unchecked Exceptions

Most modules in Spring convert exceptions into unchecked exceptions. This means you don't have to handle exceptions reducing the plumbing code which allow you concentrate on writing business logic.

Spring allows you to centralise the management of handling exceptions

Reactive Programming

Spring Web Services

Contract first SOAP web services

Spring MVC

Provides support for REST web services.

Microsservice Architecture

In the last few years, organisations have been moving towards this pattern away from monolithic applications.

Services are independently deployable.

Spring Boot

Build applications fast with standard features for microservices and web apps.

```
* external config
* health checks and monitoring
```

Spring Data

```
integration with:

    * JPA
    * MongoDB
    * Redis
    * Solr
    * Gemfire
    * Apache Cassandra
    * Neo4J for Graph Server
```

Spring Cloud

```
* configuration management
* service discovery
* circuit breakers
    * uses the "Netflix Hystrix" fault tolerance library
* api gateways
    * integrates withthe API gateway library, Netflix Zuul
```

Spring Security

```
* Authentication and Authorisation

* intergrates with Spring MVC and Servlets APIs

* can integrate with SAML and LDAP
```

Reactive Programming

Reactive programming support. microservices respond to events.

Function Web Framework

```
RouterFunction <String> route = route(
    GET("/hello-world"),
    request -> Response.ok().body(fromObject("Hello World"))
);
```

Jigsaw

Allow classes to put into modules.

Java 8 has modules which may make this redundant!

Dependency Injection

classes with injectable dependencies are known as "beans"

dependency injection is achieved through construction injection

decorate "beans" with annotations:

```
@Component
@Service
@Repository
```

if is possible to use @component for all beans but service and repository annotation has extra features. repository has some automatic transaction management.

add @Autowired annotation to instance variables of dependencies

Create an IoC container:

use ApplicationContext or BeanFactory. ApplicationContext is a superset of BeanFactory it is recommended to use this rather than BeanFactory unless memory use is a concern.

Application Context Config

Place in class file decorated with @Configuration annotation.

```
@Configuration
class SpringContext {

}
```

Tell spring where to scan for beans

```
@Configuration
@ComponentScan(basePackages = { "com.mastering.spring" })
class SpringContext {

}
```

Create Application Context Instance and resolve class

```
ApplicationContext context = new
AnnotationConfigApplicationContext(SpringContext.class);BusinessService
service = context.getBean(BusinessService.class);
```

Bean scope.

All beans created default to the singleton scope. Only one instance is created and is reused through over dependencies. If the bean contains state information. use the Prototype scope

Decorate the bean with:

```
@Scope("prototype")
```

Mock objects

Mokito

Actuator

Monitor and manage your application

Integration

- Spring Rabbit
- JMS (ActiveMQ)
- JMS (Artemis)

Jackson

A JSON serialising +library

MockMVC

Built into Spring, mocks HTTP Calls

TestRestTemplate Class

This actually tests the application using http. Use the @SpringBootTest annotation on the test class

Register a method to be used by spring to instantiate a type

Add @Bean to the public method.

```
@Bean
public MyType getAType(){
    return new MyType();
}
```

Then in the consuming component

```
class MyComponent {
    @Autowired
    private MyType mytype;
}
```

You can also inject in ApplicationContext into the component instead and the use the instance of that to instantiate your type.

```
class MyComponent {
    @Autowired
    private ApplicationContext ctx

    private void myPrivateMethod(){
        MyType mytype = ctx.getBean(MyType.class);
    }
}
```

If multiple beans are registerd with the same type. Use the `@Qualifier` annotation next to the `@Autowired` annotation.

`@Autowired` matches beans by type. If you want to match beans by name, you can use `@Resource` instead.

RestTemplate

This makes http calls to remote services and serialises the result to pojos.

If json variable name is different from pojo property name, use @JsonProperty annotation on the property name.

Or use the application property:

```
spring.jackson.property-namign-strategy=SNAKE_CASE
```

RestTemplate is Synchronous only.

Use org.springframeword.web.reactive.client.WebClient if this is not good enough.

Logging

- slf4j

```
public class MyClass {
    private Logger logger = LoggerFactory.getLogger(this.getClass());

    private MyClass(){
        logger.info("Class instantiated");
    }

}
```

JPA

Decorate pojos wity @Entity annotation

```
@Entity
public class MyType {
    private String name;
    private int age;

    // getters and setters hidden
}
```

Then setup repostory with this:

```


interface MyRepository extends JpaRepository<MyType, Long> {

}
```

You can implement your own repo. You inject in an EntityManager

```


@Repository
public MyRepository {
    @PersistenceContext
    private EntityManager entityManager;

    public MyType save(MyType obj) {
        entityManager.persist(obj);
        return obj;
    }
}
```

Atomic Long

threadsafe long which can be incremented safely.

Transactions

Decorate test class with `@Transactional`

In Tests: all database calls in a test method will be done in a transaction and rolled back after each test has run.

In Normal Code: all database calls are done in transactions and only rolled back on exception.

Usually you put the annotation on a service layer.

Change SQL logging level

in application.properties

```
logging.level.sql=debug
```

Spring Data Rest

You can turn your repository into a REST service just by adding the following dependency

```
org.springframework.boot:spring-boot-starter-data-rest
```

HAL Browser for Spring Data Rest

Web front end to allow you to call test your REST api.

Add the following dependency:

```
org.springframework.data:spring-data-rest-hal-browser
```