Tópicos em Frameworks

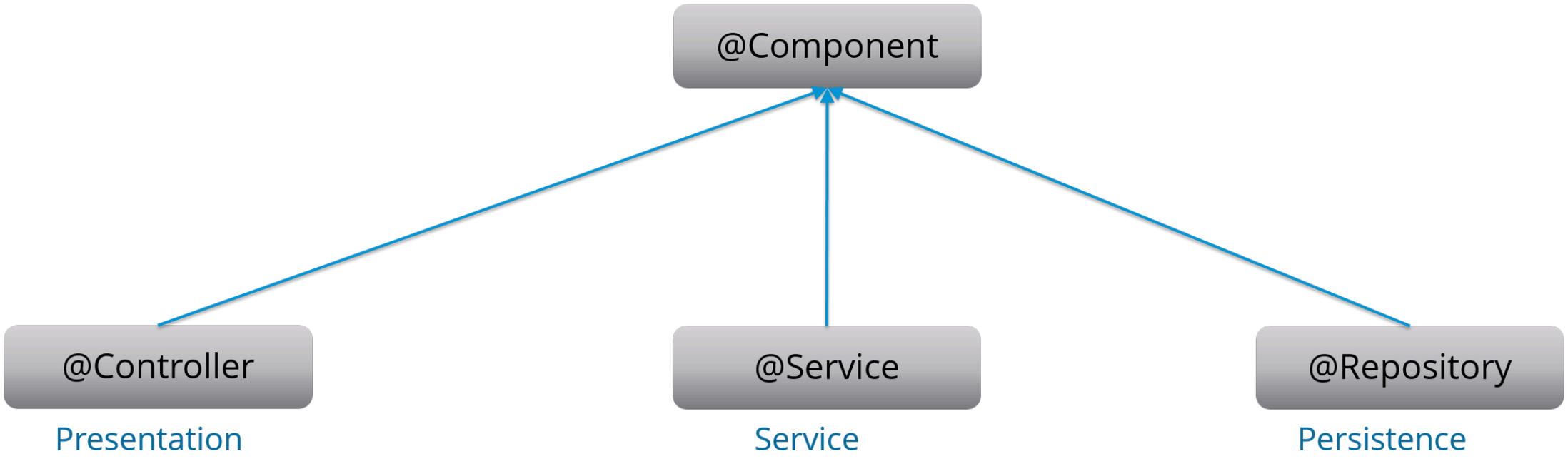
Cássio Seffrin

Spring MVC

DI - injeção de dependência

Na Injeção de Dependência, você não precisa criar seus objetos, mas descrever como eles devem ser criados. Você não conecta seus componentes e serviços diretamente no código, mas descreve quais serviços são necessários e quais componentes no arquivo de configuração. O contêiner de IoC os conectará.

Spring MVC - Anotações



Spring MVC - Anotações

Nas aplicações mais comuns, temos camadas distintas, como acesso a dados, apresentação, serviço/negócios e controladores.

E, em cada camada, temos vários Java beans. Para detectá-los automaticamente, o Spring usa anotações de verificação de caminho de classe. Em seguida, registra cada bean no ApplicationContext.

As principais anotações do Spring são:

@Component é um estereótipo genérico para qualquer componente gerenciado pelo Spring;

@Service são classes na camada de serviço (lógica de negócio);
@Repository anota classes na camada de persistência, que atuará como um repositório de banco de dados;
@Controller é usado para classes de controle do MVC.

Estas anotações são usados para diferentes classificações. Quando anotamos uma classe para detecção automática, devemos usar o respectivo estereótipo.

Spring MVC - Anotações

É importante escolhermos a anotação com base em suas convenções de camada:

@Component
Podemos usar o @Component para marcar os beans como componentes gerenciados do Spring. Em geral o Spring apenas registra beans com @Component e não procura @Service, @Repository e @Controller.

Eles são registrados no ApplicationContext quando são anotados com @Component

@Service, @Repository e @Controller são especializações de @Component. Eles são tecnicamente iguais, mas os usamos para diferentes propósitos.

O trabalho do @Repository (especialização do Component) é capturar exceções específicas de persistência e repeti-las novamente como uma das exceções não verificadas e unificadas do Spring e fornece PersistenceExceptionTranslationPostProcessor. Além de importar os DAOs para o contêiner através de DI (Dependency Injection.

Spring MVC - Anotações

Anotamos um bean com **@Service** (especialização do Component) para indicar que ele está mantendo a lógica de negócios. Não fornece nenhum comportamento adicional sobre a anotação @Component, mas é uma boa ideia usar @Service invés @Component, pois especifica melhor a intenção da classe. Além disso, o suporte a ferramentas e o comportamento adicional podem contar com isso no futuro.

A anotação **@Controller** (especialização do Component) marca uma classe como um controlador Spring Web MVC. Também é uma especialização @Component, portanto, os beans marcados com ele são importados automaticamente para o contêiner DI. Quando adicionamos a anotação @Controller a uma classe, podemos usar outras anotações derivadas, ou seja, @GetMapping, @PostMapping etc.

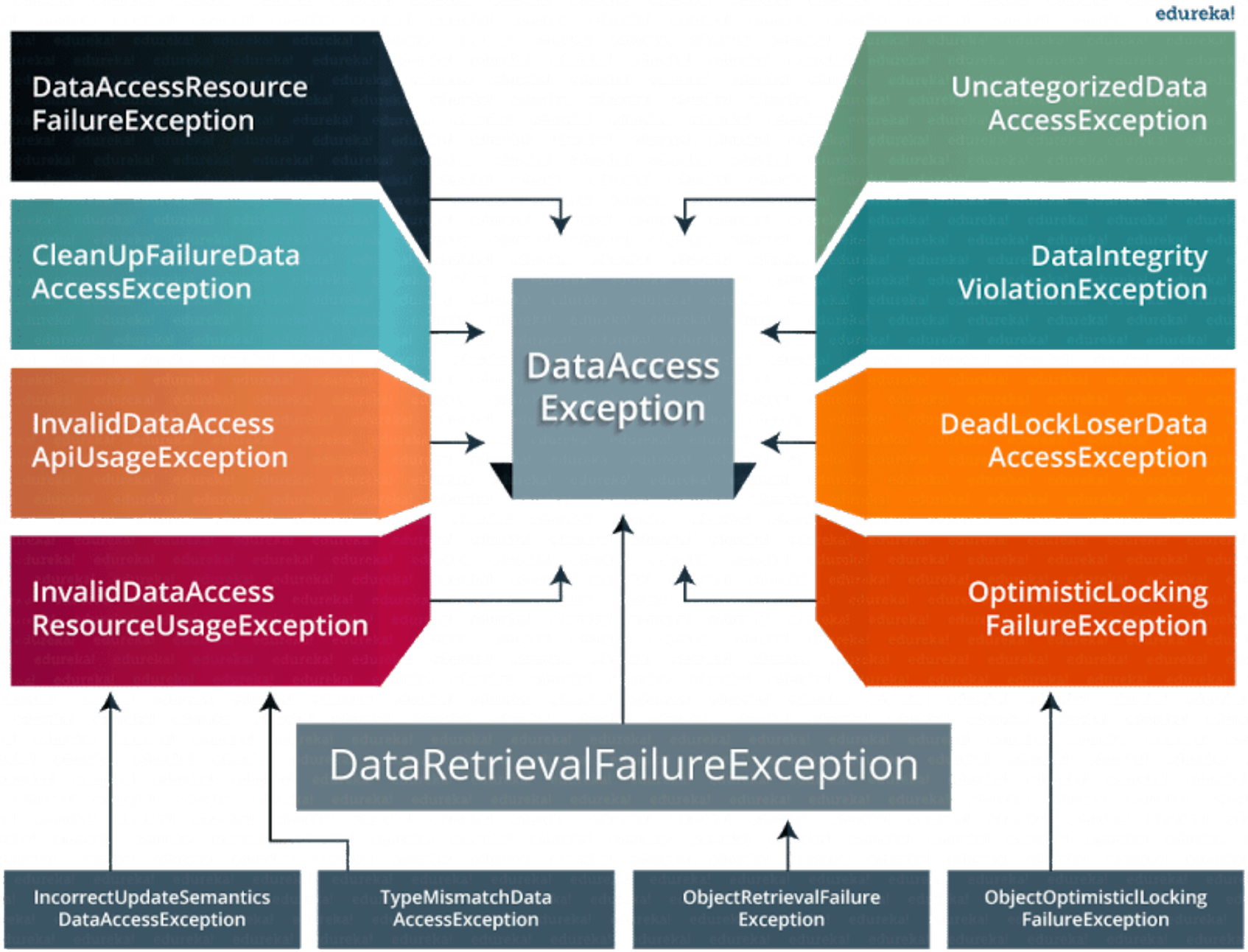
Spring MVC - Anotações

@Configuration
A anotação Spring @Configuration auxilia na configuração baseada em anotações Spring, indicando que uma classe declara um ou mais métodos @Bean e podem ser processados pelo contêiner Spring para gerar definições de bean e solicitações de serviço para esses beans em tempo de execução.
Ex:

```
@Configuration
@ComponentScan(basePackages = “com.cassio.edu”)
class MatriculaConfig {
    @Bean
    Matricula matricula() {
        retorn new Matricula ();
    }
}
```

A anotação **@Bean** que possui comportamentos destinos das outras:
O @Bean é usado para declarar explicitamente um único bean, em vez de permitir que o Spring faça isso automaticamente para nós.
Outra grande diferença é que o @Bean é uma anotação no nível do método e não da classe e, por padrão, o nome do método serve como o nome do bean.

Spring Data - Excessões



Spring MVC

```java
@Slf4j
public abstract class RESTController<T, ID extends Serializable> {

    private static final String SUCCESSO = "successo";
    private CrudRepository<T, ID> repo;

    public RESTController(CrudRepository<T, ID> repo) {
        this.repo = repo;
    }

    @GetMapping
    public @ResponseBody Collection<T> todos() {
        Iterable<T> todos = this.repo.findAll();
        return Uteis.converterInterabelParaList(todos);
    }


    @PostMapping(consumes={MediaType.APPLICATION_JSON_VALUE})
    public @ResponseBody Map<String, Object> salvar(@RequestBody T json) {
        log.info("criado() com body {} do tipo {}", json, json.getClass());
        T objetoCriado = this.repo.save(json);
        Map<String, Object> m =  new HashMap<>();
        m.put(SUCCESSO, true);
        m.put("criado", objetoCriado);
        return m;
    }
}
```

Spring MVC

```
@Controller
@RequestMapping("/curso")
public class CursoController extends RESTController<Curso,
Integer> {

    @Autowired
    public CursoController(CrudRepository<Curso, Integer> repo) {
        super(repo);
    }
}
```