Web-based programming with Java

HIBERNATE INTEGRATION

ThS. Nguyễn Nghiệm
0913.745.789 - NghiemN@fpt.edu.vn

OBJECTIVES

- Tích hợp Hibernate vào Spring
- Lập trình Hibernate với Spring Transaction
- Xây dựng Crud DAO

AGENDA

- Cấu hình thư viện cần thiết
- Cấu hình tích hợp Hibernate
- Lập trình Hibernate với Spring Transaction
- Xây dựng DAO

INTRODUCTION

- Tích hợp Hibernate vào Spring ứng dụng web sẽ chạy với hiệu suất cao hơn vì 2 lý do sau đây
  - Spring Bean:
    - SessionFactory, Session được Spring nạp vào và quản lý
  - Spring Transaction
    - Spring hỗ trợ quản lý Transaction tự động

TÍCH HỢP HIBERNATE VÀO SPRING

THƯ VIÊN PHỤ THUỘC

- Hibernate/JPA

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

- Database Driver

```
<dependency>
    <groupId>com.microsoft.sqlserver</groupId>
    <artifactId>mssql-jdbc</artifactId>
    <scope>runtime</scope>
</dependency>
```

CẤU HÌNH HIBERNATE

```
@SpringBootApplication
@EnableAutoConfiguration(exclude = {
        DataSourceAutoConfiguration.class,
        DataSourceTransactionManagerAutoConfiguration.class,
        HibernateJpaAutoConfiguration.class
    })
public class MainApplication {

    public static void main(String[] args) {
        SpringApplication.run(MainApplication.class, args);
    }
}
```

- Hủy bỏ cấu hình DataSource mặc định của Spring Boot

HIBERNATECONFIG.JAVA

```
@Configuration
@PropertySource("classpath:datasource.properties")
public class HibernateConfig {
	@Autowired
	Environment env;
	@Bean
	public DataSource getDataSource() {
		DriverManagerDataSource dataSource = new DriverManagerDataSource();
		dataSource.setDriverClassName(env.getProperty("db.driver"));
		dataSource.setUrl(env.getProperty("db.url"));
		dataSource.setUsername(env.getProperty("db.username"));
		dataSource.setPassword(env.getProperty("db.password"));
		return dataSource;
	}
	@Bean @Autowired
	public SessionFactory getSessionFactory(DataSource dataSource) throws Exception {
		LocalSessionFactoryBean factoryBean = new LocalSessionFactoryBean();
		factoryBean.setPackagesToScan(new String[] { "com.eshop.entity" });
		factoryBean.setDataSource(dataSource);
		Properties props = factoryBean.getHibernateProperties();
		props.put("hibernate.dialect", env.getProperty("hb.dialect"));
		props.put("hibernate.show_sql", env.getProperty("hb.show-sql"));
		props.put("hibernate.ddl-auto", env.getProperty("hb.ddl-auto"));
		props.put("current_session_context_class", env.getProperty("hb.session"));
		factoryBean.afterPropertiesSet();
		return factoryBean.getObject();
	}
	@Bean @Autowired
	public HibernateTransactionManager getTransactionManager(SessionFactory sessionFactory) {
		return new HibernateTransactionManager(sessionFactory);
	}
}
```

DATASOURCE.PROPERTIES

```
db.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
db.url=jdbc:sqlserver://localhost;database=EShopV10
db.username=sa
db.password=songlong
hb.show-sql=true
hb.ddl-auto=none
hb.dialect=org.hibernate.dialect.SQLServer2012Dialect
hb.session=org.springframework.orm.hibernate5.SpringSessionContext
```

- File datasource.properties đặt trong classpath nó chứa các thông số sau
  - Thông số kết nối CSDL
  - Thông số hibernate

LẬP TRÌNH HIBERNATE

```
@Transactional
@Controller
public class HelloController {
	@Autowired
	SessionFactory factory;

	@RequestMapping("read")
	public String readAll() {
		Session session = factory.getCurrentSession();

		String hql = "FROM Product";
		TypedQuery<Product> query = session.createQuery(hql, Product.class);
		List<Product> list = query.getResultList();

		return "index";
	}

	@RequestMapping("create")
	public String create(Category category) {
		Session session = factory.getCurrentSession();
		session.persist(category);
		return "index";
	}
}
```

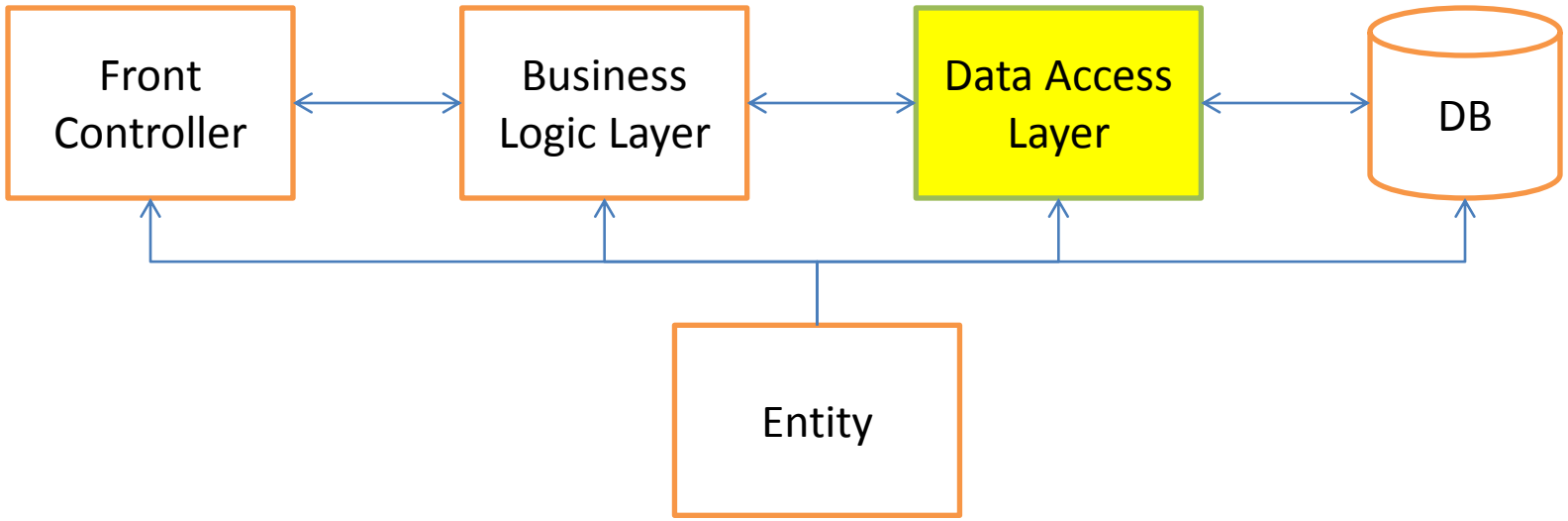
Để Spring điều khiển Transaction

Tiêm SessionFactory

Lấy Session hiện tại

XÂY DỰNG DAO

Professional Application Organization

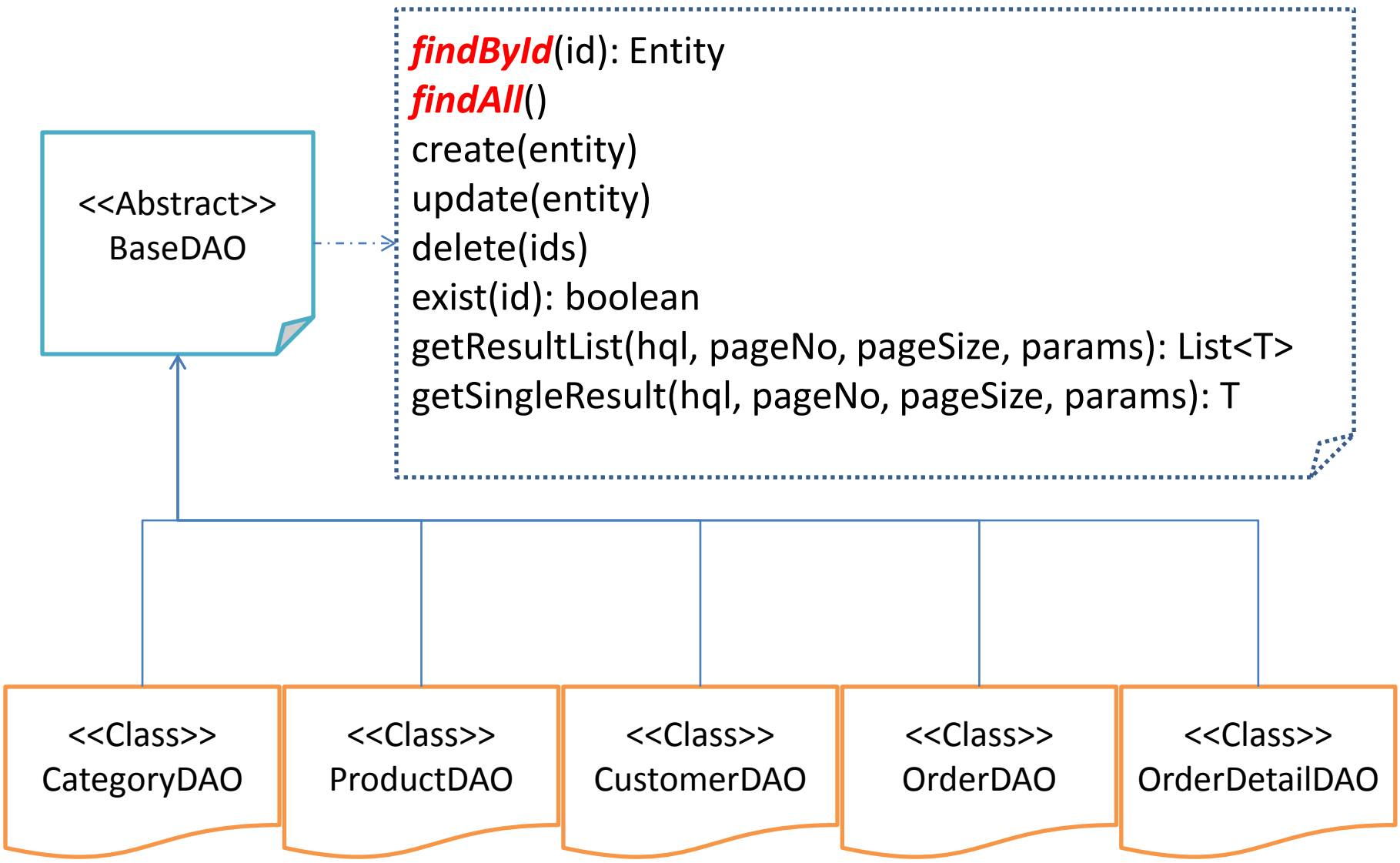


- Data Access Layer gồm các DAO (Data Access Object)
- DAO là các components làm việc với CSDL

BUILDING DAO

- Các hoạt động thông thường
  - Truy vấn tất cả
  - Truy vấn một thực thể theo id
  - Truy vấn theo hql có phân trang và điều kiện
  - Tạo mới
  - Cập nhật
  - Xóa
- Tùy vào đối tượng và yêu cầu, mỗi DAO có thể bổ sung thêm các truy vấn và thao tác khác

DAO Components



BaseDAO

```
@Transactional
abstract public class BaseDAO<E, K> {
        @Autowired
        protected SessionFactory factory;

        abstract public E findById(K id);
        abstract public List<E> findAll();

        public void create(E entity) {...}
        public void update(E entity) {...}
        public void delete(K...ids) {...}
        public boolean exist(K id) {...}
        public <T> List<T> getResultList(Class<T> clazz,
                        int pageNo, int pageSize, String hql, Object...params){...}
        public <T> T getSingleResult(Class<T> clazz, String hql, Object...params) {...}
}
```

CATEGORYDAO

```
@Repository
public class CategoryDAO extends BaseDAO<Category, Integer>{
	@Override
	public Category findById(Integer id) {
		Session session = factory.getCurrentSession();
		return session.find(Category.class, id);
	}

	@Override
	public List<Category> findAll() {
		String hql = "FROM Category";
		return this.getResultList(Category.class, hql, 0, 0);
	}
}
```

USING DAO

```
@Controller
public class CategoryController {
	@Autowired
	CategoryDAO dao;

	@PostMapping("category/create")
	public String create(Category category) {
		dao.create(category);
		return "category/index";
	}

	@PostMapping("category/edit/{id}")
	public String update(Model model, @PathVariable("id") Integer id) {
		Category category = dao.findById(id);
		model.addAttribute("item", category);
		return "category/index";
	}
}
```

SUMMARY

- Cấu hình thư viện cần thiết
- Cấu hình tích hợp Hibernate
- Lập trình Hibernate với Spring Transaction
- Xây dựng DAO