LẬP TRÌNH JAVA WEB SPRING MVC

Nguyễn Thái Sơn

SPRING JDBC

SPRING JDBC TEMPLATE

Dependency

- Sử dụng thư viện hộ trợ JDBC Spring MVC
- Sử dụng Driver cho hệ quản trị CSDL

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>5.2.7.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.19</version>
</dependency>
```

Tạo Bean

- DataSource: để sử dụng trong JDBC
- JDBC Template: Bean sử dụng trong spring mvc

```
@Bean
public DataSource dataSource() {
    DriverManagerDataSource dataSource = new DriverManagerDataSource();
    dataSource.setDriverClassName("com.mysql.jdbc.Driver");
    dataSource.setUrl("jdbc:mysql://localhost:3306/BAN_HANG");
    dataSource.setUsername("root");
    dataSource.setPassword("");
    return dataSource;
}
@Bean
public JdbcTemplate jdbcTemplate() {
        return new JdbcTemplate(dataSource());
}
```

PROPERTES FILES

SPRING PROPERTIES FILE

Spring Properties Files

- Tải các file properties vào trong chương trình của spring MVC
- Sử dụng dữ liệu trong file thông qua key/value
- classpath prefix: trỏ đến src/main/resource
- File: prefix là trỏ đến một thư mục trong máy tính
- Mặc định là webapp

Cấu hình

```
@PropertySource(value = {"classpath:db.properties"})
public class SpringConfiguration implements WebMvcConfigurer{
@Autowired
Environment enviroment;
@Bean
public static PropertySourcesPlaceholderConfigurer
placeholderConfigurer() {
        return new PropertySourcesPlaceholderConfigurer();
}
DriverManagerDataSource dataSource = new
DriverManagerDataSource();
dataSource.setDriverClassName(enviroment.getProperty("driver"));
dataSource.setUrl(enviroment.getProperty("url"));
dataSource.setUsername(enviroment.getProperty("username"));
dataSource.setPassword(enviroment.getProperty("password"));
```

JDBC TRANSACTION

SPRING JDBC TRANSACTION

Dependency

- Transaction quản lý giao dịch, đảm bảo thành công hoặc rollback

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-tx -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
            <version>5.2.6.RELEASE</version>
        </dependency>
```

Cấu hình

- @Transactional: cho class hoặc phương thức áp dụng transaction

```
@EnableTransactionManagement
@PropertySource(value = {"classpath:db.properties"})
public class SpringConfiguration implements WebMvcConfigurer{

@Bean
public DataSourceTransactionManager dataSourceTransactionManager() {
        return new DataSourceTransactionManager(dataSource());
}
```

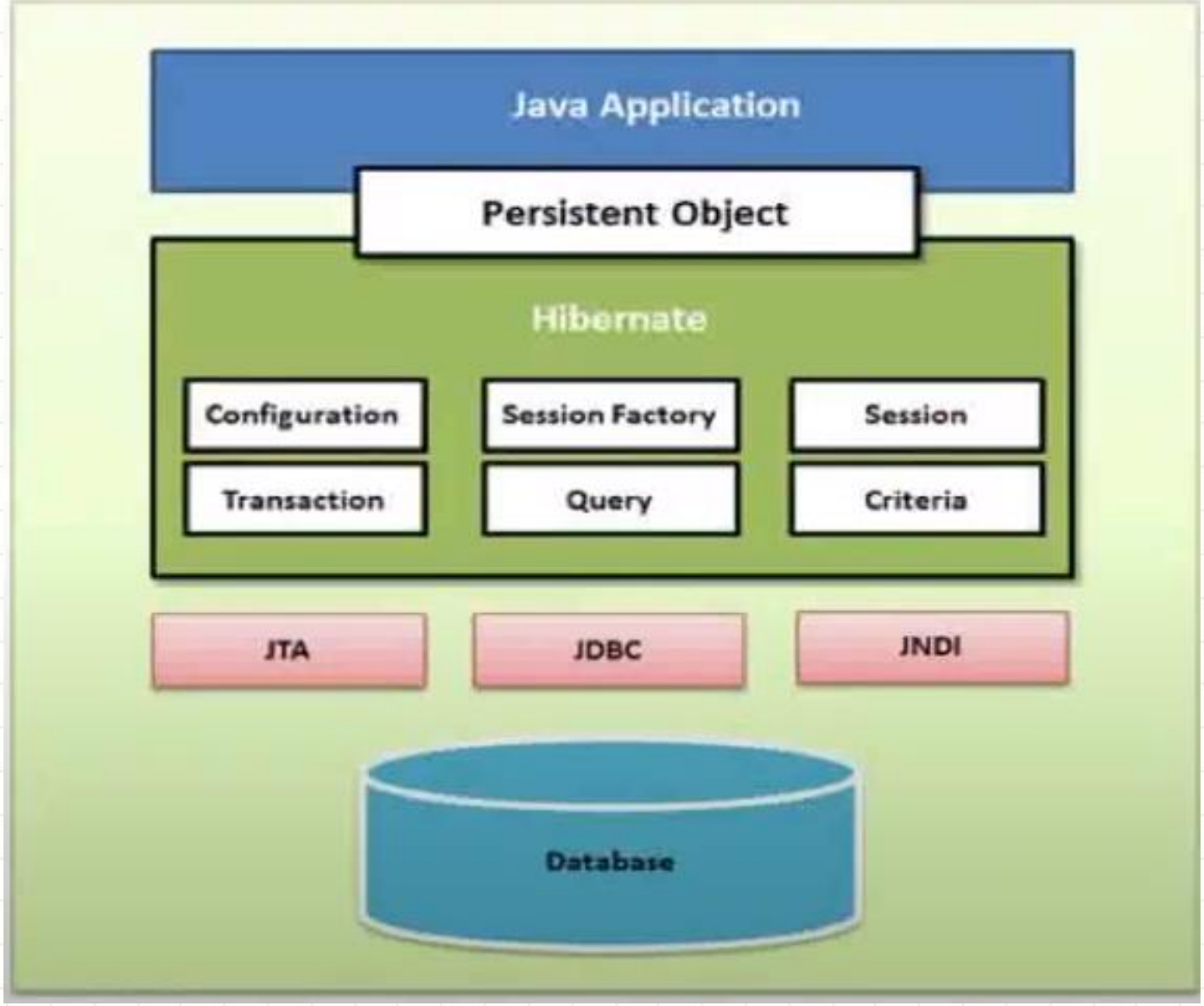
SPRING HIBERNATE

SỬ DỤNG HIBERNATE TRONG SPRING

Hibernate

- Hibernate là ORM Framework. Ánh xạ các thuộc tính của đối tượng với các cột trong bảng của CSDL
- Làm việc với đối tượng Java, không quan tâm đến CSDL
- Free open-source và dễ dàng áp dụng cho các hệ quản trị CSDL quan hệ khác nhau
- Sử dụng API nền tảng như JDBC, JTA, JNDI
- Hỗ trợ đầy đủ các hàm truy vấn từ dạng object đến native sql

Hibernate


Java Application
Persistent Object
Hibernate
Configuration
Session Factory
Session
Transaction
Query
Criteria
JTA
JDBC
JNDI
Database

Dependency

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-orm -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-orm</artifactId>
    <version>5.2.6.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.4.18.Final</version>
</dependency>
```

Cấu hình

```
@Bean
public LocalSessionFactoryBean localSessionFactoryBean() {
    LocalSessionFactoryBean bean = new LocalSessionFactoryBean();
    bean.setDataSource(dataSource());
    bean.setPackagesToScan("com.ezcloud.entity");
    Properties properties = new Properties();
    properties.put("hibernate.dialect", enviroment.getProperty("hibernate.dialect"));
    properties.put("hibernate.show_sql",
    enviroment.getProperty("hibernate.show_sql"));
    return bean;
}
@Bean(name = "transactionManager")
@Autowired
public HibernateTransactionManager
hibernateTransactionManager(SessionFactory sessionFactory) {
    HibernateTransactionManager hibernateTransactionManager = new
    HibernateTransactionManager();
    hibernateTransactionManager.setSessionFactory(sessionFactory);
    return hibernateTransactionManager;
}
```

EXCEPTION HANDLER

BẮT LỖI NGOẠI LỆ TRONG SPRING

Exception Handler

- Tạo controller class và sử dụng annotation @ControllerAdvice
- @ExceptionHandler để bắt lỗi xẩy ra và trả về view

```
DispatcherServlet dispatcherServlet = new DispatcherServlet(ctx);
dispatcherServlet.setThrowExceptionIfNoHandlerFound(true);
```

Exception Handler

```
@ControllerAdvice
public class ExceptionController {
    @ExceptionHandler(value = {NoHandlerFoundException.class})
        public String exceptionHandler(Exception exception) {
        System.out.println(exception);
        return "error";
    }
    @ExceptionHandler(value = {Exception.class})
    public String exceptionAll(Exception exception) {
        System.out.println(exception);
        return "error";
    }
}
```

SPRING LOG4J

SPRING LOGGING

Log4j

- Thư viện dùng để in các thông tin log ra màn hình console, file, …
- Tích hợp dễ dàng với Spring MVC để log các thông tin chương trình
- Phân chia các mức độ thông tin log
  - All, debug, info, warn, error, fatal, of

Dependency

```
<!-- https://mvnrepository.com/artifact/log4j/log4j -->
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
</dependency>
```

Log4j.properties

```
# Root logger option
log4j.rootLogger=DEBUG, stdout, file

# Redirect log messages to console
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p
%c{1}:%L - %m%n

# Redirect log messages to a log file
log4j.appender.file=org.apache.log4j.RollingFileAppender
#outputs to Tomcat home
log4j.appender.file.File=E:/ezCloud/Log4j/logs/myapp.log
log4j.appender.file.MaxFileSize=5MB
log4j.appender.file.MaxBackupIndex=10
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L
- %m%n
```

Sử dụng log4j

private static Logger *logger = Logger.getLogger(UserController.class);*

@RequestMapping(value = "/danh-sach-khach-hang", method = RequestMethod.***GET)***

```
public String getAllUser(HttpServletRequest request) {
    logger.info("thong tin khach hang");
    logger.error("thong tin khach hang");
    List<UserDTO> users = userService.getAllUsers();
    request.setAttribute("users", users);
    return "user/listUser";
}
```

SPRING REST

SPRING RESTFUL

Spring Rest

- Rest có thể sử dụng làm web service
- Có thể đọc bởi nhiều thiết bị, ngôn ngữ
- Spring Rest hộ trợ trả về nhiều định dạng khác nhau, JSON, XML, …
- Dễ dàng cấu hình và sử dụng
- Cần thêm các dependency giúp chuyển đổi đối tượng thành json hoặc xml

Dependency

```
<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core -->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.11.1</version>
</dependency>

<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/jackson-dataformat-xml -->
<dependency>
    <groupId>com.fasterxml.jackson.dataformat</groupId>
    <artifactId>jackson-dataformat-xml</artifactId>
    <version>2.11.0</version>
</dependency>
```

Sử dụng

- @ResponseBody
- @RequestBody
- @RestController

QUESTIONS?