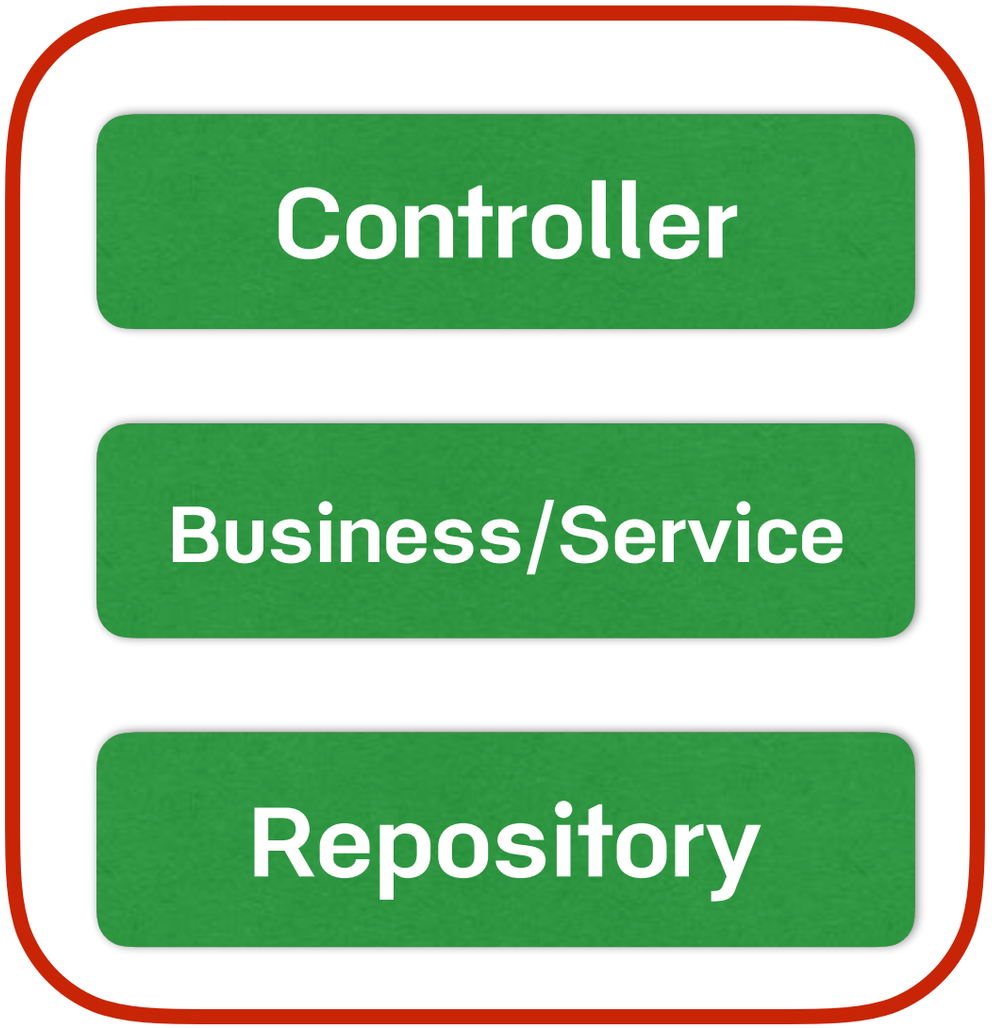
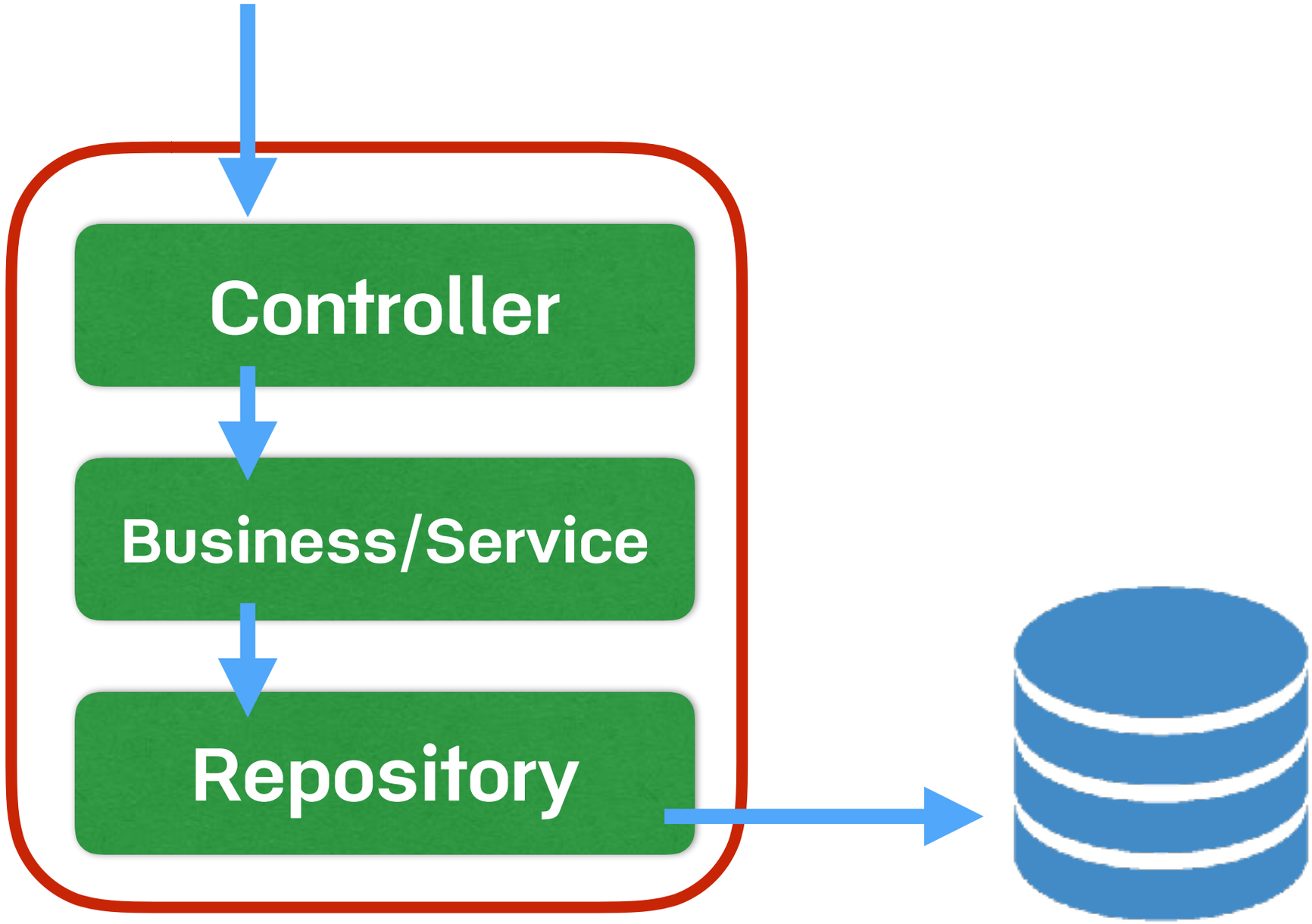
Working with Database

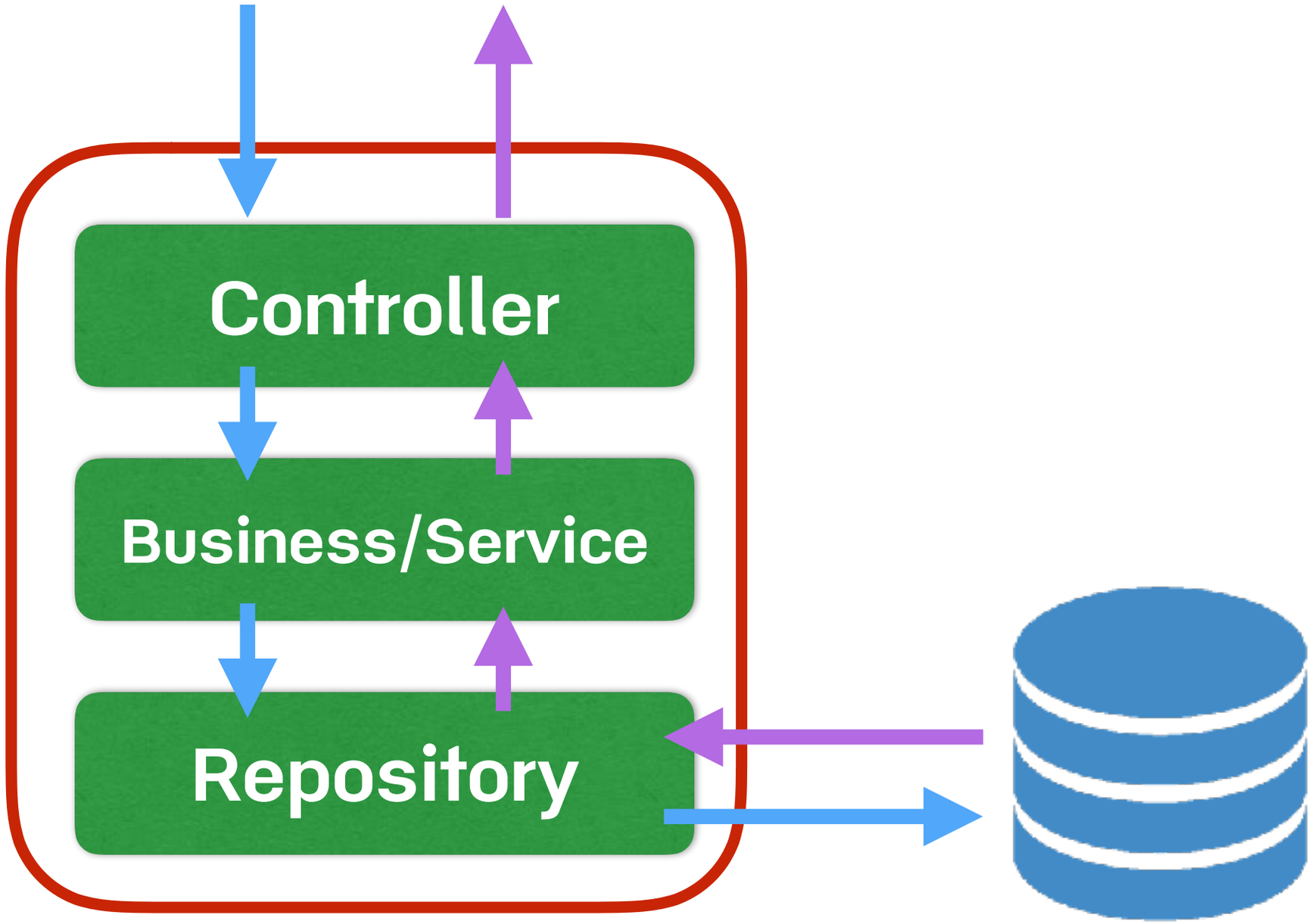
Service structure



Service structure



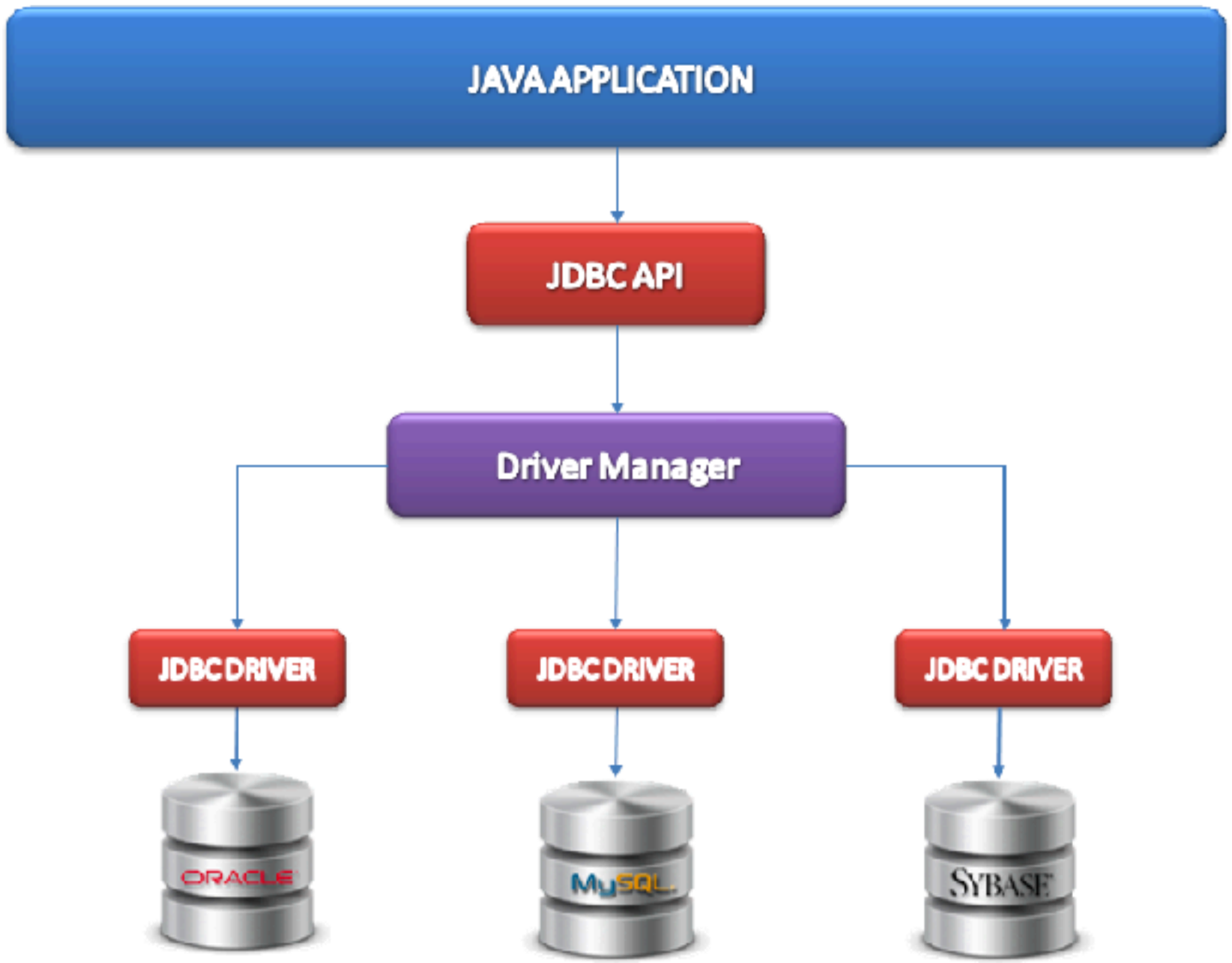
Service structure



Database :: MySQL

https://dev.mysql.com/downloads/

Working with database



JDBC Driver for MySQL

https://dev.mysql.com/downloads/connector/j/

Try by yourself

Create database = **demo**

Try by yourself

Create table = **USERS**

Column = id, firstname, lastname

Workshop

Working with JDBC (MySQL)

Working with Spring Boot

1. Add library

pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jdbc</artifactId>
    <exclusions>
        <exclusion>
            <groupId>org.apache.tomcat</groupId>
            <artifactId>tomcat-jdbc</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-dbcp2</artifactId>
    <version>2.1.1</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>6.0.5</version>
</dependency>
```

2. Config of database

/resources/application.properties

```
server.port = 9001

datasource.driver-class-name=com.mysql.cj.jdbc.Driver
datasource.url=jdbc:mysql://localhost:3306/demo
datasource.username=root
datasource.password=123456
```

3. Config database with code

DatabaseConfiguration.java

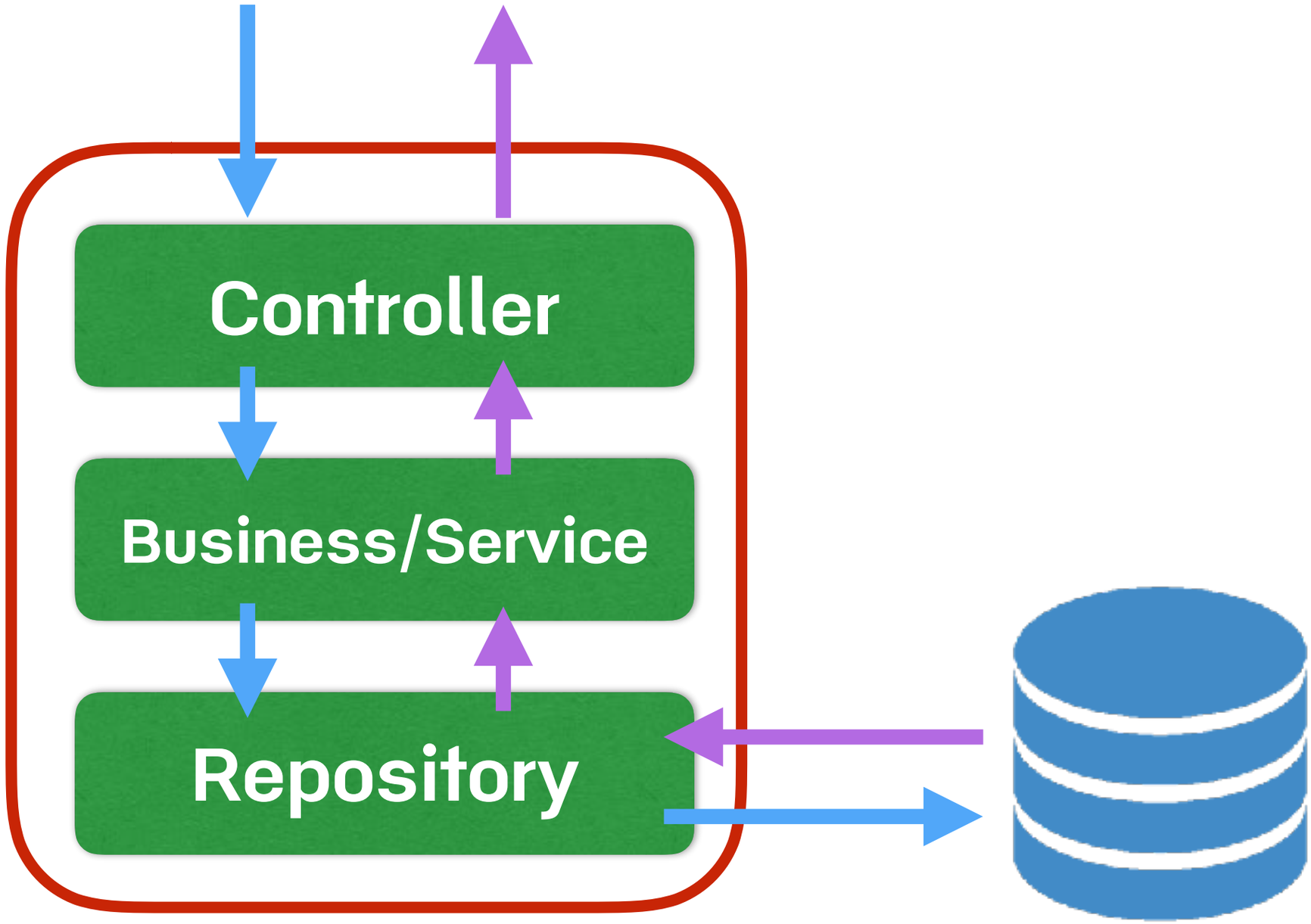
```java
@Configuration
@ComponentScan
@EnableTransactionManagement
@PropertySource(value = {"classpath:application.properties"})
public class DatabaseConfiguration {

    @Autowired
    private Environment environment;

    @Value("${initial-database:false}")
    private String initialDatabase;

    @Bean
    public static PropertySourcesPlaceholderConfigurer placeHolderConfigurer() {
        return new PropertySourcesPlaceholderConfigurer();
    }

    @Bean
    public JdbcTemplate jdbcTemplate(DataSource dataSource) { return new JdbcTemplate(dataSource); }

    @Bean
    public PlatformTransactionManager transactionManager(DataSource dataSource) {
        return new DataSourceTransactionManager(dataSource);
    }

    @Bean
    public DataSource dataSource() {
        BasicDataSource dataSource = new BasicDataSource();
        dataSource.setDriverClassName(environment.getProperty("datasource.driver-class-name"));
        dataSource.setUrl(environment.getProperty("datasource.url"));
        dataSource.setUsername(environment.getProperty("datasource.username"));
        dataSource.setPassword(environment.getProperty("datasource.password"));
        return dataSource;
    }

}
```

3. Config database with code

DatabaseConfiguration.java

```
@Bean
public DataSource dataSource() {
    BasicDataSource dataSource = new BasicDataSource();
    dataSource.setDriverClassName(
            environment.getProperty("datasource.driver-class-name"));
    dataSource.setUrl(
            environment.getProperty("datasource.url"));
    dataSource.setUsername(environment.getProperty("datasource.username"));
    dataSource.setPassword(environment.getProperty("datasource.password"));
    return dataSource;
}
```

Repository ?



4.1 UserRepository.java

Find data of user by id

```java
@Repository
public class UserRepository {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    public UserRepository(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    @Transactional(readOnly = true)
    public User findById(Long id) {
        try {
            String sql = "SELECT * FROM USERS WHERE id=?";
            return this.jdbcTemplate.queryForObject(sql,
                    new Object[]{id}, new UserRowMapper());
        }catch (Exception exception) {
            throw new UserNotFoundException(id);
        }
    }
```

4.2 UserRepository.java

Create a new user

```java
@Transactional
public void save(User user) {
    String sql = "INSERT INTO USERS(id, firstname, lastname) VALUES (?,?,?)";
    this.jdbcTemplate.update(sql, user.getId(),
            user.getFirstname(), user.getLastname());
}
```

4.3 UserRepository.java

Delete user by id

```
@Transactional
public void delete(Long id) {
    String sql = "DELETE FROM USERS WHERE id=?";
    this.jdbcTemplate.update(sql, id);
}
```

4.4 UserRowMapper.java

Mapping columns and Java code

```java
public class UserRowMapper implements RowMapper<User> {
    @Override
    public User mapRow(ResultSet resultSet, int rowNumber) throws SQLException {
        User user = new User();
        user.setId(resultSet.getLong("id"));
        user.setFirstname(resultSet.getString("firstname"));
        user.setLastname(resultSet.getString("lastname"));
        return user;
    }
}
```

Run and Test

$mvn clean install

$java -jar ./target/user-service.jar

Result

GET http://localhost:9001/user?id=1

Body Cookies Headers (4) Tests (0/1)

Pretty Raw Preview JSON


```
{
  "id": 1,
  "firstname": "SOMKIAT",
  "lastname": "PUI"
}
```