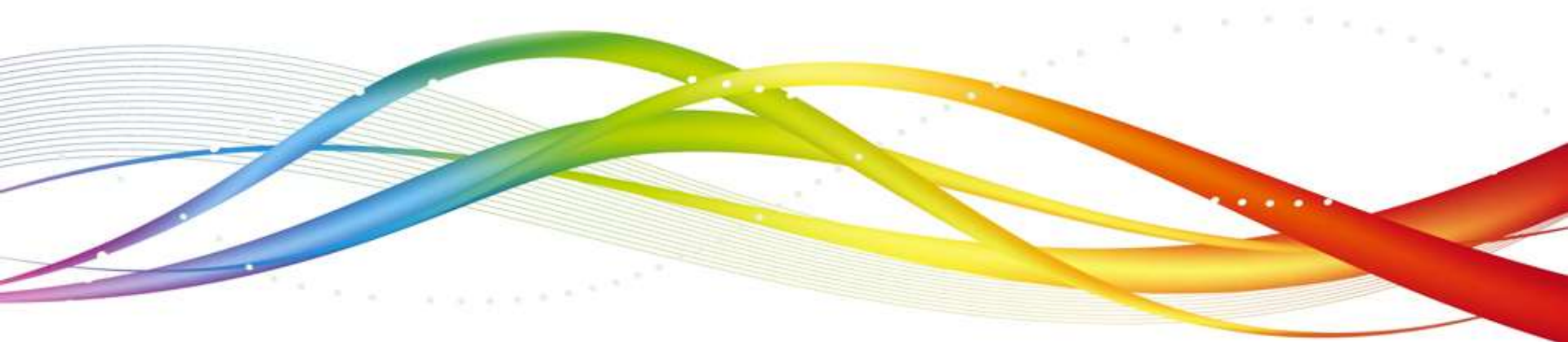
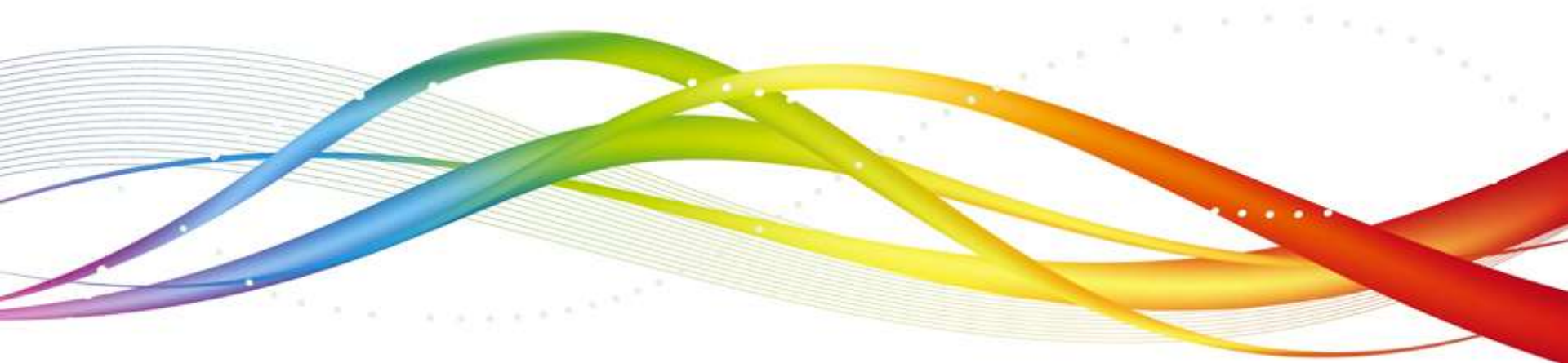




# Spring Database Interaction



# Spring Database Interaction



# Agenda

---

## **Spring Database Interaction**

# Objectives

---

- This module is aimed at:
  - Understanding Spring interaction with database using
    - JDBC
    - ORM (Hibernate)

# Spring JDBCTemplate

---

- **JdbcTemplate** class is the central class in the JDBC core package.
- It simplifies the use of JDBC in Spring applications by handling the creation and release of resources.
- This class performs basic tasks of core JDBC workflow like SQL statement creation and execution.
- This class executes SQL queries, update statements and stored procedure calls, performs iteration over ResultSets and extraction of returned parameter values and catch and handle JDBC exceptions.

# Integrating Hibernate with Spring

---

- The Spring Framework provides integration with **Hibernate**, in terms of:
  - Resource management
  - DAO implementation support
  - transaction strategies
- There are usually two integration styles:
  - using Spring's DAO '**templates**' (helper classes)
  - or coding DAOs against plain Hibernate/JDO/TopLink/etc APIs.
- In both cases, DAOs can be configured through Dependency Injection and participate in Spring's resource and transaction management.

# Spring support for Hibernate

---

- All the individual data access features are usable on their own but integrate nicely with Spring's application context concept
- Provides XML-based configuration and cross-referencing of plain JavaBean instances that don't need to be Spring-aware
- OR mapping inside an **ApplicationContext** or BeanFactory yields the benefits of ease of configuration and deployment.
- Spring AOP provides several aspects that make it possible to declare transaction policies for JavaBeans
- **TransactionProxyFactoryBean** is a convenience proxy class that can intercept method calls to an existing class and apply a transaction context to a transaction bean.

# Spring support for Hibernate (Contd.).

---

- Spring allows you to define resources such as a JDBC DataSource or a Hibernate SessionFactory as beans in the Spring container
- Application objects that need to access resources simply receive references to such pre-defined instances through bean references
- Spring offers Hibernate support, consisting of:
  - HibernateTemplate
  - HibernateInterceptor
  - Hibernate transaction manager



# Hibernate Template

---

- The **HibernateTemplate** class provides many methods that mirror the methods exposed on the Hibernate **Session** interface
- **HibernateTemplate** will ensure that Session instances are properly opened and closed, and automatically participate in transactions.
- The template instances are thread-safe and reusable, they can thus be kept as instance variables of the surrounding class
- The '**SessionFactory**' bean can be injected into DAO setter method, which is required for creating **HibernateTemplate** object
- If you need access to the Session to invoke methods that are not exposed on the HibernateTemplate, you can always drop down to a callback-based approach I

# Summary

---

- In this module, we have learnt:
  - Understanding Spring interaction with database using
    - JDBC
    - ORM (Hibernate)



**Thank You**

