Hibernate - An ORM Tool

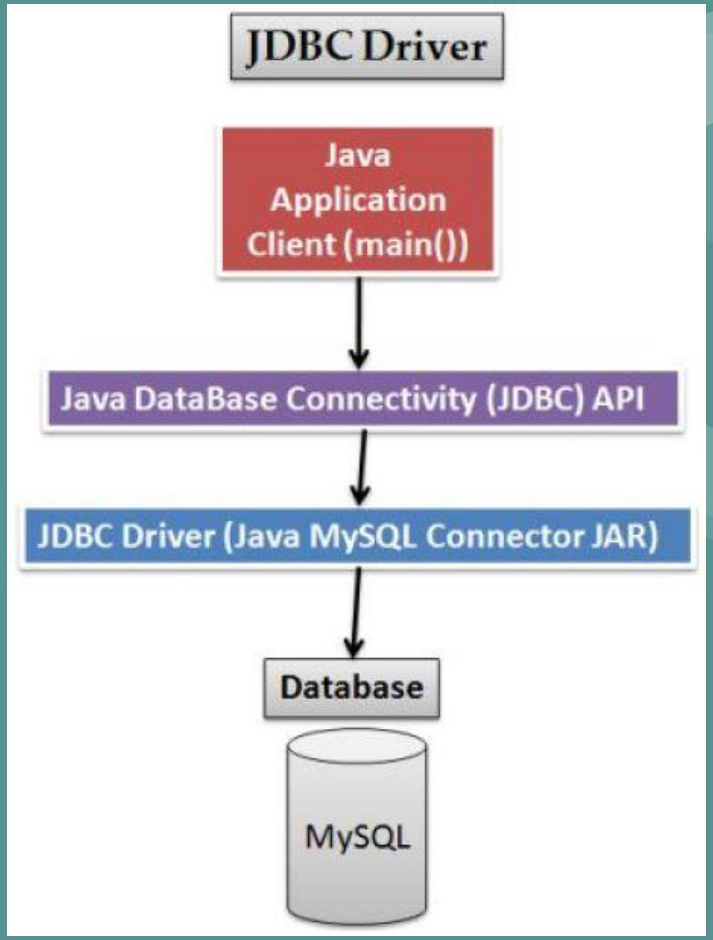
Hibernate ORM is an object-relational mapping tool for the Java programming language.

It provides a framework for mapping an object-oriented domain model to a relational database.

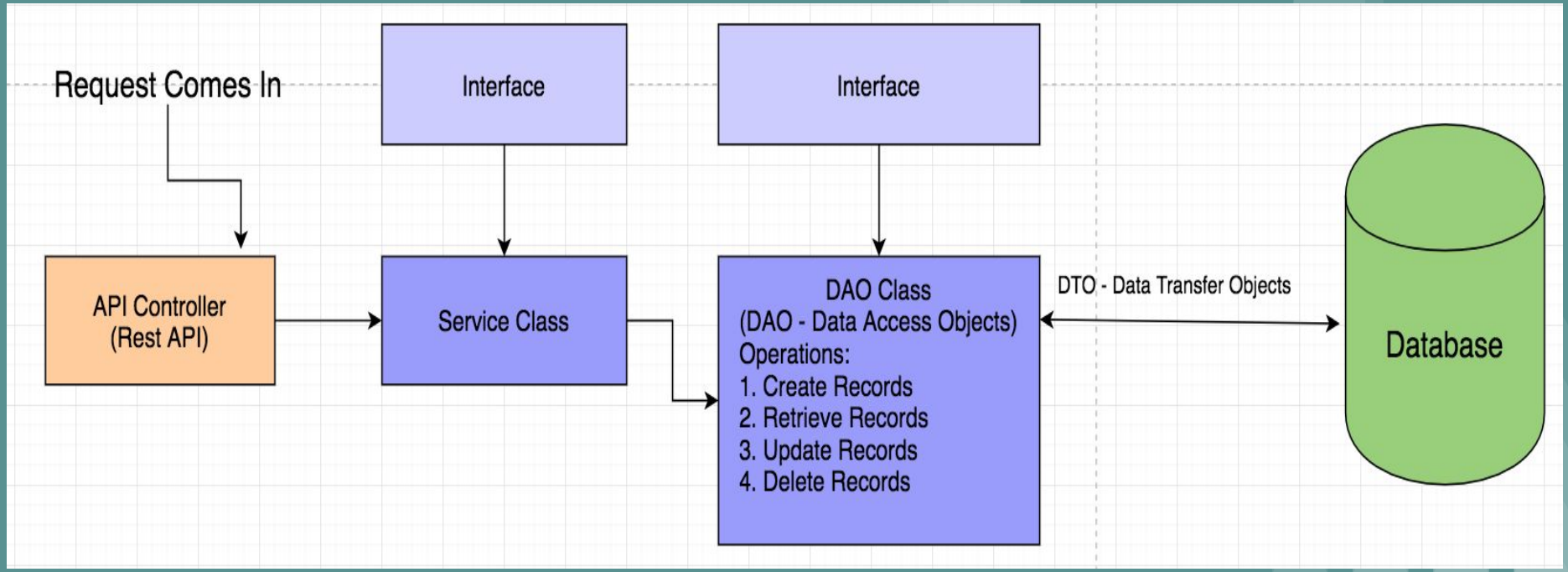
Java

MySQL

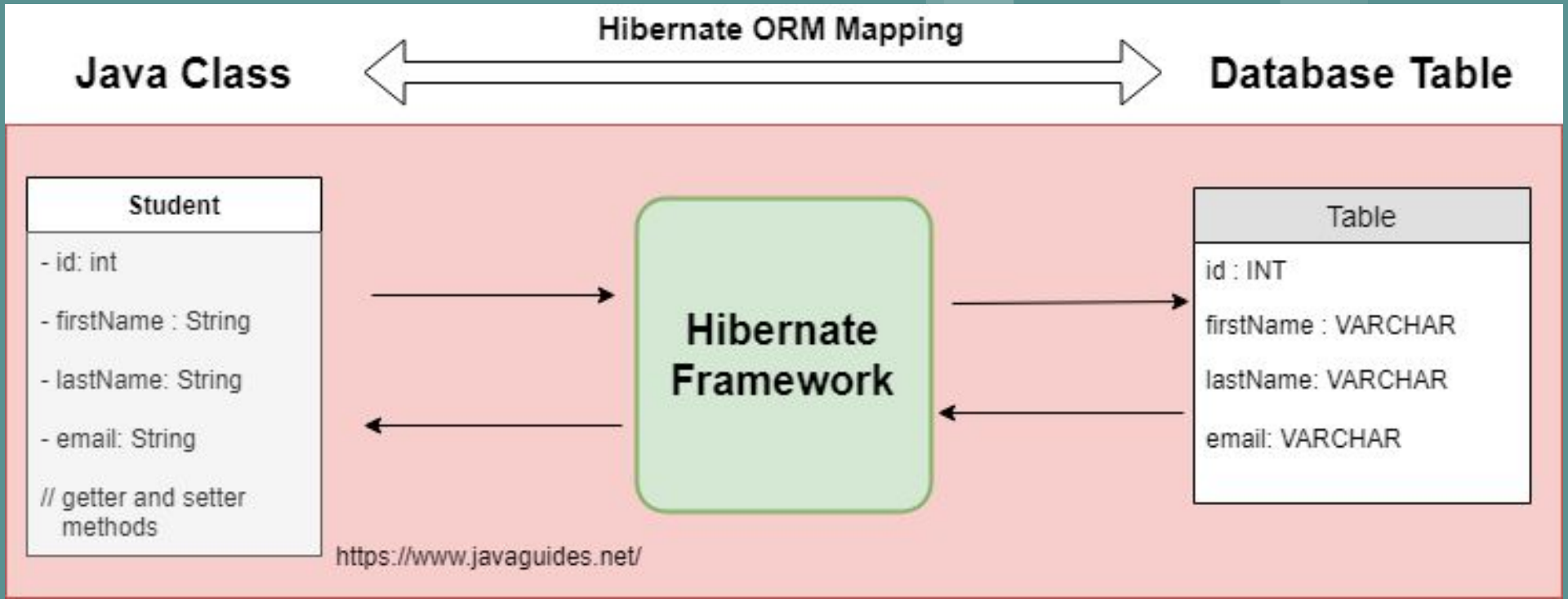
Connection



Hibernate - Spring Implementation



Hibernate - How It Works?



DTO (Data Transfer Object)

```java
@Entity
@Table(name = "app_data.orders")
public class OrdersDTO {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    public Long id;

    @Column(name = "order_id")
    public String orderId;

    @Column(name = "mechant_id")
    public String merchantId;

    @Column(name = "created_date")
    public String createdDate;
}
```

DAO (Data Access Object)

```
@Repository("ordersDAO")
@Transactional
public class OrderDAOImpl implements OrdersDAO {

    @Autowired
    private SessionFactory sessionFactory;

    private Session getCurrentSession() {
        return sessionFactory.getCurrentSession();
    }

    @Override
    public void createOrder(OrdersDTO ordersDTO) {
        getCurrentSession().saveOrUpdate(ordersDTO);
    }

    @Override
    public void deleteOrder(OrdersDTO ordersDTO) {
        getCurrentSession().delete(ordersDTO);
    }
```

DAO (Data Access Object)

```
    @Override
    public void update(OrdersDTO ordersDTO) {
        getCurrentSession().saveOrUpdate(ordersDTO);
    }

    @SuppressWarnings("unchecked")
    @Override
    public List<OrdersDTO> listOrders() {
        return sessionFactory.getCurrentSession().createQuery("FROM OrdersDTO").getResultList();
    }

    @Override
    public OrdersDTO findByOrderId(String orderId) {
        Session session = sessionFactory.getCurrentSession();
        return (OrdersDTO) session.createQuery("FROM OrdersDTO where order_id = '" + orderId + "'").getResultList()
                .get(0);
    }
```

Service Layer

```java
@Service
public class OrderServiceImpl implements OrderService {

    @Autowired
    private OrderDAOImpl ordersDAO;

    @Override
    public void createOrder(OrdersDTO ordersDTO) {
        ordersDAO.createOrder(ordersDTO);
    }

    @Override
    public void update(OrdersDTO ordersDTO) {
        ordersDAO.update(ordersDTO);
    }

    @Override
    public void deleteOrder(OrdersDTO ordersDTO) {
        ordersDAO.deleteOrder(ordersDTO);
    }

    @Override
    public OrdersDTO findByOrderId(String orderId) {
        return ordersDAO.findByOrderId(orderId);
    }

    @Override
    public List<OrdersDTO> listOrders() {
        return ordersDAO.listOrders();
    }

}
```

Application Properties

```
jdbc.driverClassName = com.mysql.cj.jdbc.Driver
jdbc.url = jdbc:mysql://localhost:3307/app_data
jdbc.username = root
jdbc.password =
hibernate.dialect = org.hibernate.dialect.MySQLDialect
hibernate.show_sql = false
hibernate.format_sql = false
```

Creating Beans

```
private Properties hibernateProperties() {
    Properties properties = new Properties();
    properties.put("hibernate.dialect", hibernateConfiguration.dialect);
    properties.put("hibernate.show_sql", hibernateConfiguration.showSQL);
    properties.put("hibernate.format_sql", hibernateConfiguration.formatSQL);
    return properties;
}
```

```
@Bean
public DataSource dataSource() {
    DriverManagerDataSource dataSource = new DriverManagerDataSource();
    dataSource.setDriverClassName(hibernateConfiguration.jdbcDriverClassName);
    dataSource.setUrl(hibernateConfiguration.url);
    dataSource.setUsername(hibernateConfiguration.userName);
    dataSource.setPassword(hibernateConfiguration.password);
    return dataSource;
}
```

Creating Beans

```
@Bean
public LocalSessionFactoryBean sessionFactory() {
    LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();
    sessionFactory.setDataSource(dataSource());
    sessionFactory.setPackagesToScan(new String[] { "com.training" });
    sessionFactory.setHibernateProperties(hibernateProperties());
    return sessionFactory;
}
```

```
@Bean
@Autowired
public HibernateTransactionManager transactionManager(SessionFactory s) {
    HibernateTransactionManager txManager = new HibernateTransactionManager();
    txManager.setSessionFactory(s);
    return txManager;
}
```