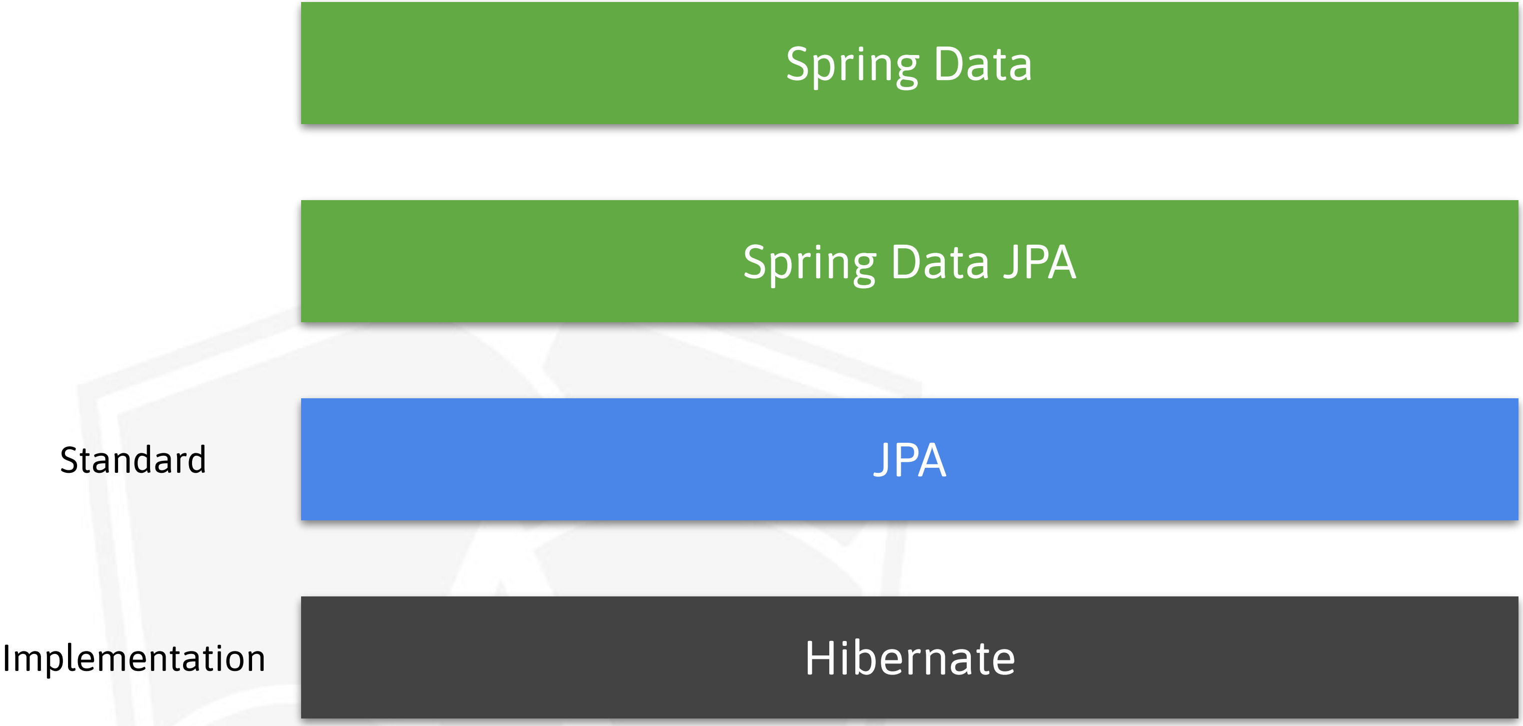
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Spring Workshop

4 - Data



Spring Data

Spring Data JPA

Standard JPA

Implementation Hibernate

@Entity and other Annotations

- Associates a database table to a class
- Classname can be different from table name
- Be aware of Naming Strategies
  - Implicit
  - Physical
  - Spring

Entity

- @Entity
- @Table
- @Id
- @GeneratedValue
  - IDENTITY: auto increment MySQL, identity in SQL Server
  - SEQUENCE: Oracle
  - (TABLE)
- @Transient
- @Column
- BeanValidation

Repository

- Interface implemented by Spring Data
- Manages database specific actions
- Common Subtypes
    - (List)CrudRepository
    - (List)PagingAndSortingRepository
    - JpaRepository
- Customisable via find* methods

Customer as Entity

```
public class Customer {

  private Long id;

  private String firstname;
  private String lastname;

  private Boolean signedGdpr;
}
```

Customer as Entity

```
@Entity
public class Customer {

  private Long id;

  private String firstname;
  private String lastname;

  private Boolean signedGdpr;
}
```

Customer as Entity

```
@Entity
public class Customer {

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

  private String firstname;
  private String lastname;

  private Boolean signedGdpr;
}
```

CustomersRepository

```java
package com.softarc.eternal.data;

import com.softarc.eternal.domain.Customer;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CustomersRepository extends JpaRepository {}
```

CustomersRepository

```java
package com.softarc.eternal.data;

import com.softarc.eternal.domain.Customer;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CustomersRepository extends JpaRepository<Customer, Long> {}
```

CustomersRepository

```java
package com.softarc.eternal.data;

import com.softarc.eternal.domain.Customer;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CustomersRepository extends JpaRepository<Customer, Long> {
  List<Customer> findByFirstnameOrLastnameOrderByLastname(
    String firstname,
    String lastname
  );
}
```

Database Connection in application.yml for H2

```
spring:
  datasource:
    url: jdbc:h2:./dev-db
    driver-class-name: org.h2.Driver
    username: sa
    password: 'Sag ich dir nicht :)'
```

- H2 is embedded/file/server
- Possibility to use multiple databases
- Make use of profiles
- Credentials
  - plaintext
  - encrypted
  - OS env

Further useful configuration properties

```yaml
spring:
  jpa:
    hibernate:
      ddl-auto: create-drop

logging:
  level:
    org.hibernate.orm.jdbc.bind: trace

spring:
  jpa:
    show-sql: true
```

Impact on Testing

- No on impact on Unit Tests (no DI)
- Override the datasource properties via @SpringBootTest
  - `jdbc:h2:mem:holidays-controller`
  - in-memory instance
- Mock the Repository via @MockedBean in Integration Tests

Demo

```
private void executeLoad(long timeout, int usersCount) {
    showDebugInfo(timeout);
    Load.setPages(URL, parsingTimeout);
    Load.setTimeout(timeout);
    List<Load> threads = new ArrayList<>();
    for (int i = 0; i < usersCount; i++) {
        threads.add(new Load(this.URL));
    }
    logger.info( s: usersCount + "threads are created");
    for (Load thread : threads) {
        thread.start();
    }
    logger.info( s: "All threads are started");
    progressInfo(timeout);
    System.out.print("........DONE\nProcessing with data...\n");
}
private void executeAvailability(long timeout, int
}
private void
```

Lab Time

@DataJpaTest

- In-Memory Database is normally used
- Transactional with Rollback
- SQL commands and queries are printed

@DataJpaTest

```java
@DataJpaTest(
  properties = {
    "spring.datasource.url=jdbc:h2:mem:test",
    "spring.jpa.hibernate.ddl-auto=create-drop",
  }
)
public class CustomersControllerIntegrationTest {

  @Autowired
  CustomersRepository repository;

  @Test
  void testRepo() {
    var franz = CustomerMother.franz().build();
    repository.save(franz);
    Customer customer = repository.findAll().get(0);
    assertThat(customer)
      .usingRecursiveComparison()
      .ignoringFields("id")
      .isEqualTo(franz);
  }
}
```

Flyway

- Versioned Migration Tool
- Uses SQL
- Files in resources with convention V{n}__{name}.sql
- Liquibase (database agnostic)
- Careful with line ending (sql files are hashed)
- Rollback possible

Flyway / MySql Setup

1. Setup MySQL via Docker or in another way
2. Register datasource.url
    a. `jdbc:mysql://localhost:3306/eternal`
3. Add dependencies
    a.
    ```
    implementation 'mysql:mysql-connector-java'
    implementation 'org.flywaydb:flyway-core'
    implementation 'org.flywaydb:flyway-mysql'
    ```
4. Create init schema in src/main/resources/db/main/V1__init.sql
5. Restart Spring Boot

Relations

- Used quite often
  - @ManyToOne
  - @OneToMay
- Rarely
  - @OneToOne
  - @ManyToMany

Relations

```
@Entity
public class Country {

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

  private String name;

  @OneToMany(mappedBy = "country")
  private List<Customer> customers;
}
```

Relations

```
@Entity
public class Customer {

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

  private String firstname;
  private String lastname;

  private Boolean signedGdpr;

  @ManyToOne
  @JoinColumn(name = "country_id") // optional
  private Country country;
}
```

Demo

```
private void executeLoad(long timeout, int usersCount) {
    showDebugInfo(timeout);
    Load.setPages(URL, parsingTimeout);
    Load.setTimeout(timeout);
    List<Load> threads = new ArrayList<>();
    for (int i = 0; i < usersCount; i++) {
        threads.add(new Load(this.URL));
    }
    logger.info( s: usersCount + " threads are created");
    for (Load thread : threads) {
        thread.start();
    }
    logger.info( s: "All threads are started");
    progressInfo(timeout);
    System.out.print("........DONE\nProcessing with data...\n");
}

private void executeAvailability(long timeout, int
}

private void
```

Lab Time