

Spring data: Paging, Sorting, Custom Repositories, Auditing, Locking

Pageable and Sort

Pluralsight Offline Player

Paging and Sorting

```
Repository<T, ID extends Serializable>  
├── CrudRepository<T, ID extends Serializable>  
│   └── PagingAndSortingRepository<T, ID extends Serializable> ──> Iterable<T> findAll(Sort sort);  
│   └── Page<T> findAll(Pageable pageable);  
└── JpaRepository<T, ID extends Serializable>
```

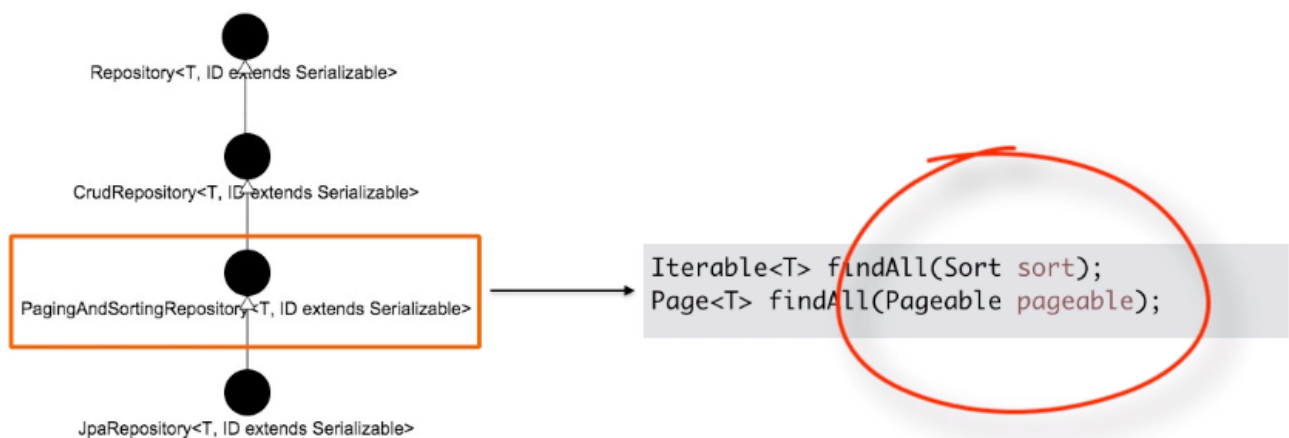
More Fun with Queries
29:38

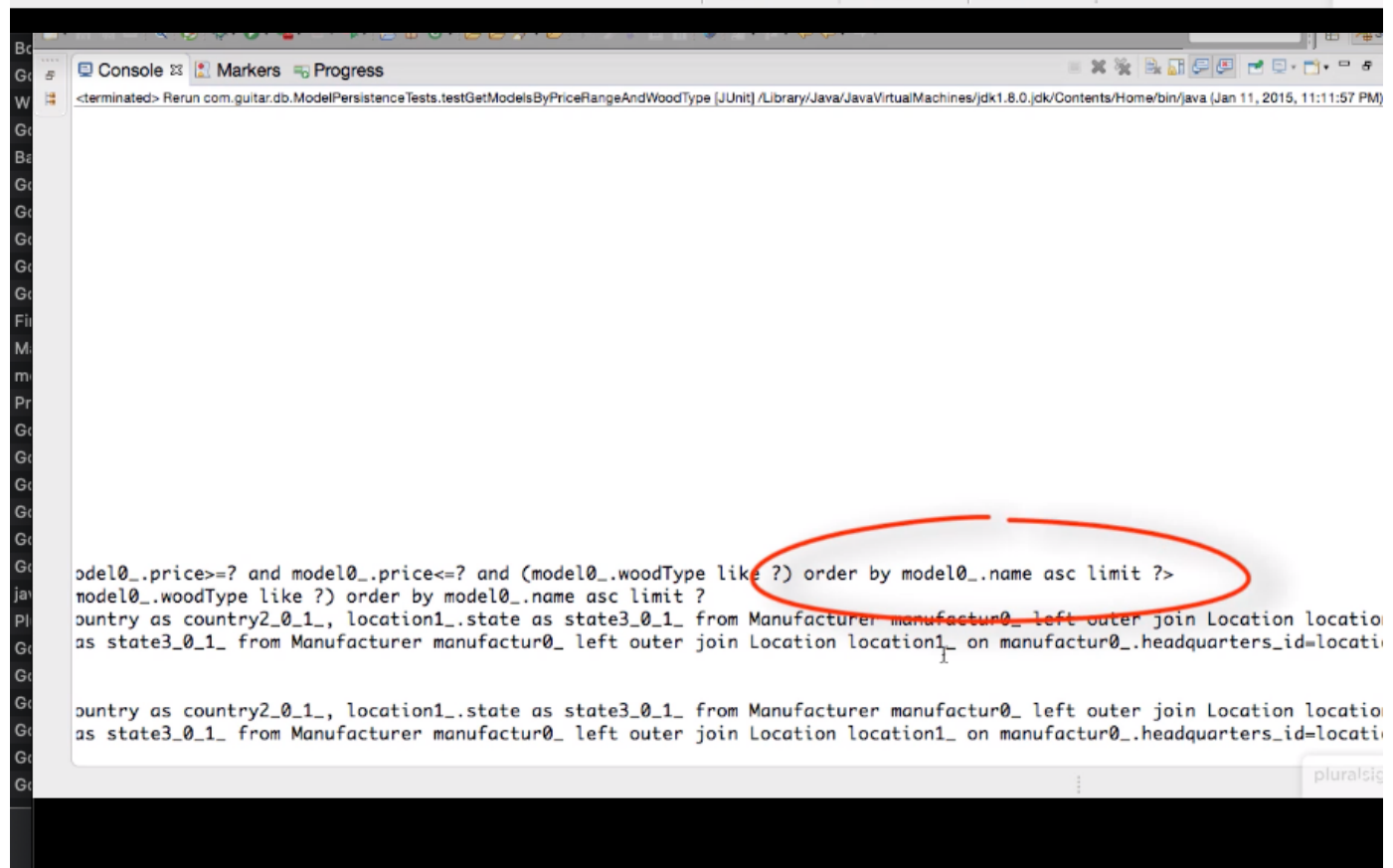
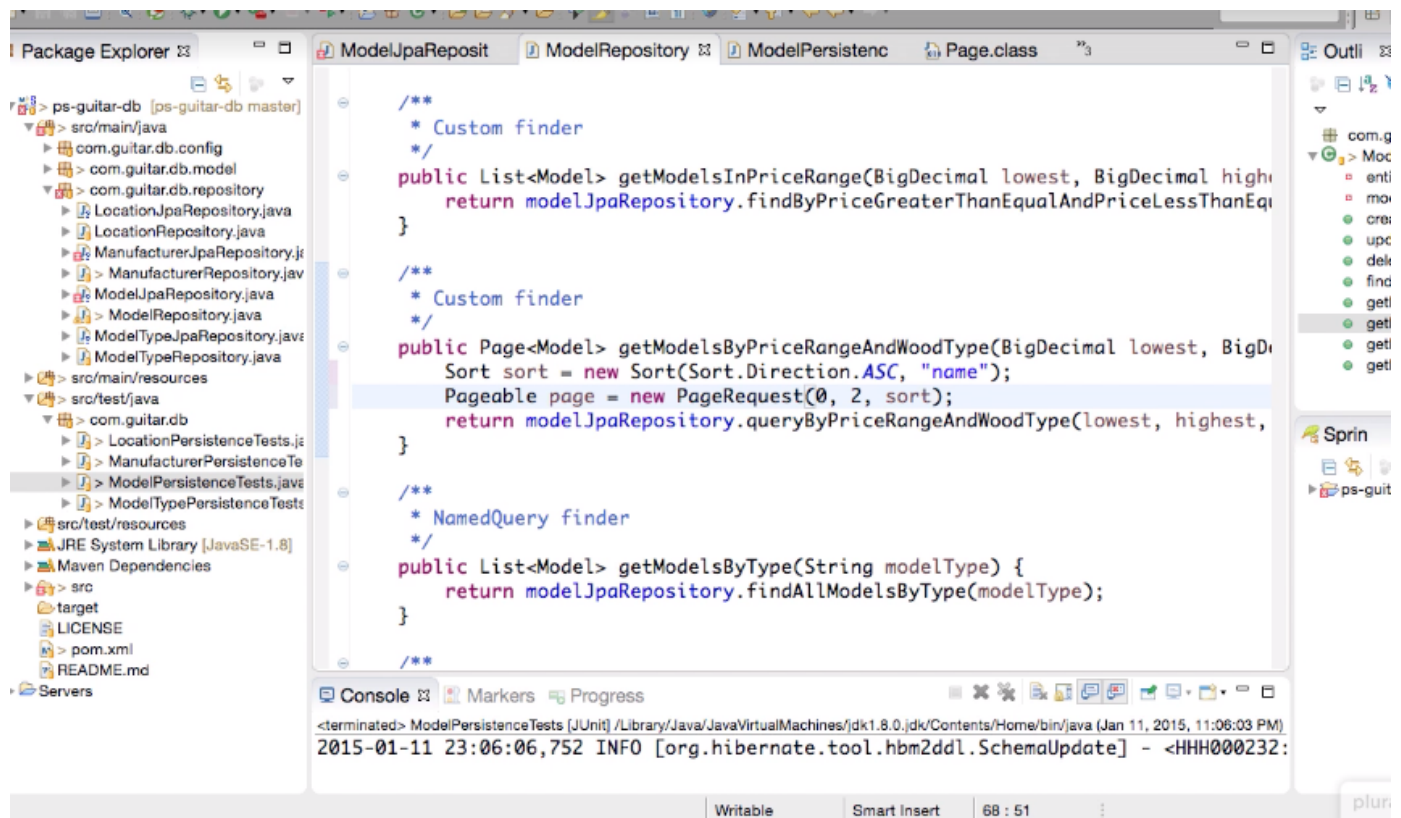
- Introduction
- Demo: @Query Annotation
- @Query Annotation
- @Query Options
- JPA NamedQueries
- Demo: NamedQueries
- Native SQL Support
- Demo: NamedNativeQueries
- Tip: Where Should I Put My Queries?
- JpaRepository Query Precedence
- Summary

Advanced Features
29:28

- Introduction
- Paging and Sorting** 0:40 / 1:21
- Demo: Paging and Sorting
- Custom Repositories
- Demo: Custom Repositories
- Auditing Support
- Locking
- Summary

Paging and Sorting

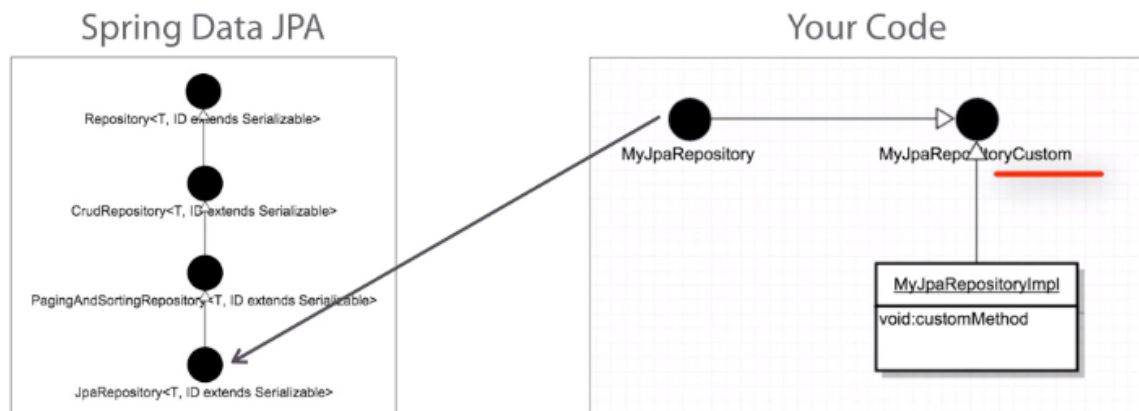




Preuve: on voit cela dans le code sql genere avec order by... et limit ...

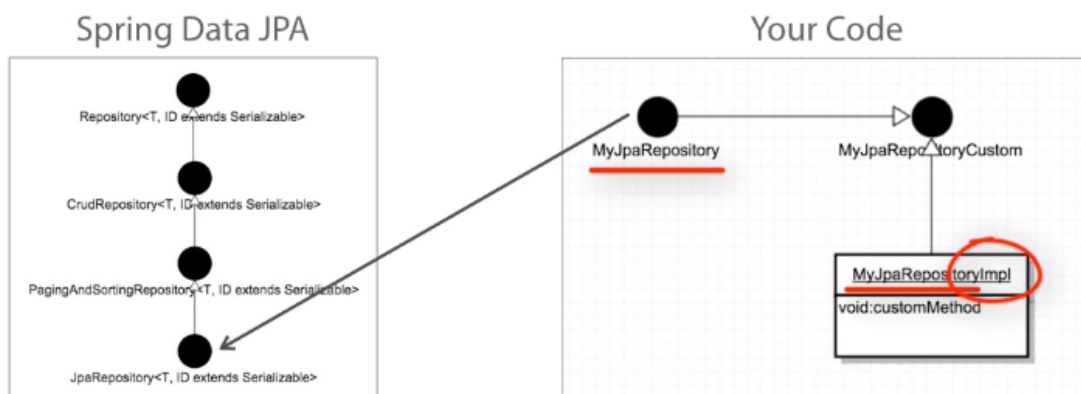
Custom repository

Custom Repositories



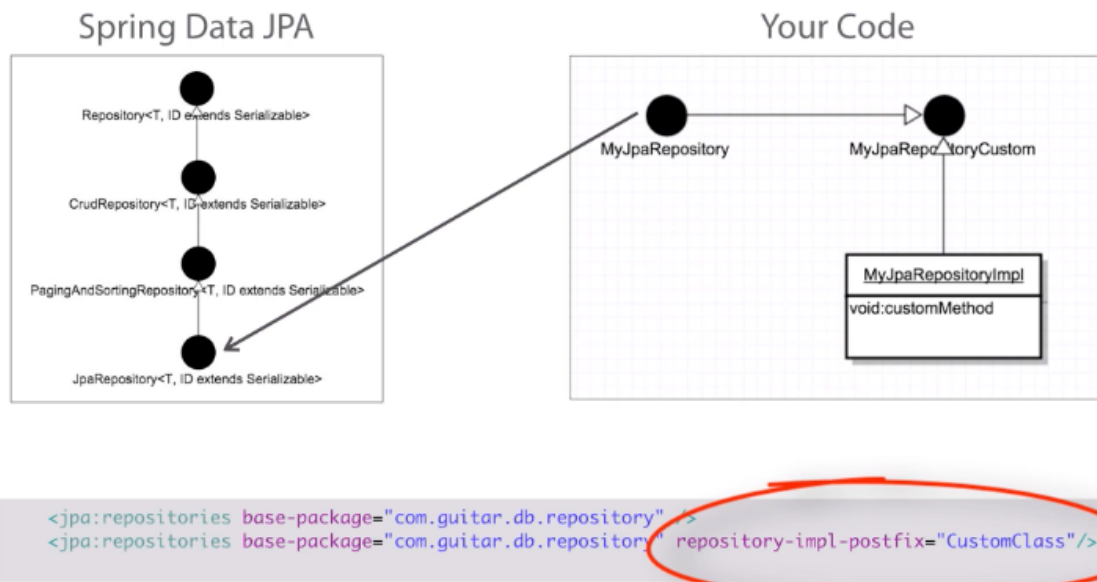
pluralsight

Custom Repositories



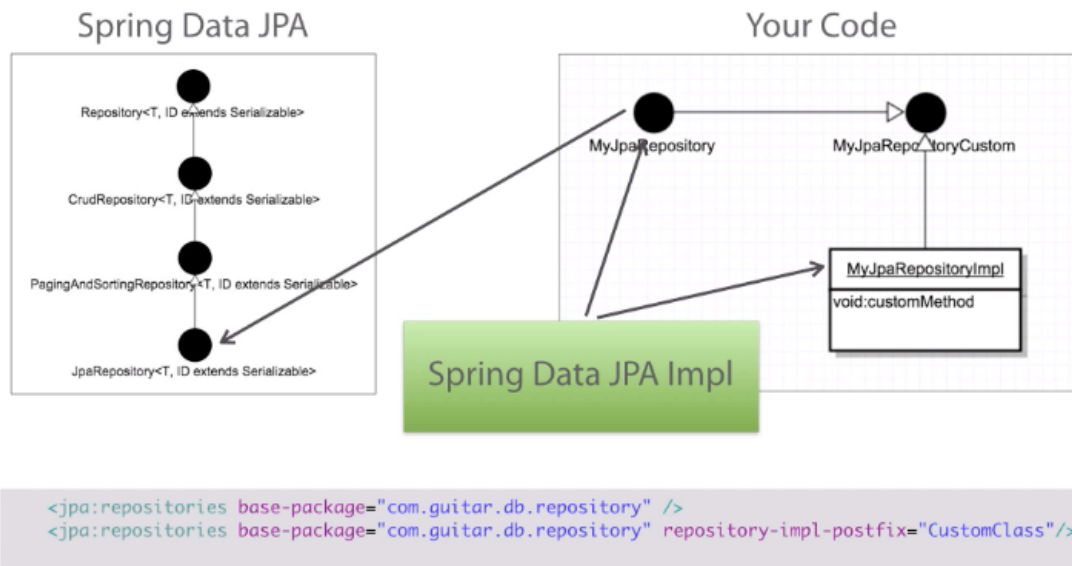
pluralsight

Custom Repositories



pluralsight

Custom Repositories



pluralsight

Auditing support

need of enterprises:

- tracking Data access layer when writing data
 - who created the data?
 - when was it created
 - who modified the record?
 - when was the last modified?

Auditing Support

Spring Data JPA Annotations

- @CreatedBy
- @CreatedDate
- @LastModifiedBy
- @LastModifiedDate

Auditing Support

Spring Data JPA Annotations

- @CreatedBy
- @CreatedDate
- @LastModifiedBy
- @LastModifiedDate

```
@Entity
public class Model {
    @CreatedBy
    private User user;


    @CreatedDate
    private DateTime createdDate;
}
```

plura

Auditing Support

Spring Data JPA Annotations

- @CreatedBy
- @CreatedDate
- @LastModifiedBy
- @LastModifiedDate



Auto Set

```
@Entity
public class Model {
    @CreatedBy
    private User user;

    @CreatedDate
    private DateTime createdDate;
}
```

pluralsight

Auditing Support

Spring Data JPA Annotations

- @CreatedBy
- @CreatedDate
- @LastModifiedBy
- @LastModifiedDate

Auto Set

```
@Entity
public class Model {
    @CreatedBy
    private User user;

    @CreatedDate
    private DateTime createdDate;
}
```

```
public class SecurityAuditorAware
    implements AuditorAware<User> {

    public User getCurrentAuditor() {
        //...
        return user;
    }
}
```

pluralsight

Auditing Support

Spring Data JPA Annotations

- @CreatedBy
- @CreatedDate
- @LastModifiedBy
- @LastModifiedDate

Auto Set

```
@Entity
public class Model {
    @CreatedBy
    private User user;

    @CreatedDate
    private DateTime createdDate;
}
```

```
public class SecurityAuditorAware
    implements AuditorAware<User> {

    public User getCurrentAuditor() {
        //...
        return user;
    }
}
```

```
<jpa:auditing auditor-aware-ref="securityAuditorAware" />
```

```
@Configuration
@EnableJpaAuditing
public class Config {
    @Bean
    public AuditorAware<User> auditorProvider() {
        return new SecurityAuditorAware();
    }
}
```

pluralsight

Locking

pluralsight

3. Auditing

Locking

for concurrent access data

Locking

```
@Entity
public class Model {
    @Version
    private int version;
    ...
}
```

pluralsight

Locking

```
@Entity
public class Model {
    @Version
    private int version;
    ...
}
```

Optimistic Locking

- If version number doesn't match, throws OptimisticLockException

pluralsight

Locking

```
@Entity
public class Model {
    @Version
    private int version;
    ...
}
```

Optimistic Locking

- If version number doesn't match, throws OptimisticLockException

Pessimistic Locking

- Long term locks the data for the transaction duration, preventing others from accessing the data until the transaction commits

pluralsight

Locking

```
@Entity
public class Model {
    @Version
    private int version;
    ...
}
```

Optimistic Locking

- If version number doesn't match, throws OptimisticLockException

Pessimistic Locking

- Long term locks the data for the transaction duration, preventing others from accessing the data until the transaction commits

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
List<Model> findByModelTypeNameIn(List<String> types);
```

pluralsigh