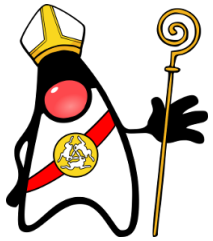


Build faster persistence layers with Spring Data JPA

Thorben Janssen

Independent consultant, trainer and author

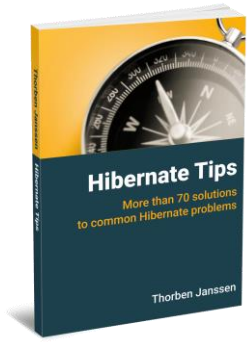




Co-Organizer
JUG Paderborn



thorben-janssen.com



hibernate-tips.com

Persistence Hub

persistence-hub.com

thorben-janssen.com

@thjanssen123

/c/ThoughtsOnJava

thorbenjanssenofficial


Fast & Scalable

Performance

- Find problems as early as possible
- Avoid / Fix performance problems

Performance

- Persistence provider: Hibernate
  - Follow best practices
  - Avoid pitfalls



Identify Issues

Hibernate Statistics

- Activate via system property
  - hibernate.generate_statistics = true
- Configure logging
  - org.hibernate.stat = DEBUG

Code Samples



Association Fetching

FetchType

- Defines when the relationship will be fetched
- Static definition in entity mapping

```
@ManyToOne(fetch = FetchType.LAZY)
private Publisher publisher;
```

FetchType

- Lazy
  - Relationship gets loaded at first access
  - Default for to-many relationships
- Eager
  - Loads relationships immediately
  - Default for to-one relationships

Recommendation

- To-many relationships
  - Stick to the default mapping (FetchType.LAZY)
  - Use eager fetching for specific queries, if required
- To-one relationships
  - Check existing mappings individually
  - Use FetchType.LAZY for new ones



Query-Specific Fetching

N+1 Select

- Most common problem
  - Lazy fetching of related entities

```
List<ChessPlayer> players = playerRepo.findWithTournamentsBy();

for (ChessPlayer player : players) {
        log.info(player.getFirstName() + " " + player.getLastName() +
                " played in “ + player.getTournaments().size() + " tournaments.");
}
```

Query-Specific fetching

- Fetch all required entities with one query
  - JOIN FETCH
  - EntityGraph

Code Samples



Many-to-Many Association

Many-to-Many

- Inefficient handling of List
  - Remove all associations
  - Add remaining ones
- Use Set instead

Code Samples



Projections

Projections

- Entities
- Scalar values
- DTO

DTO Projections

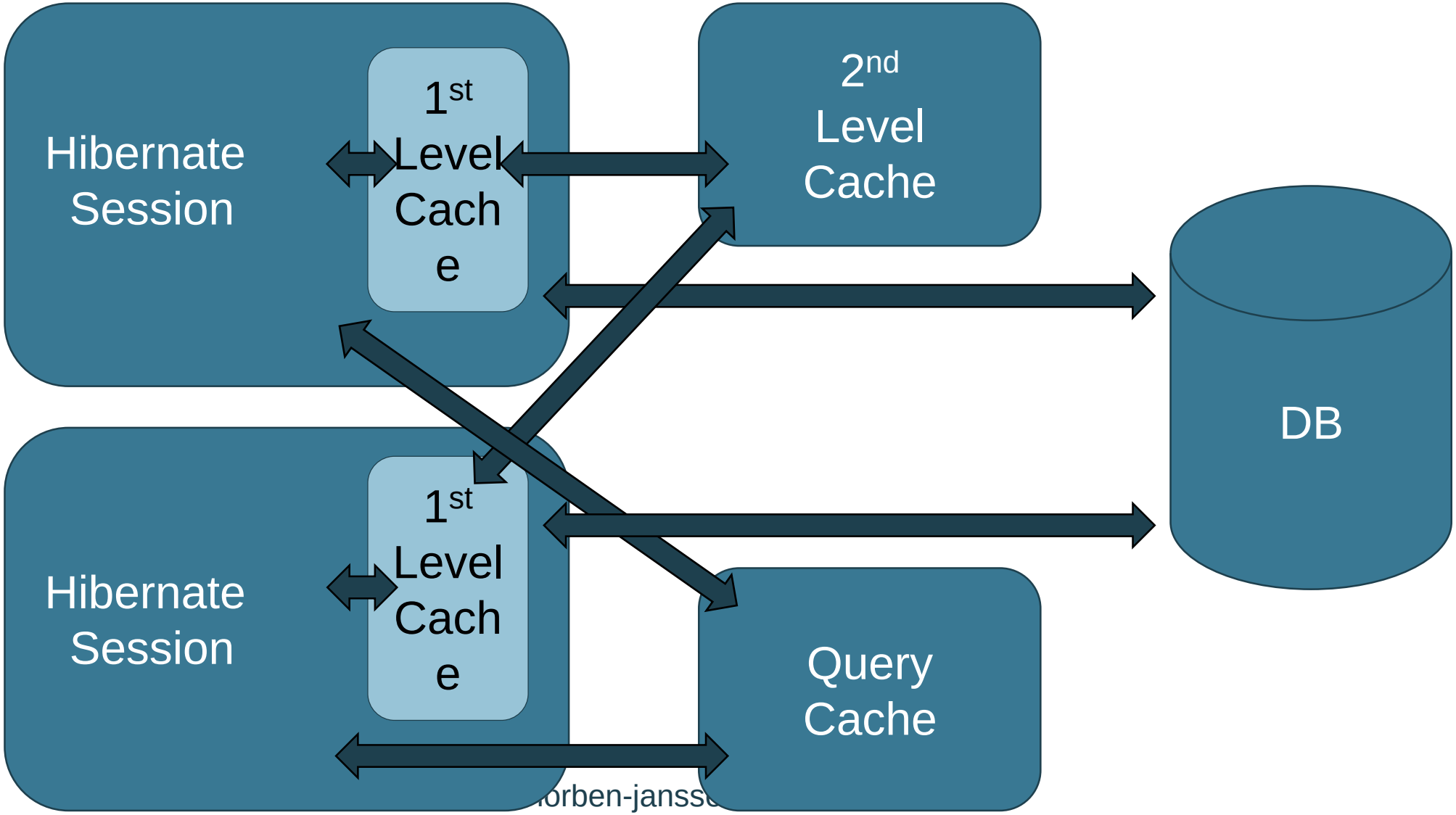
- DTO records or classes
- DTO interfaces
  - Spring Data JPA generates class

Code Samples



Caching

Caching


Hibernate Session
1st Level Cache
Hibernate Session
1st Level Cache
2nd Level Cache
Query Cache
DB



2nd Level Cache

Shared Cache Mode

- ALL — all entities
- NONE — no entities
- ENABLE_SELECTIVE — requires activation
- DISABLE_SELECTIVE — can be deactivated
- UNSPECIFIED — use default

settings

Code Samples

The Persistence Hub

Get your skills to the next level!

- Certification Courses
- Expert Sessions
- Java Persistence News
- Coding Problems

Persistence-hub.com