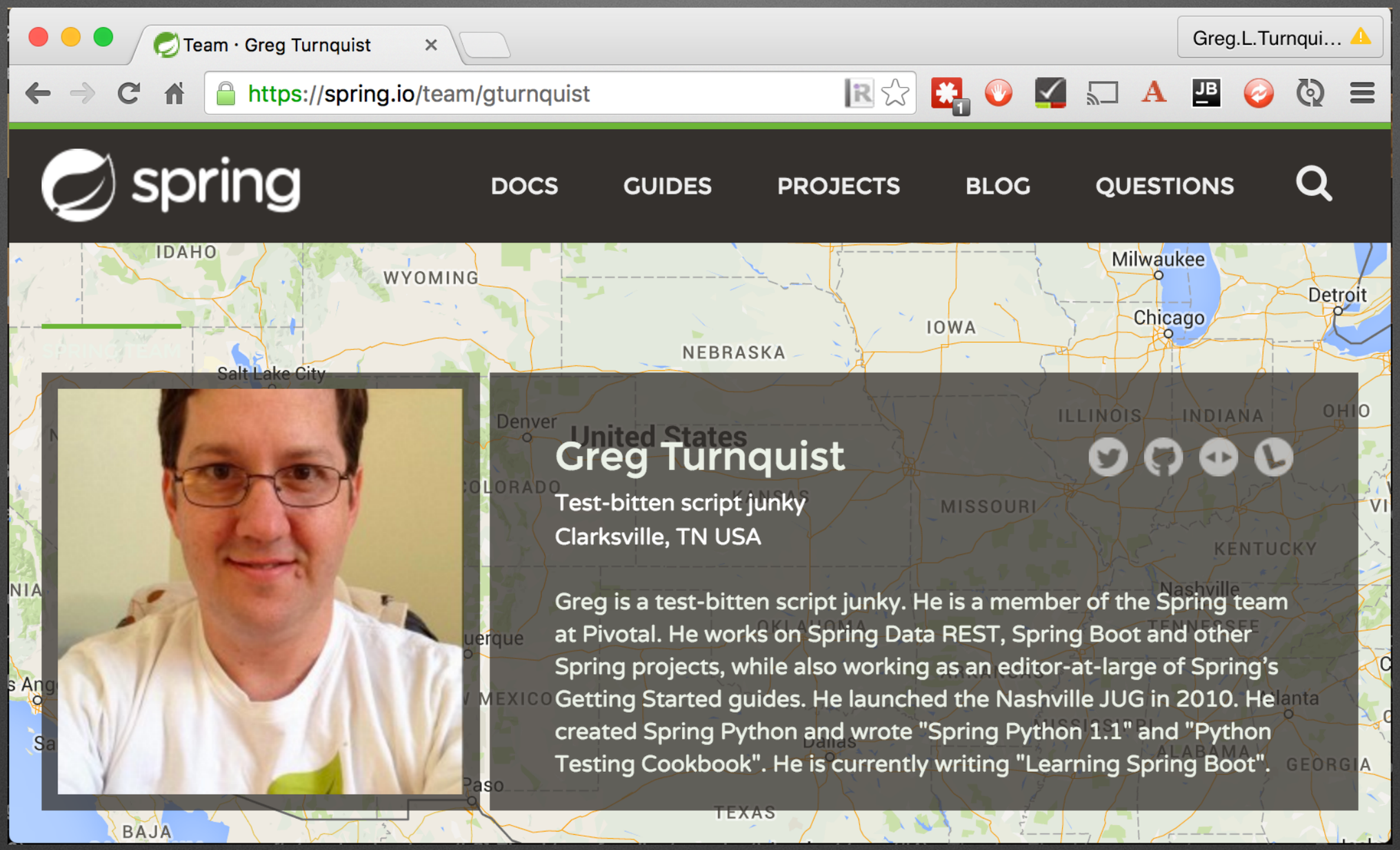
Introduction to Spring Data

Greg Turnquist
@gregturn
github.com/gregturn

DEVNEXUS™

spring by Pivotal



GregLTurnquist.com/list

DEVNEXUS™

spring by Pivotal

Team · Greg Turnquist

Greg.L.Turnqui…

https://spring.io/team/gturnquist

spring

DOCS GUIDES PROJECTS BLOG QUESTIONS

Greg Turnquist

Test-bitten script junky
Clarksville, TN USA

Greg is a test-bitten script junky. He is a member of th
at Pivotal. He works on Spring Data REST, Spring Boo
Spring projects, while also working as an editor-at-lar
Getting Started guides. He launched the Nashville JU
created Spring Python and wrote "Spring Python 1.1
Testing Cookbook". He is currently writing "Learning

GregLTurnquist.com/list

DEVNEXUS™

spring by Pivotal

Pivotal™

DEVNEXUS™

spring by Pivotal™

Pivotal™

We're hiring!

(Atlanta + the world)

http://pivotal.io/careers

DEVNEXUS™

spring by Pivotal™

It's 2016

```java
@Data
@Entity
class Employee {
    @Id
    @GeneratedValue
    private Long id;
    private String name;
    private String role;

    @ManyToOne
    private Manager manager;

    public Employee(String name, String role) {
        this.name = name;
        this.role = role;
    }
}
```

It's 2016

Why are we still writing…

It's 2016

Why are we still writing…

```
select * from EMPLOYEE
where FIRST_NAME = %1
```

It's 2016

Why are we still writing…

```
select * from EMPLOYEE
where FIRST_NAME = %1
```

```
select e from Employee e
where e.firstName = :name
```

It's 2016

Why are we still writing…

```
select * from EMPLOYEE
where FIRST_NAME = %1
```

```
select e from Employee e
where e.firstName = :name
```

```
create
        .select()
        .from(EMPLOYEE)
        .where(EMPLOYEE.FIRST_NAME
                .equal(name))
        .fetch()
```

It's 2016

Why are we still writing…

SQL

```
select * from EMPLOYEE
where FIRST_NAME = %1
```

JPA

```
select e from Employee e
where e.firstName = :name
```

jOOQ

```
create
      .select()
      .from(EMPLOYEE)
      .where(EMPLOYEE.FIRST_NAME
            .equal(name))
      .fetch()
```

When we could write...

```
interface EmployeeRepository extends CrudRepository<Employee, Long> {

    List<Employee> findByFirstName(String firstName);
}
```

When we could write...

```
interface EmployeeRepository extends CrudRepository<Employee, Long> {

    List<Employee> findByFirstName(String firstName);
}
```

...includes **save()**,
findOne(),
findAll()
exists(),
delete(),
and **count()**

10+ years ago

Today

SQL

NoSQL

Today

CQL

N1QL

SQL

NoSQL

Cypher

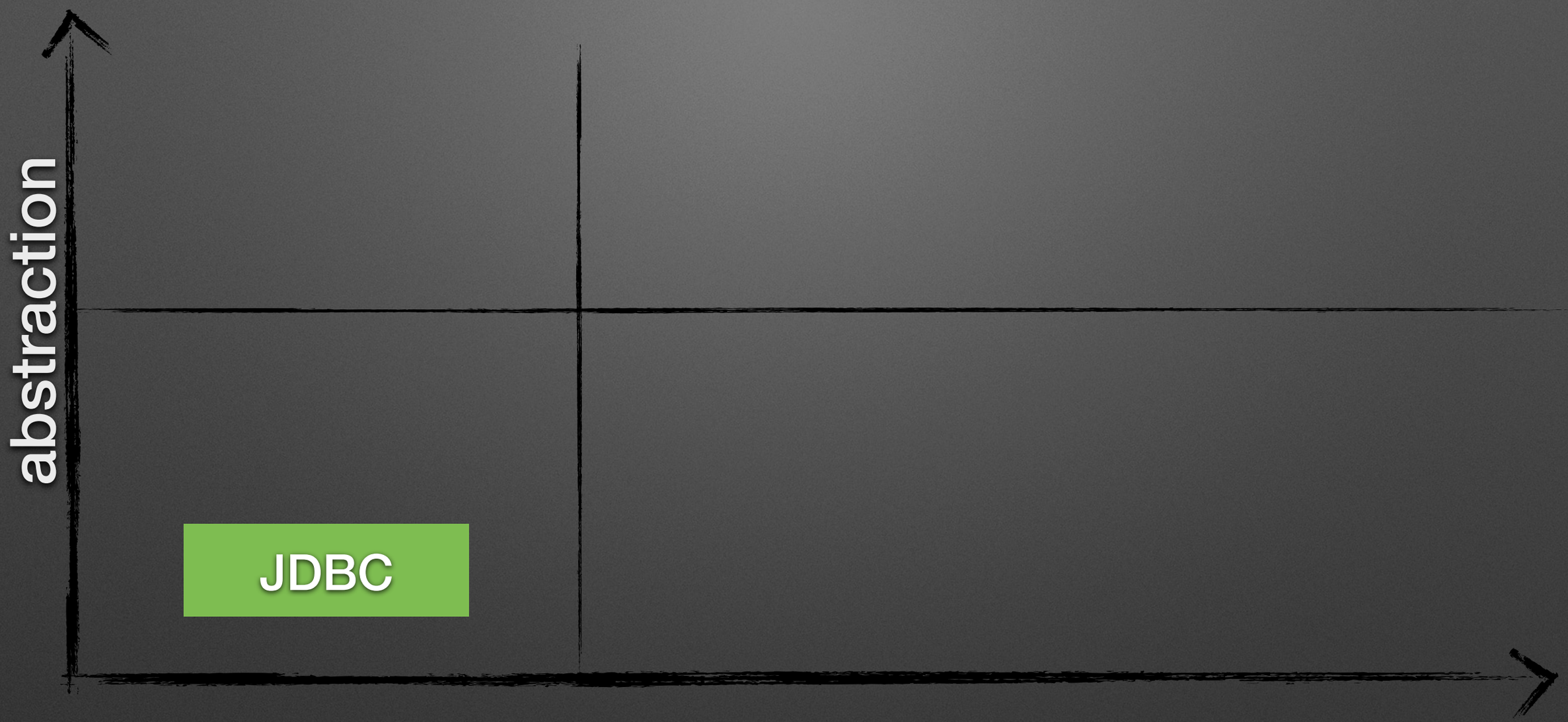
MongoDB CLI

OQL

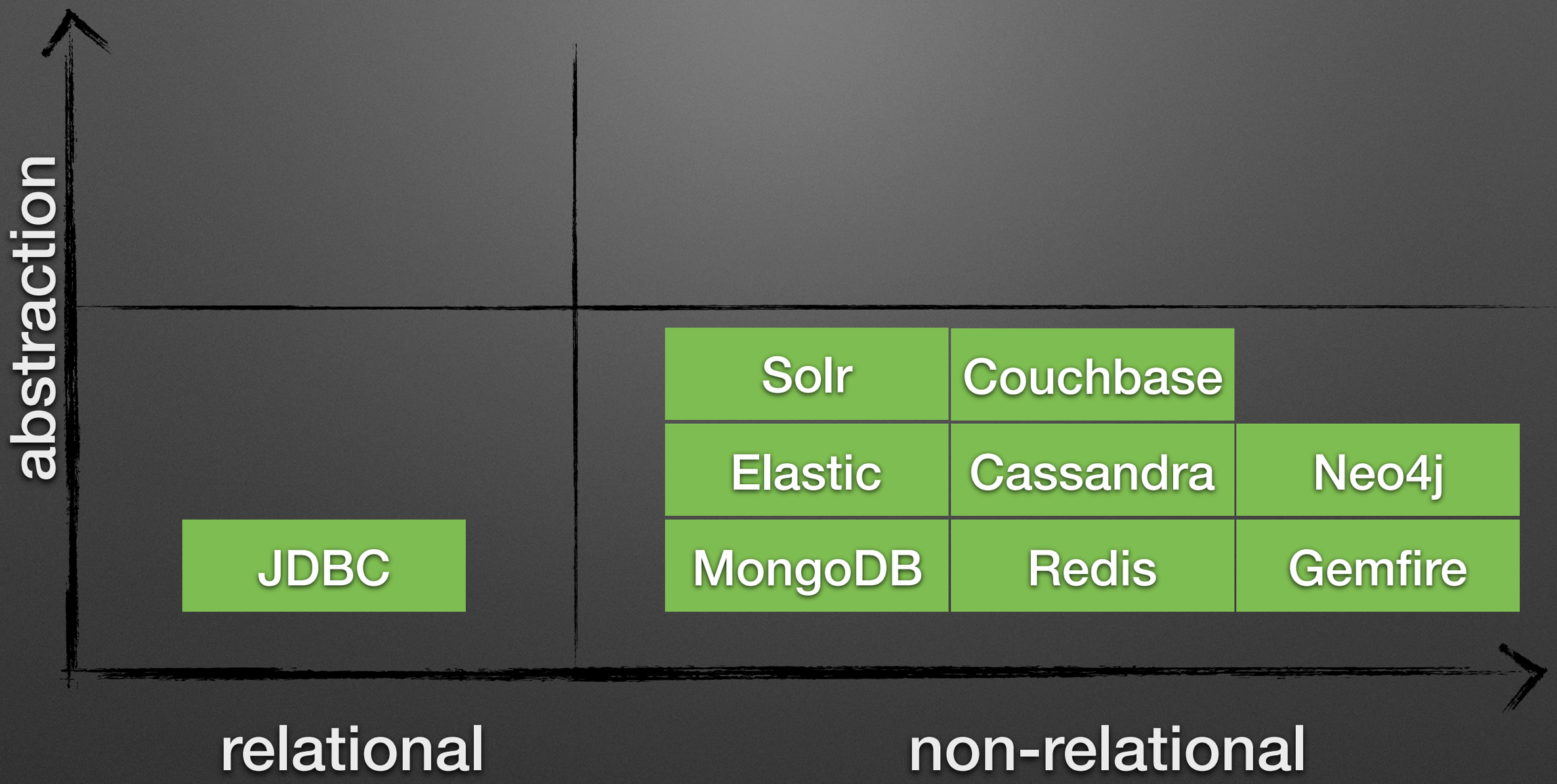
HQL

EJBQL

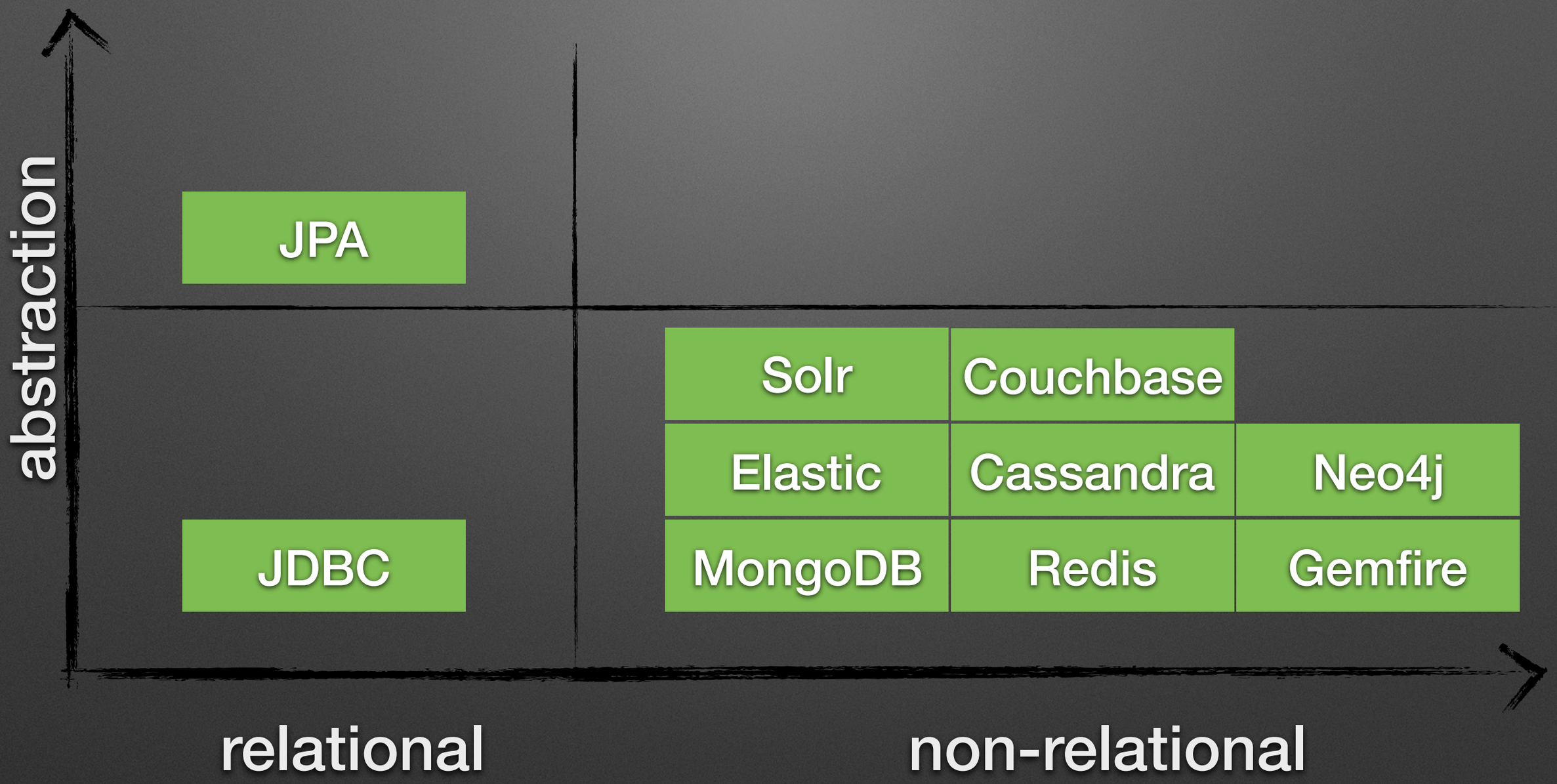
Spring Data provides...



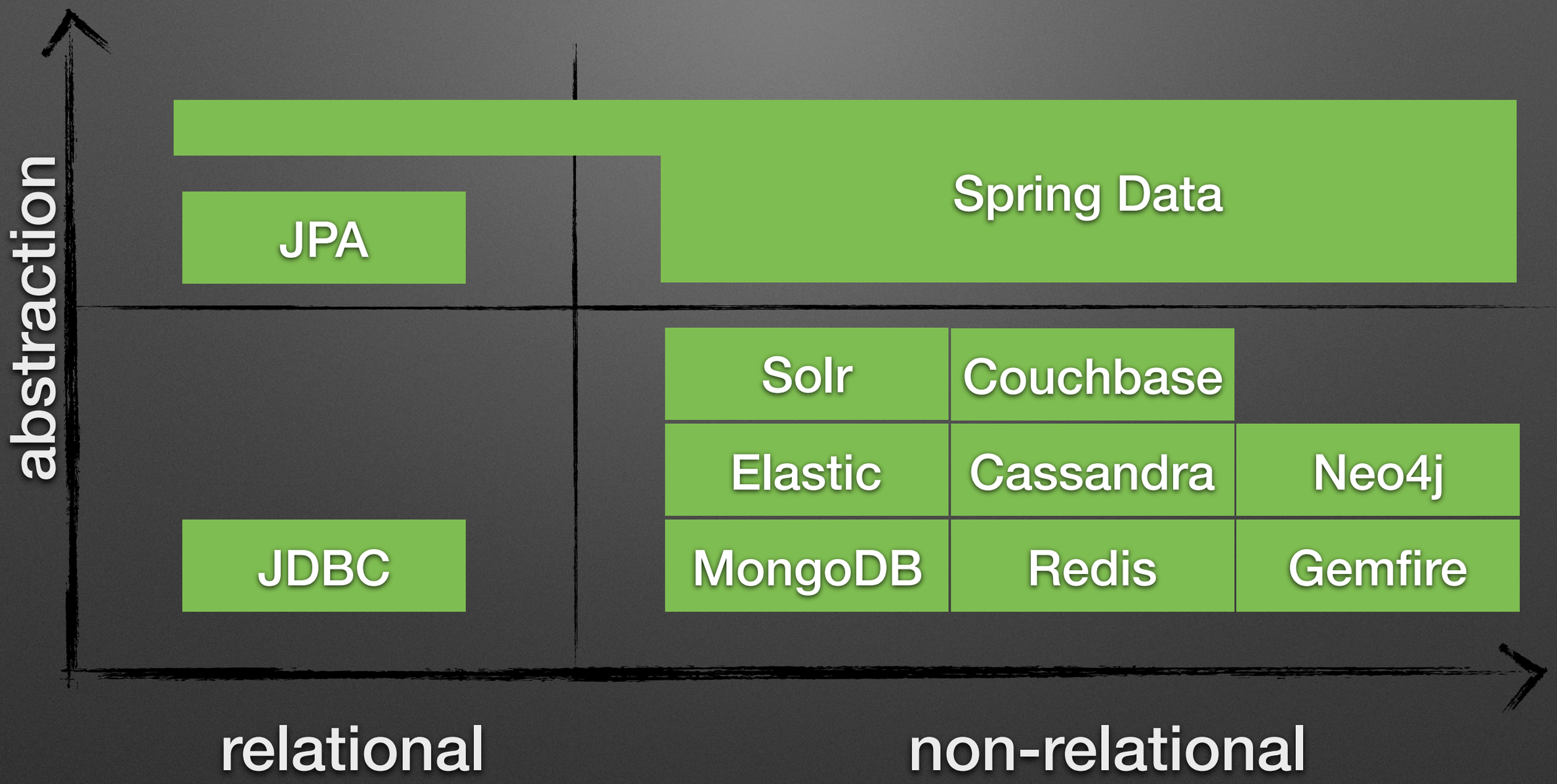
Spring Data provides...



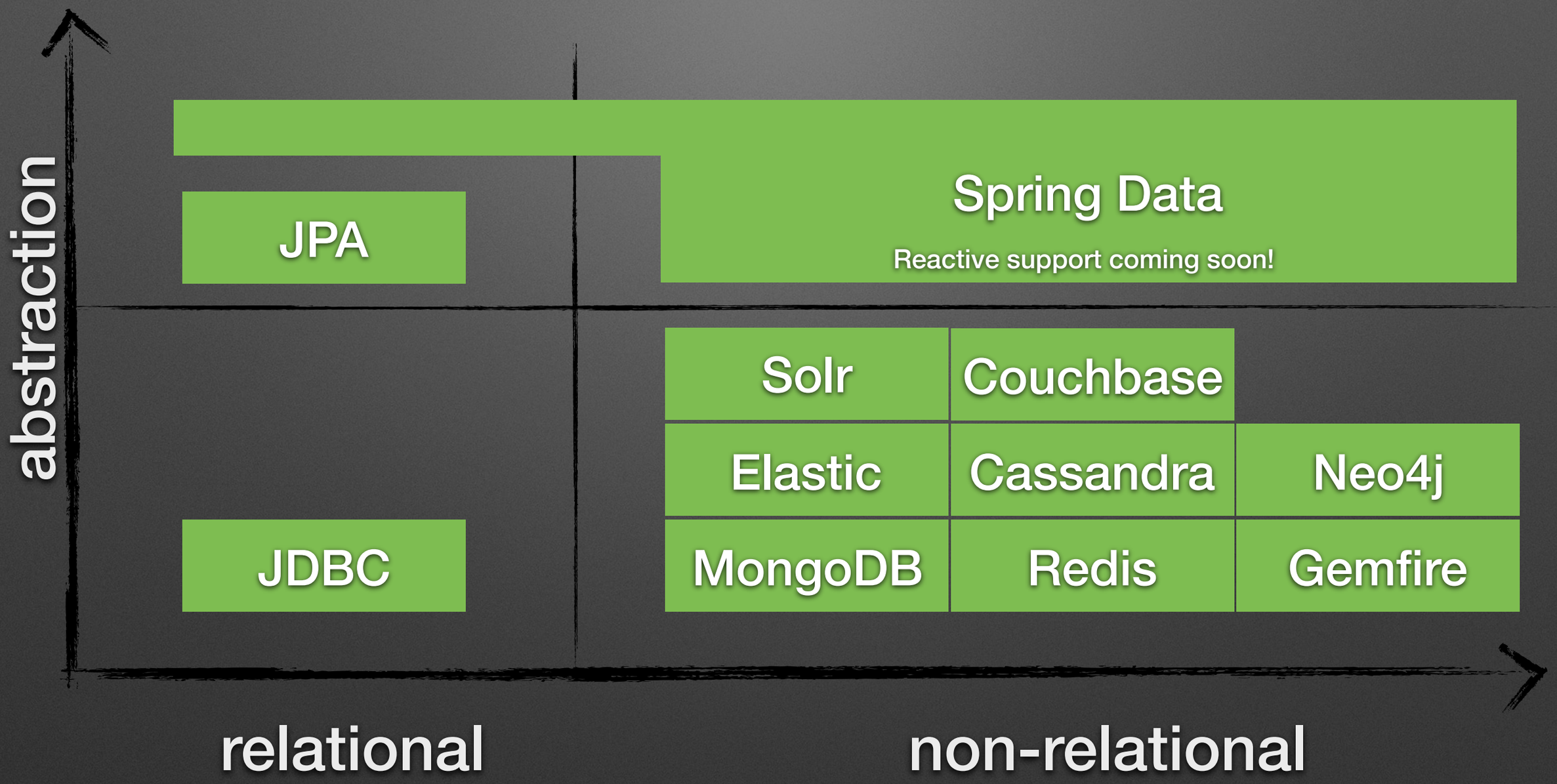
Spring Data provides...



Spring Data provides...



Spring Data provides...



Dialing it up

```
interface EmployeeRepository extends CrudRepository<Employee, Long> {

    List<Employee> findByLastName(String f)
    List<Employee> findByFirstNameAndLastName(String f, String l)
    List<Employee> findByFirstNameAndManagerName(String f, String m)
    List<Employee> findTop10ByFirstName(…) // or findFirst10ByFirstName
    List<Employee> findDistinctEmployeesByFirstName(...)
    List<Employee> findByFirstNameAndLastNameAllIgnoreCase(...)
    List<Employee> findByFirstNameOrderByLastNameAsc(...)
    List<Employee> findByLastNameIsNull(...)

}
```

Why stop there?

```
interface CoolRepo extends CrudRepository<Employee, Long> {
    Stream<Employee> findByLastname(String lastname)
    @Async Future<Employee> findByLastname(…)
    @Async CompletableFuture<Employee> findByLastname(…)
    @Async ListenableFuture<Employee> findByLastname(…)

    Page<Employee> findAll(Pageable p)
    Page<Employee> findByFirstName(String f, Pageable p)
    List<Employee> findAll(Sort s)
    Page<Employee> findByFirstName(String f, Pageable p, Sort s)

}
```

Why stop there?

```
interface CoolRepo extends CrudRepository<Employee, Long> {
    Stream<Employee> findByLastname(String lastname)
    @Async Future<Employee> findByLastname(…)
    @Async CompletableFuture<Employee> findByLastname(…)
    @Async ListenableFuture<Employee> findByLastname(…)

    Page<Employee> findAll(Pageable p)
    Page<Employee> findByFirstName(String f, Pageable p)
    List<Employee> findAll(Sort s)
    Page<Employee> findByFirstName(String f, Pageable p, Sort s)

}
```

Use *@Query* to write custom queries

Spring Data

Demo

DEVNEXUS™

spring by Pivotal

Links

- http://projects.spring.io/spring-data
- https://github.com/spring-projects/spring-data-examples
- https://spring.io/guides/tutorials/react-and-spring-data-rest/
- https://spring.io/guides?filter=spring%20data

Introduction to Spring Data

Greg Turnquist
@gregturn
github.com/gregturn

DEVNEXUS™

spring by Pivotal