PROFILING AND OPTIMIZATION

We performed 3 profilings and the corresponding optimizations, all using glowroot.

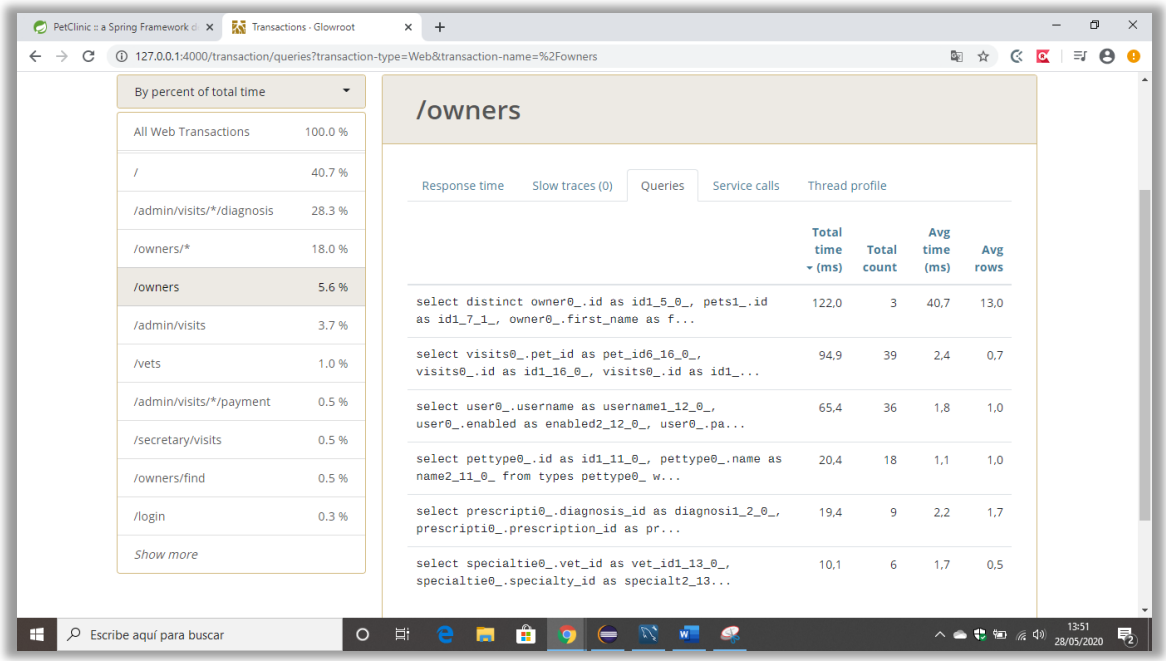
PROFILING 1

Description:

A N+1 Query problem has been detected when, by logging in as admin, a search is made for all the owners that exist in our system.

When the view of all owners (/owners) is loaded, all the owners and pets of each one appear. It has been detected that, in that view, for each pet that appears the visits of each one are loaded.

In our system we have 13 pets associated with different owners, so for each pet that we have included in our database, 13 queries are made that return the visits of each pet has.



It can be seen with:

When that view is loaded once:

Response time | Slow traces (0) | Queries | Service calls | Thread profile

	Total time (ms)	Total count	Avg time (ms)	Avg rows
`select distinct owner0_.id as id1_5_0_, pets1_.id as id1_7_1_, owner0_.first_name as ...`	117,8	1	117,8	13,0
`select visits0_.pet_id as pet_id6_16_0_, visits0_.id as id1_16_0_, visits0_.id as id1...`	29,1	13	2,2	0,7

When that view is loaded twice :

Response time | Slow traces (0) | Queries | Service calls | Thread profile

	Total time (ms)	Total count	Avg time (ms)	Avg rows
`select distinct owner0_.id as id1_5_0_, pets1_.id as id1_7_1_, owner0_.first_name as f...`	119,7	2	59,9	13,0
`select visits0_.pet_id as pet_id6_16_0_, visits0_.id as id1_16_0_, visits0_.id as id1_...`	68,7	26	2,6	0,7

When that view is loaded three times:

Response time | Slow traces (0) | Queries | Service calls | Thread profile

	Total time (ms)	Total count	Avg time (ms)	Avg rows
`select distinct owner0_.id as id1_5_0_, pets1_.id as id1_7_1_, owner0_.first_name as f...`	122,0	3	40,7	13,0
`select visits0_.pet_id as pet_id6_16_0_, visits0_.id as id1_16_0_, visits0_.id as id1_...`	94,9	39	2,4	0,7

Problem:

It has been detected in the model that the relationship of pet with visits was of type **.EAGER**, which means that whenever a pet is loaded, it's visits are loaded.

OwnerService.java | Owner.java | Pet.java

```java
33 public class Pet extends NamedEntity {
34
35     // ATTRIBUTES -------------------------------------------------------------
36
37     @Column(name = "birth_date")
38     @DateTimeFormat(pattern = "yyyy/MM/dd")
39     private LocalDate   birthDate;
40
41     // RELATIONSHIPS ----------------------------------------------------------
42
43     @ManyToOne
44     @JoinColumn(name = "type_id")
45     private PetType     type;
46
47     @ManyToOne
48     @JoinColumn(name = "owner_id")
49     private Owner       owner;
50
51     @OneToMany(cascade = CascadeType.ALL, mappedBy = "pet", fetch = FetchType.EAGER)
52     private Set<Visit>  visits;
53
```

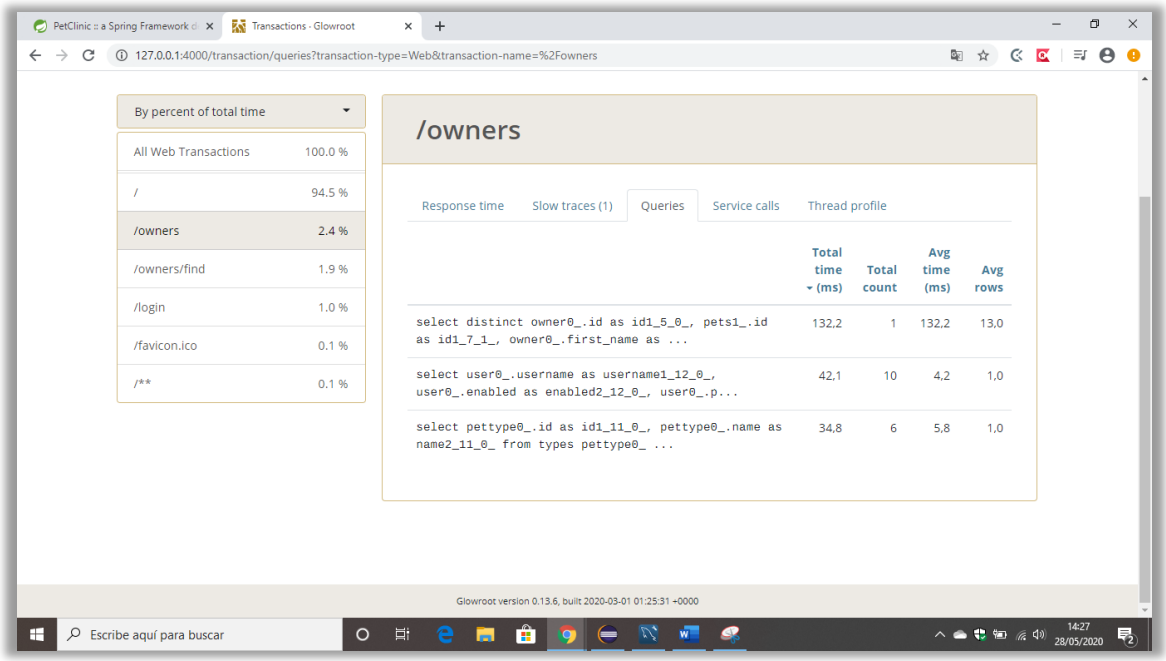
Solution:

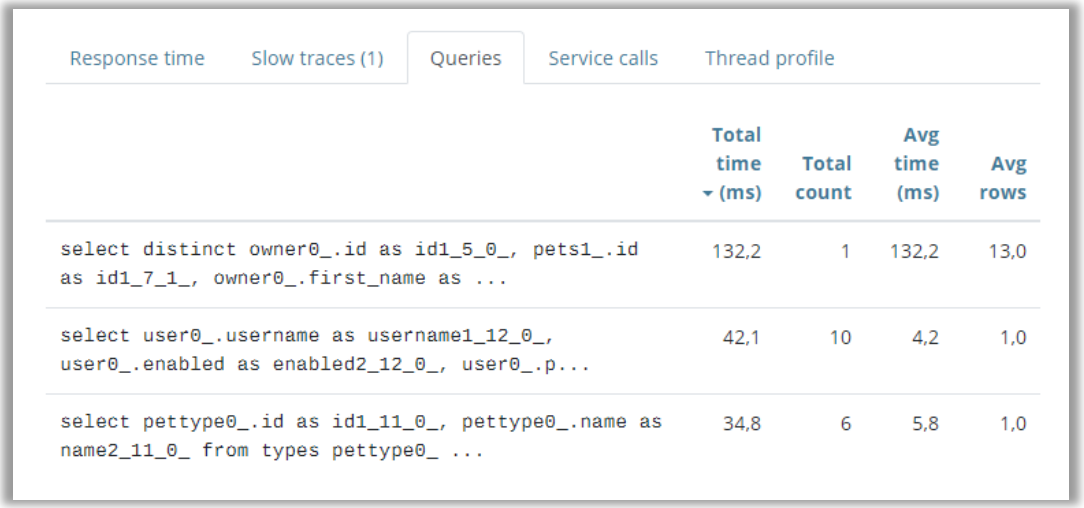
It has been changed and not the relationship pet-visits has been set to type .LAZY so that ii only loads when necessary (since visits is something we don't need in the /owners view we are talking about).

OwnerService.java | Owner.java | Pet.java

```java
33 public class Pet extends NamedEntity {
34
35     // ATTRIBUTES -------------------------------------------------------------
36
37     @Column(name = "birth_date")
38     @DateTimeFormat(pattern = "yyyy/MM/dd")
39     private LocalDate   birthDate;
40
41     // RELATIONSHIPS ----------------------------------------------------------
42
43     @ManyToOne
44     @JoinColumn(name = "type_id")
45     private PetType     type;
46
47     @ManyToOne
48     @JoinColumn(name = "owner_id")
49     private Owner       owner;
50
51     @OneToMany(cascade = CascadeType.ALL, mappedBy = "pet", fetch = FetchType.LAZY)
52     private Set<Visit>  visits;
53
```

In this way, now in Glowroot you can see that those N Querys that were made for each pet in our database have disappeared.





PROFILING 2

Situation before:

When accessing the view dp2.com/vet/visits/8 as a vet, 7 queries are made to the database, which we consider to be too many:

All Web Transactions

Response time | Slow traces (0) | Queries | Service calls | Thread profile

	Total time (ms)	Total count	Avg time (ms)	Avg rows
select user0_.username as username1_12_0_, user0_.enabled as enabled2_12_0_, user0_.pass...	2,1	6	0.35	1,0
select specialtie0_.vet_id as vet_id1_13_0_, specialtie0_.specialty_id as specialt2_13_0...	0.97	6	0.16	0,8
select vet0_.id as id1_14_, vet0_.first_name as first_na2_14_, vet0_.last_name as last_n...	0.47	1	0.47	6,0
select visittype0_.id as id1_15_, visittype0_.name as name2_15_, visittype0_.duration as...	0.20	1	0.20	3,0
select pet0_.id as id1_7_0_, pet0_.name as name2_7_0_, pet0_.birth_date as birth_da3_7_0...	0.16	1	0.16	1,0
select visit0_.id as id1_16_, visit0_.description as descript2_16_, visit0_.diagnosis_id...	0.15	1	0.15	1,0
select payment0_.id as id1_6_0_, payment0_.creditcard_id as creditca5_6_0_, payment0_.fi...	0.15	1	0.15	1,0

Solution:

We added a cache for findVisitById.

First, we added the cache configuration as explained in the video on EV:

```
package org.group2.petclinic.configuration;

import org.springframework.cache.annotation.EnableCaching;

@Configuration
@EnableCaching
public class CacheConfiguration {

}
```

We added a cache logger:

```java
package org.group2.petclinic.configuration;

import org.ehcache.event.CacheEvent;

public class CacheLogger implements CacheEventListener<Object, Object> {
    private final Logger LOG = LoggerFactory.getLogger(CacheLogger.class);
    @Override
    public void onEvent(CacheEvent<?, ?> cacheEvent) {
      LOG.info("Key: {} | EventType: {} | Old value: {} | New value: {}",
                cacheEvent.getKey(), cacheEvent.getType(), cacheEvent.getOldValue(),
                cacheEvent.getNewValue());
    }
  }
```

We added the ehcache3 template:

```xml
<config
            xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
            xmlns='http://www.ehcache.org/v3'
            xsi:schemaLocation="
                http://www.ehcache.org/v3
                http://www.ehcache.org/schema/ehcache-core-3.7.xsd">

        <!-- Persistent cache directory -->
        <!--<persistence directory="spring-boot-ehcache/cache" />-->

        <!-- Default cache template -->
        <cache-template name="default">
            <expiry>
                <ttl unit="seconds">120</ttl>
            </expiry>
            <listeners>
                <listener>
                    <class>org.group2.petclinic.configuration.CacheLogger</class>
                    <event-firing-mode>ASYNCHRONOUS</event-firing-mode>
                    <event-ordering-mode>UNORDERED</event-ordering-mode>
                    <events-to-fire-on>CREATED</events-to-fire-on>
                    <events-to-fire-on>EXPIRED</events-to-fire-on>
                    <events-to-fire-on>EVICTED</events-to-fire-on>
                </listener>
            </listeners>
            <resources>
                <heap>1000</heap>
            </resources>
        </cache-template>

        <cache alias="visitById" uses-template="default">
            <key-type>java.lang.Integer</key-type>
            <value-type>org.group2.petclinic.model.Visit</value-type>
        </cache>

         <cache alias="ownerById" uses-template="default">
            <key-type>java.lang.Integer</key-type>
            <value-type>org.group2.petclinic.model.Owner</value-type>
        </cache>

    </config>
```

We added the necessary annotations:

```java
// FIND VISIT -------------------------------------------------------------

@Transactional(readOnly = true)
@Cacheable("visitById")
public Visit findVisitById(int id) throws DataAccessException {
    return visitRepository.findById(id);
}

// SAVE VISITS ------------------------------------------------------------

@Transactional
@CacheEvict(cacheNames="visitById", allEntries=true)
public void saveVisit(final Visit visit) throws DataAccessException {
    this.visitRepository.save(visit);
}
```

Situation after:

Now, 4 queries are made to the database when the view dp2.com/vet/visits/8 is loaded, while previously it was 7. With the cache, we were able to avoid 3 queries.

All Web Transactions

Response time | Slow traces (0) | Queries | Service calls | Thread profile

	Total time (ms)	Total count	Avg time (ms)	Avg rows
select specialtie0_.vet_id as vet_id1_13_0_, specialtie0_.specialty_id as specialt2_13_0...	1,1	6	0.18	0,8
select user0_.username as username1_12_0_, user0_.enabled as enabled2_12_0_, user0_.pass...	1,0	6	0.17	1,0
select vet0_.id as id1_14_, vet0_.first_name as first_na2_14_, vet0_.last_name as last_n...	0.37	1	0.37	6,0
select visittype0_.id as id1_15_, visittype0_.name as name2_15_, visittype0_.duration as...	0.093	1	0.093	3,0

PROFILING 3

Situation before:

When accessing the view dp2.com/owners/1 as an admin, 5 queries are made to the database, even though the data could be stored in a cache:

All Web Transactions

Response time | Slow traces (0) | Queries | Service calls | Thread profile

	Total time (ms)	Total count	Avg time (ms)	Avg rows
select owner0_.id as id1_5_0_, pets1_.id as id1_7_1_, owner0_.first_name as first_na2_5...	0.35	1	0.35	1,0
select visits0_.pet_id as pet_id6_16_0_, visits0_.id as id1_16_0_, visits0_.id as id1_1...	0.31	1	0.31	2,0
select pettype0_.id as id1_11_0_, pettype0_.name as name2_11_0_ from types pettype0_ wh...	0.22	1	0.22	1,0
select user0_.username as username1_12_0_, user0_.enabled as enabled2_12_0_, user0_.pas...	0.19	1	0.19	1,0
select specialtie0_.vet_id as vet_id1_13_0_, specialtie0_.specialty_id as specialt2_13_...	0.17	1	0.17	0

Solution:

We added a cache for findOwnerById.

We did not have to add the cache configuration as we already added it during the previous profiling (profiling 2).

We added the necessary annotations:

```
46    @Transactional(readOnly = true)
47    @Cacheable("ownerById")
48    public Owner findOwnerById(final int id) throws DataAccessException {
49        return this.ownerRepository.findById(id);
50    }
```

```
36    @Transactional
37    @CacheEvict(cacheNames="ownerById", allEntries=true)
38    public void saveOwner(final Owner owner) throws DataAccessException {
39        this.ownerRepository.save(owner);
40        this.userService.saveUser(owner.getUser());
41        this.authoritiesService.saveAuthorities(owner.getUser().getUsername(), "owner");
42    }
```

Situation after:

With the cache, no more queries are made to the database.

By percent of total time

All Web Transactions	100.0 %
/owners/*	81.6 %
/**	18.4 %

All Web Transactions

Response time | Slow traces (0) | Queries | Service calls | Thread profile

No data for this time period