8 JANUARY 2014 / RAILS

Lazy loading in rails – Rails Feature

Lazy loading in rails is the amazing feature provided with rails. In console you might have tried to examine how lazy loading in rails actually works. In this tutorial, we will learn about this Rails - Lazy loading feature with examples.

What exactly is Lazy Loading?

As the name suggests the **data is loaded in lazy manner**(Really!)

i.e. Your database is queried only when data from the database is required for some kind of manipulation in code. You will get more of this after you read how-to of lazy loading below.

How lazy loading works:

Whenever you try to get some data from database,

For example, users is the database table that you have. And you are querying database to get users having age less than 20.

Then, you will write code like,

```
result = User.where("age < 20")
```

when above statement is executed, your database is not queries yet(because the resultant data is not required yet).

When you execute following code,

```
records = result.all
```

That, you asked for data (All rows from the required data). This is when actually query happens on database and it returns all rows of the result set are returned to results variable. So, results will be array of ActiveRecord's. This is what lazy loading in rails is!

When database is actually queried (end of lazy loading)?

The database is queried when data is required or asked for, in cases like **all, first, count, each.** These will trigger database load and thus lazy loading will end.

Rails Console and Lazy loading in Rails

This is where you need to be patient. We will take same example as discussed above for examining lazy loading on rails console. We can start rails console by command

```
rails console or <strong>rails c</strong>
```

Getting data from users table having age less than 20. Then you execute command,

```
result = User.where("age < 20")
```

And hit enter. Then you will see that **console has printed array of ActiveRecord's as result**. And thus you might think that query has already been expected on your database and lazy loading hasn't worked quite well. But, this is not what happened. You got this result **because rails console inspects last statement** that it executes which gives the array of ActiveRecord's.

i.e. If you try

```
result = User.where("age < 20"); nil
=> nil
```

Then you will see that rails **console has printed nil** and not array of ActiveRecord's as a result. As the last statement that is executed was **nil**.

There is simple way to **check whether lazy loading is in place** (working correctly) -

Just check the class of result by,

```
result.class
=> ActiveRecord::Relation
```

It is ActiveRecord::Relation which means database is actually not queried. And it is just the way console works.

Conclusion:

We have **learned how lazy loading in rails works** in general and why lazy loading doesn't seem to be working properly in Rails Console. We also seen a way to verify if lazy loading is in place or not.