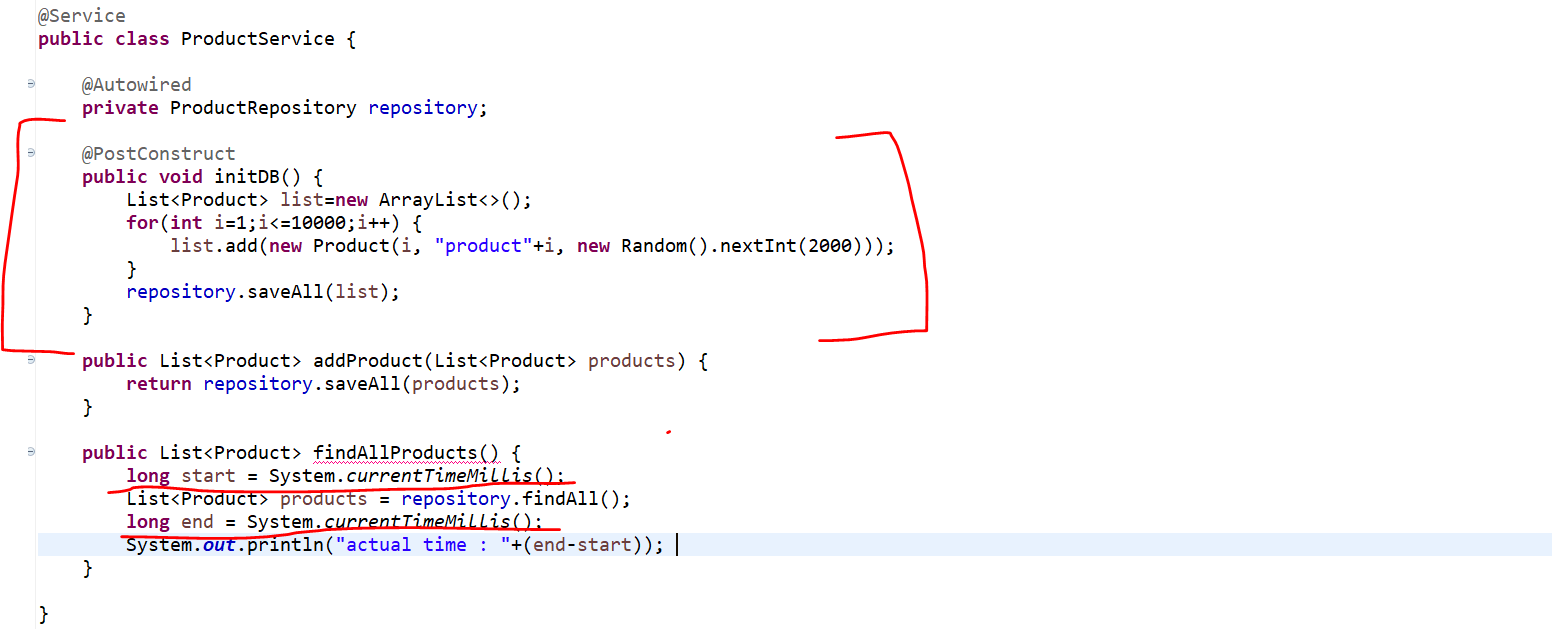
**Spring AOP - Custom annotation to log method execution Time @TrackExecutionTime**

As a developer we need to focus on performance of our code means I should always expect the piece of code which I wrote should give faster response let's say I have a method who is doing some complex business so I want to evaluate how much time this method is taking to execute let's consider this example I have a product service so at the time of application load I am creating 10,000 product object and I am persisting them in database.



And I have a method called Find All products will page list of product object from database. So here I want to track execution time for this find all product method. I want to know how much time this method is taking to complete his execution I can do this, so this is very simple approach which people are following. what we can do we will use **system**.**currentTimeMillisecond**().

So, let's write long start time equal to **system**.**currentTimeMillisecond**() so I'll write **system**.**currentTimeMillisecond**() before and after method call.

And here instead of returning we can write list of product and at end we can return, or we can print the value for this end time minus start time. So, this particular where we are subtracting end time with our start time will give the actual result actual millisecond it will give so that we will get to know how much time this particular method is taking to execute. So, this will work no doubt.

But if same kind of track execution time I want in each and every method or let's say this is in service, I want same track execution time for controller. So how we can do this? Right? Again, we need to copy paste same kind of code across all the method in all the class which is not a feasible solution.

**Why not we will create our own custom annotation who will handle this execution time tracking.**

So, for that we can use Spring AOP around advice and we can create our own custom annotation. That's what we are going to learn in this tutorial. So, we'll create one custom annotation, and we can write on top of method so that that annotation will help us to evaluate execution time for a particular method.



So, let's quickly create a Spring boot project to demonstrate the **Spring AOP Around advice**. So, let me remove this piece of code which we just added which is cross cutting logic and that we want to segregate from our code.

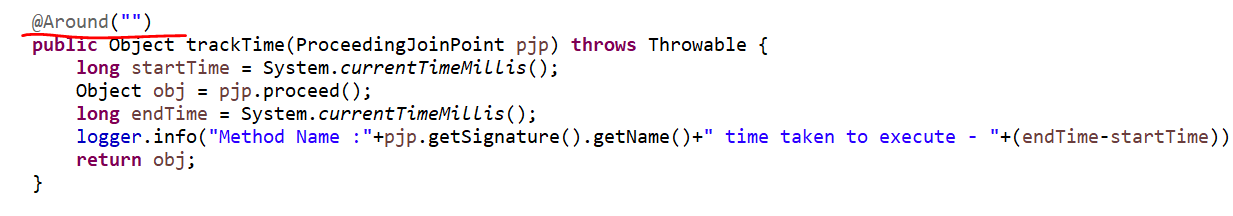
So, let's create a package called **advice**. Let’s create a class called **ExecutionTimeTrackerAdvice**.

As this is the Around advice we are going to use. We need to pass the proceeding join point as an argument. Then we need to get the start time and end time. Once we called the

**ProceedingJoinPoint pjp = new ProceedingJoinPoint();**

**pjp.proceed()**

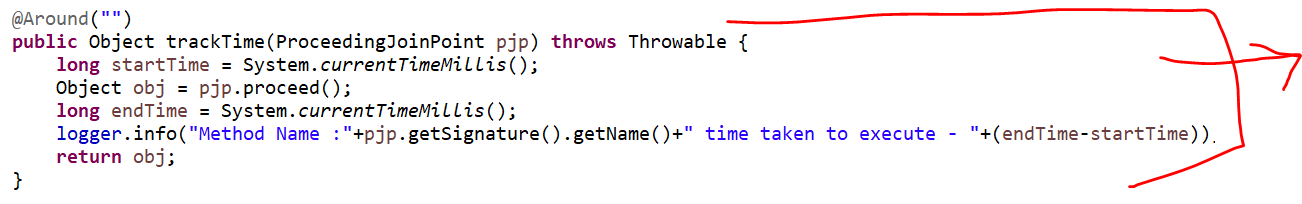
it will behave like an Around advice so, before this method call it will behave like a Before Advice and once, we call this method and after that it will behave as a After Returning Advice that’s what the Around Advice we learnt in our earlier tutorial. Now we get the start time before method call and end time after method call. Now we can easily log the actual execution time.



Here we created a method which will log execution time. What we need to do we need to annotate here @**Around**() advice, and we need to specify the Pointcut here.

But I don’t want to apply this track Time logic in a specific method or in a specific class. I just want to create an annotation who will execute this business logic. So, for that instead of giving pointcut here we will create an annotation and we will give the fully qualified name of the annotation here.

So, whenever you will use that annotation it will execute this piece of code.

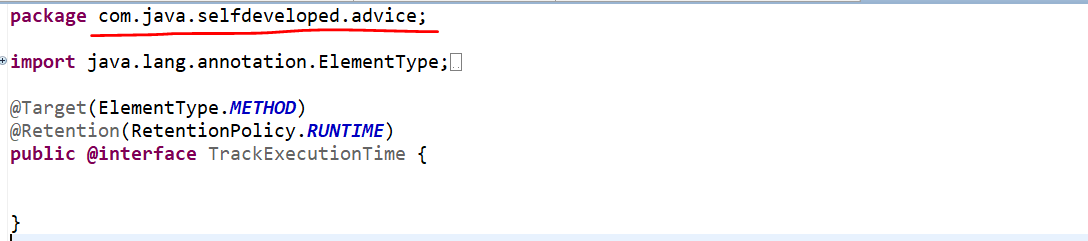


So, this piece of code that annotation will execute. Let me create an annotation first…

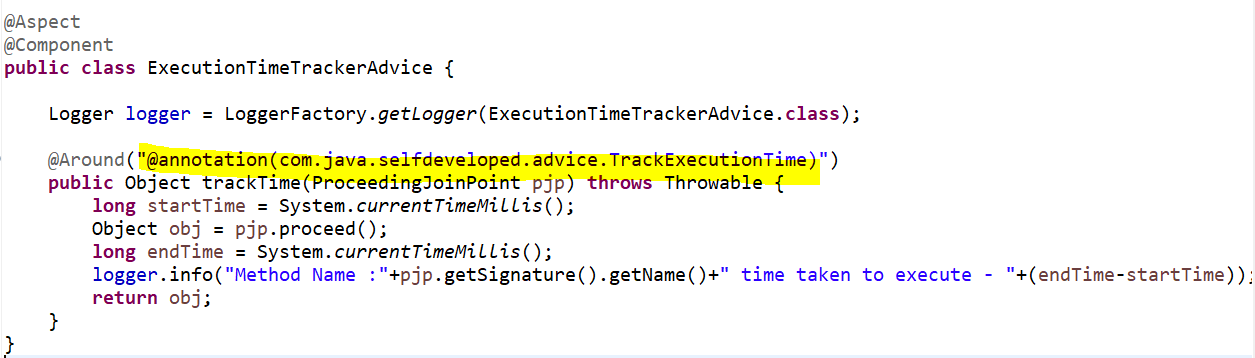
So, in advice package only let me create an annotation called **TrackExecutionTime** and this will be an Annotation.



So, this **TrackExecutionTime** is my custom annotation. U need to annotate with **@Target** to use onwhich level we want to specify this annotation, so we want to apply on method level. **@Retention** this only to use either we want to apply at compile time or run time. So, we want it to be apply on Run-Time.



We created this annotation, and we need to give fully qualified path of this annotation to our Around advice.



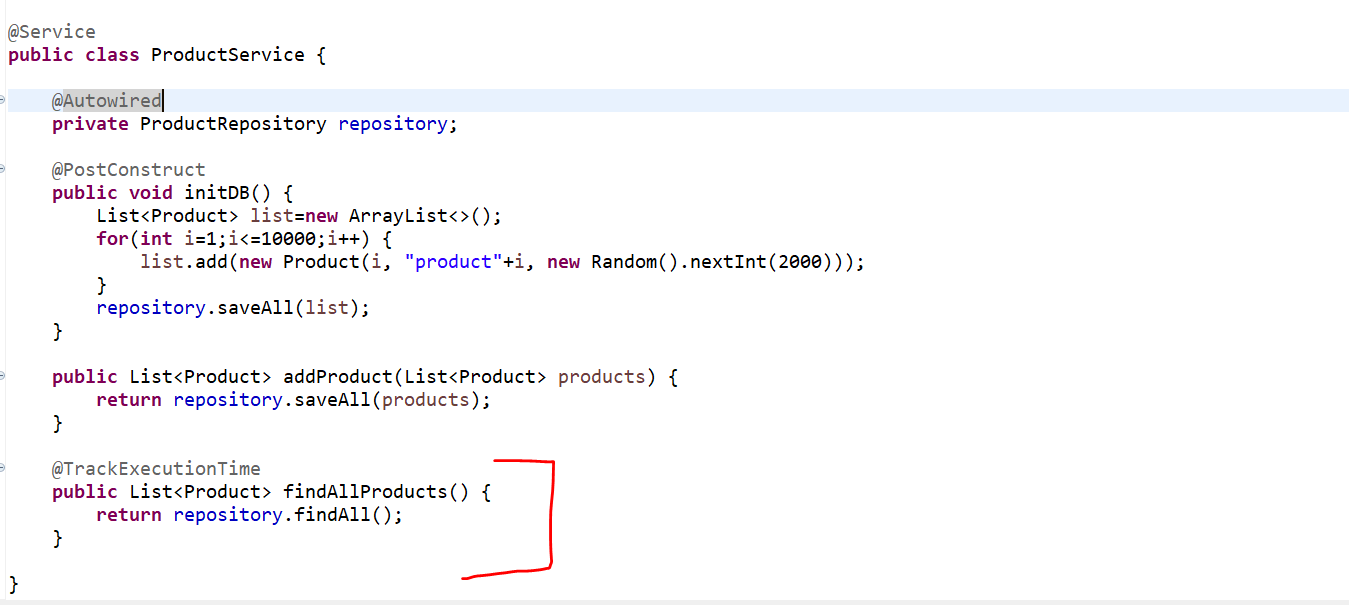
Now wherever I use this **TrackExecutionTime** annotation the request or the control will come to this advice which is our Around advice. And this piece of code will execute. That’s what I didn’t specified the pointcut here I specified the fully qualified annotation here.

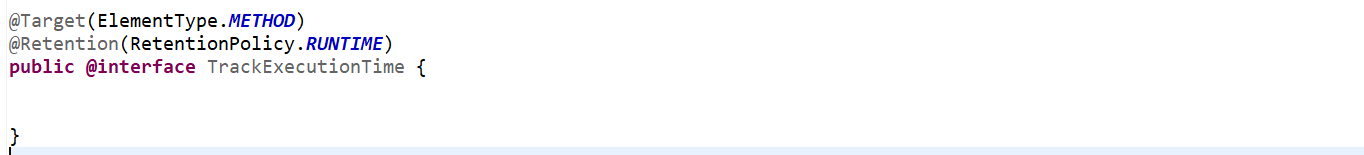
**So, let’s use this annotation in our method level where exactly we want to track our execution time.**

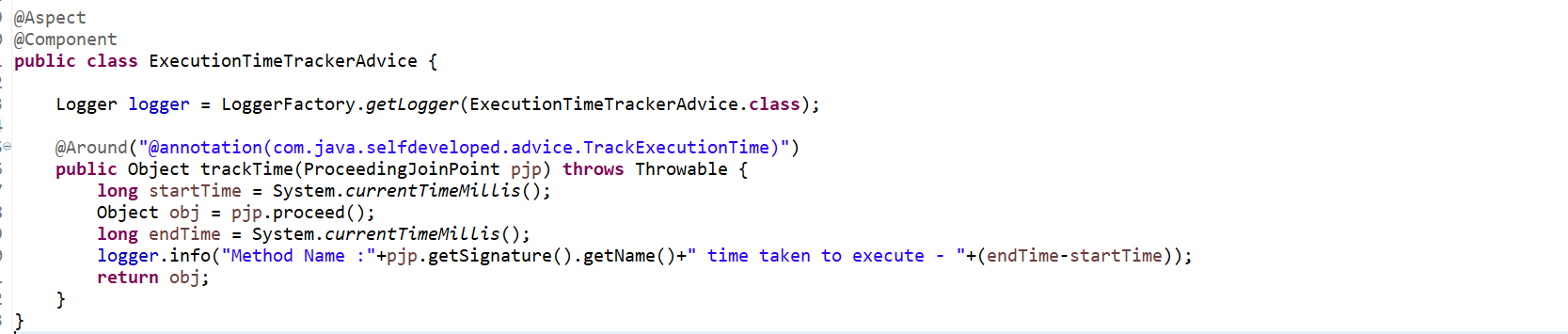


Let’s say I just want to track the execution time for this particular **findAllProducts**() method.

So what we can do we can use our own annotation **TrackExecutionTime** now once the Request will come to findAllProducts() it will go to based on annotation it will go to the advice this **TimeTrackerExecutionAdvice** and then it will execute that piece of code.



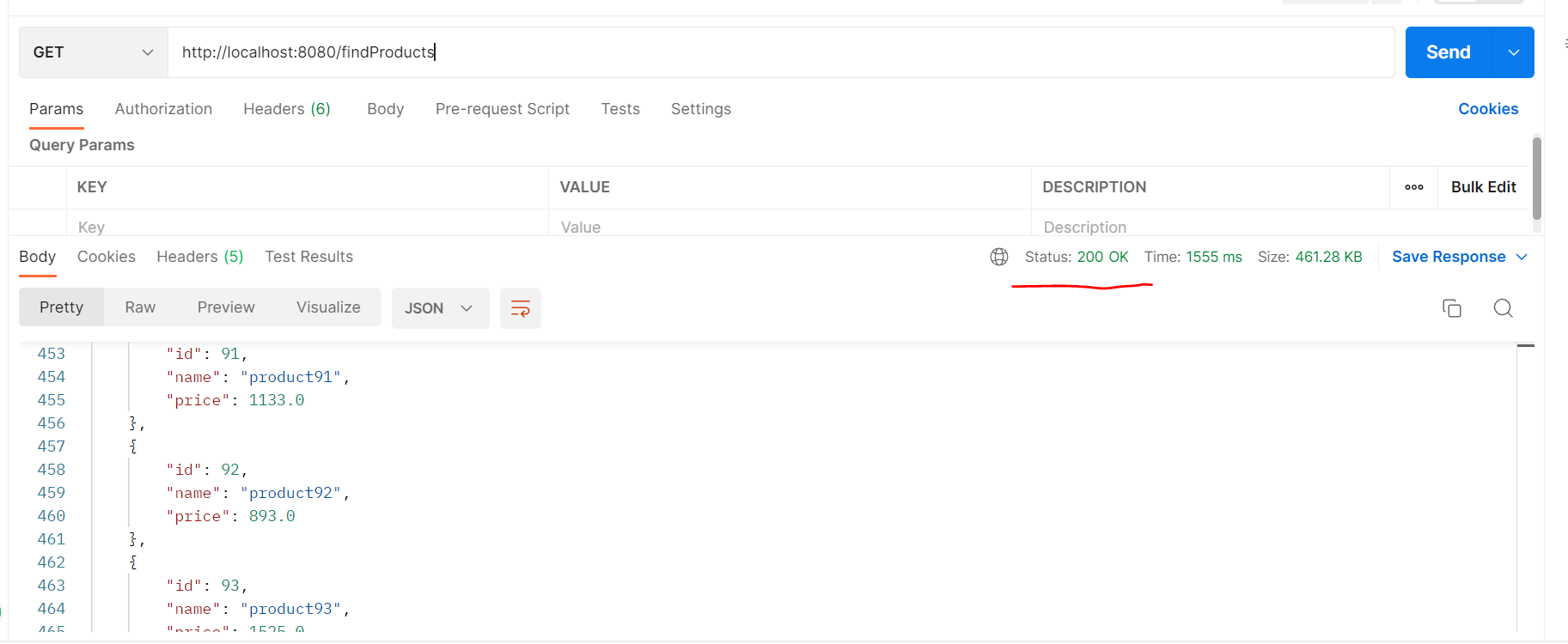




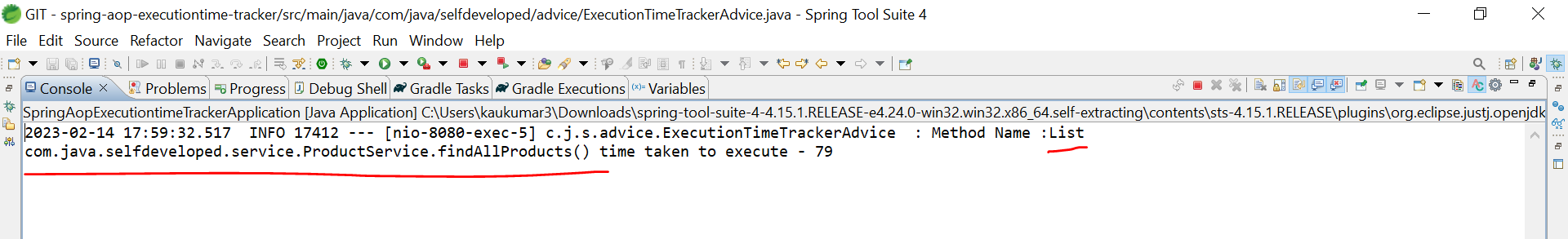
So, now let’s run and verify …

We are hitting one of the endpoints….and we are getting 10000 records.

<http://localhost:8080/findProducts>



Let’s go to console to verify our log statements.

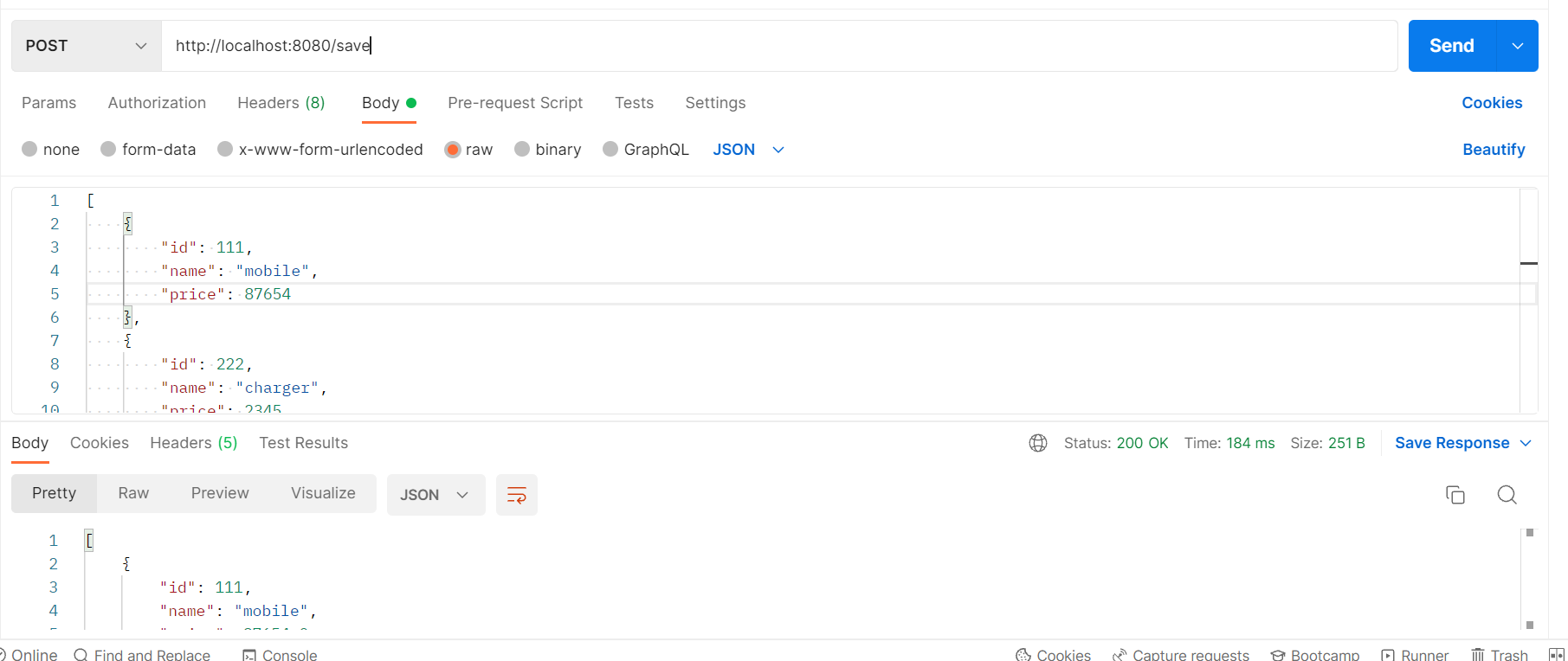


So, it took time 79 miliseconds to execute.

If u want to track for other method as well let’s go to ProductService and let’s say I want to track the execution time for **addProduct()** as well. So, u can do that using the annotation like…



<http://localhost:8080/save>



So, wherever you want to get the execution time as a log you can simply use this annotation.

