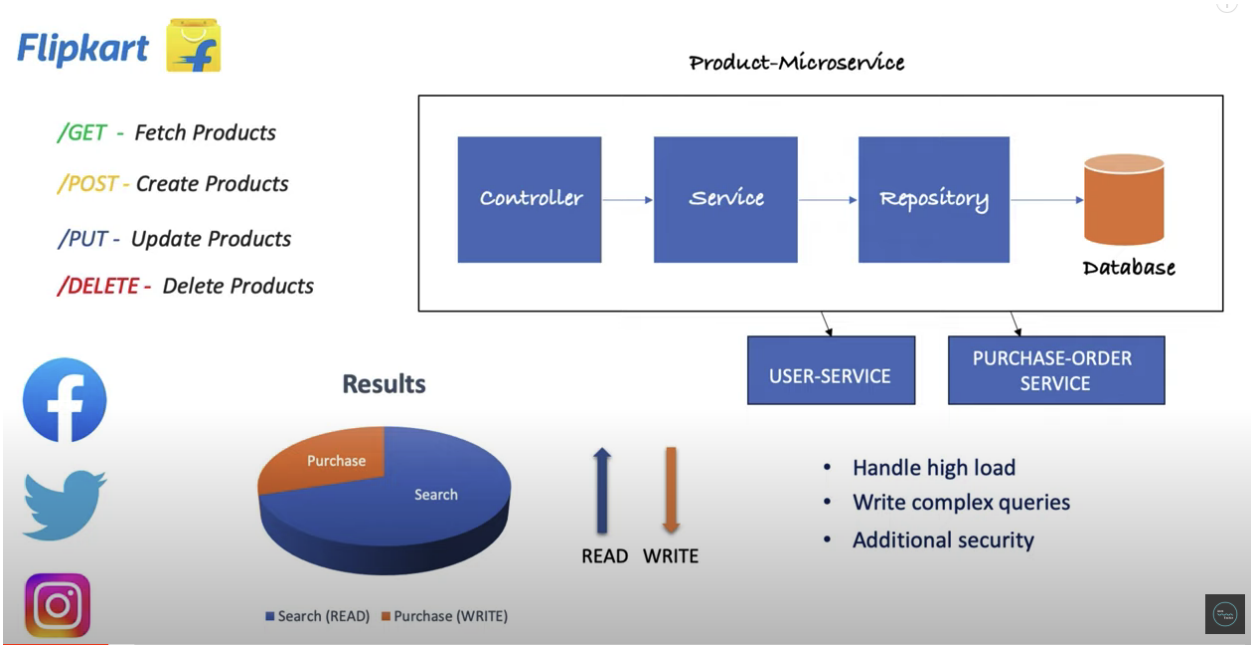
Segregate read and write operations instead of single micro service

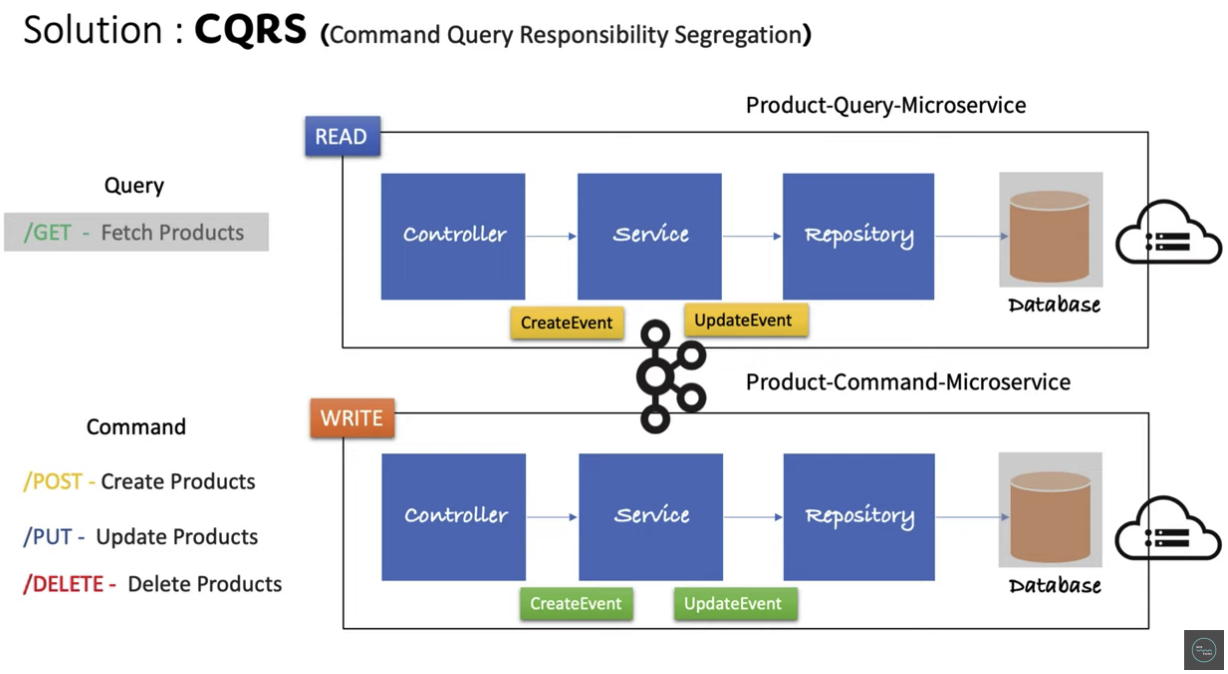


Regular approaches and its problems.

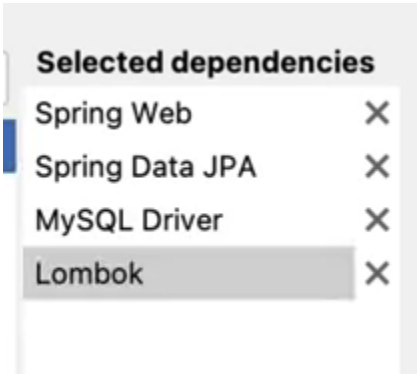
Problem Statement: Because we have read operations more - we can scale the application only for read operation during Peak.

We have to maintain more complex queries if our Micro service integrate with other micro services to fetch orders created by a user and so on.

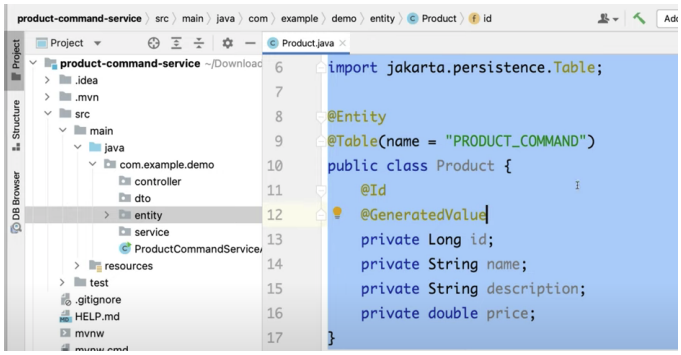
Need to add additional security system.



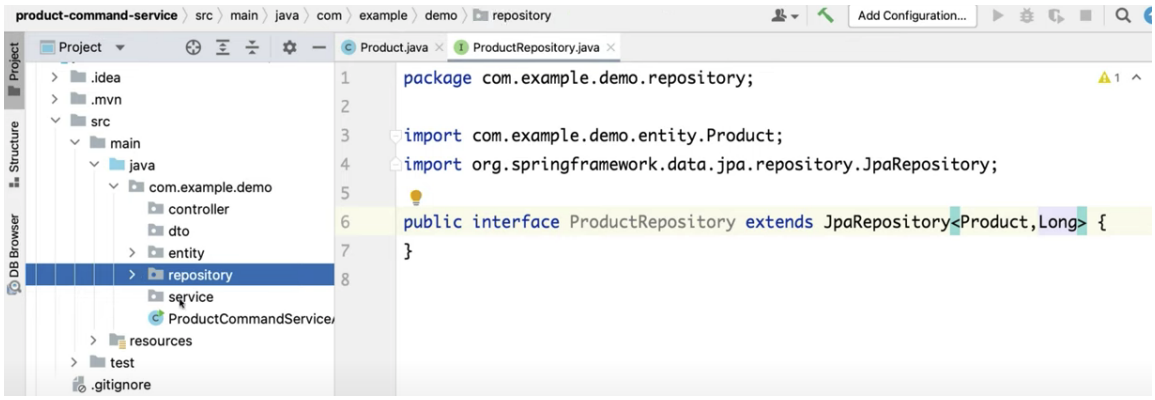
Because our DBs are different whatever create order or update order wont available in the Read application. So bring kafka as solution whatever events happen in the command application publish them to read application.

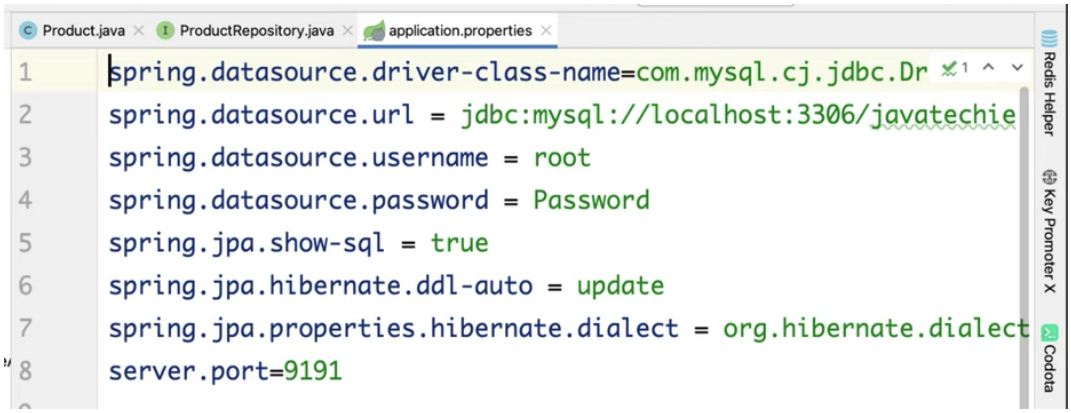


Create a spring boot micro service for command application and read application with above dependencies




```java
@Entity
@Table(name = "PRODUCT_COMMAND")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Product {
```

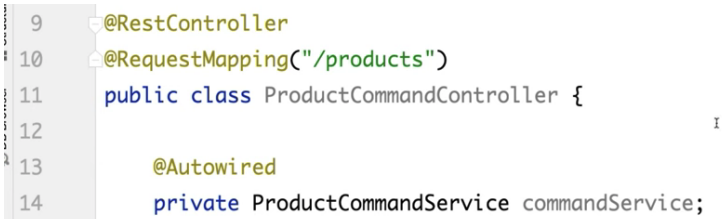




Service class with create and update logic.


```
public Product updateProduct(long id,Product product){
    Product existingProduct = repository.findById(id).get();
    existingProduct.setName(product.getName());
    existingProduct.setPrice(product.getPrice());
    existingProduct.setDescription(product.getDescription());
    return repository.save(existingProduct);
}
```

Controllers for create and update.



Copy entity and repository classes to query service

One advantage is there is no need for an exact match for Entity. Only whatever fields we are interested in - we can define only them.

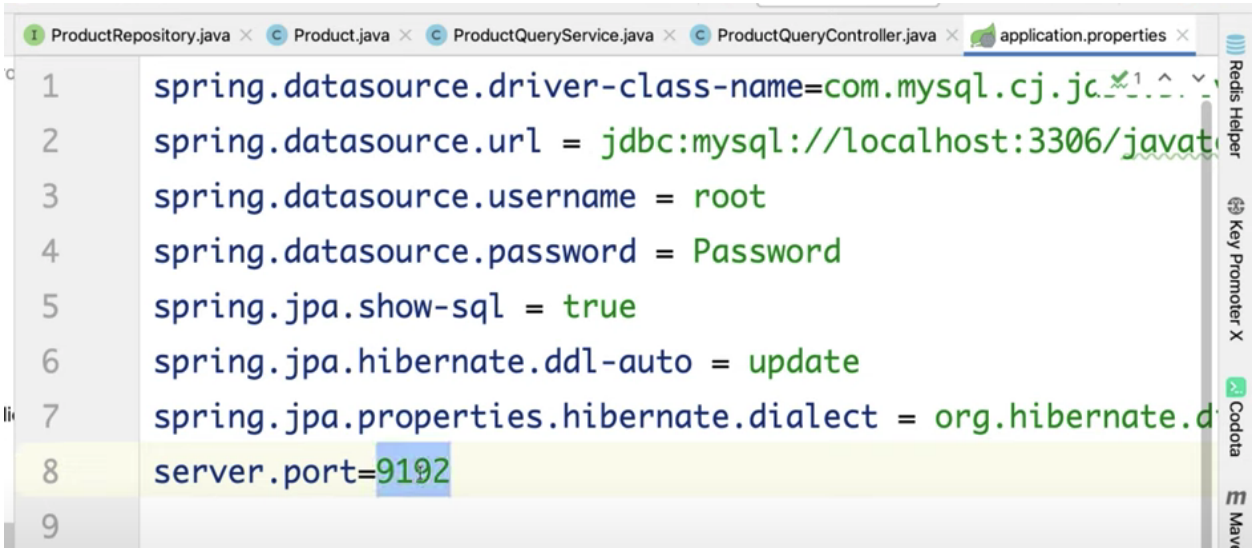
Query service class

```
@Service
public class ProductQueryService {

    @Autowired
    private ProductRepository repository;

    public List<Product> getProducts(){
        return repository.findAll();
    }
}
```

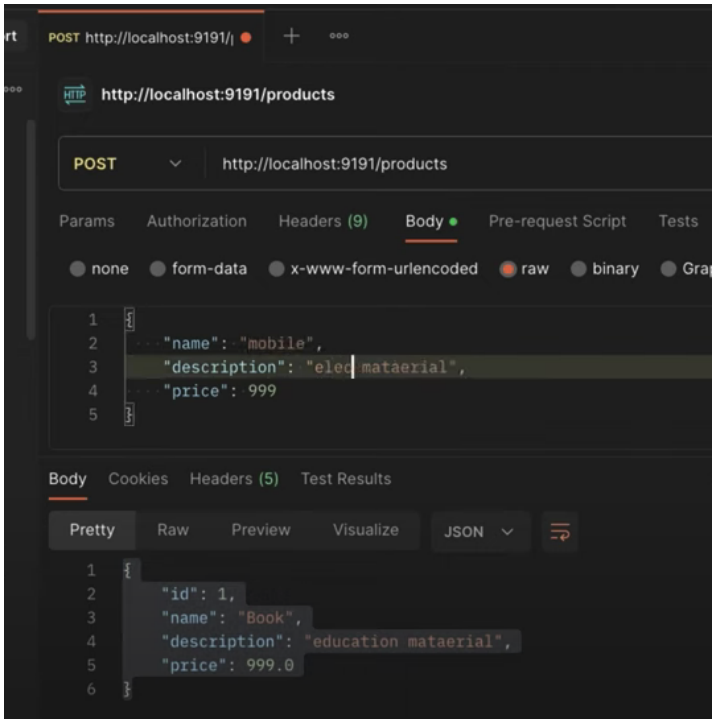
Query controller class

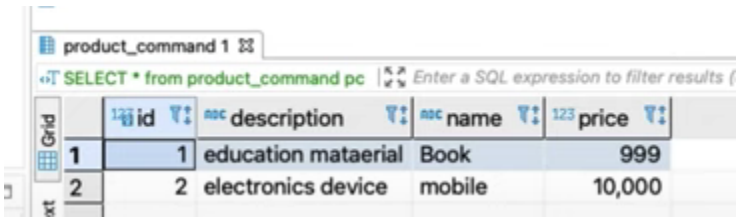
```
@RequestMapping("/products")
@RestController
public class ProductQueryController {

    @Autowired
    private ProductQueryService queryService;

    @GetMapping
    public List<Product> fetchAllProducts(){
        return queryService.getProducts();
    }
}
```

Application properties of query service same as command service except port number

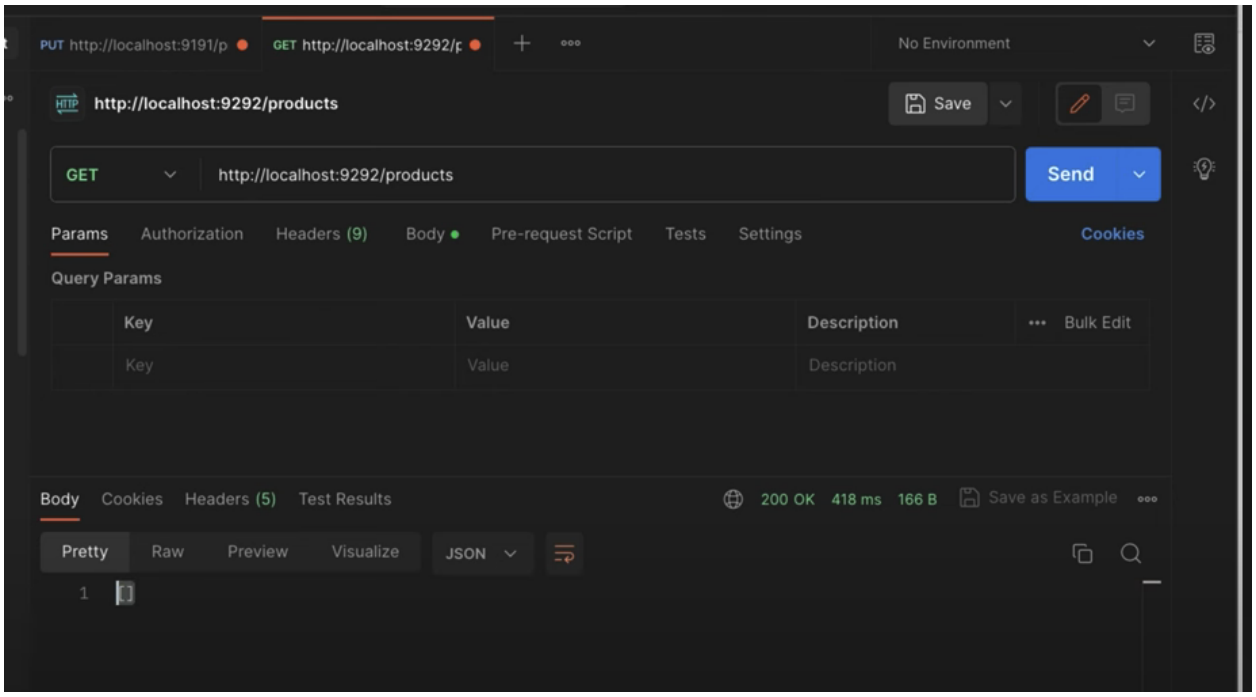


Start the application and add some products

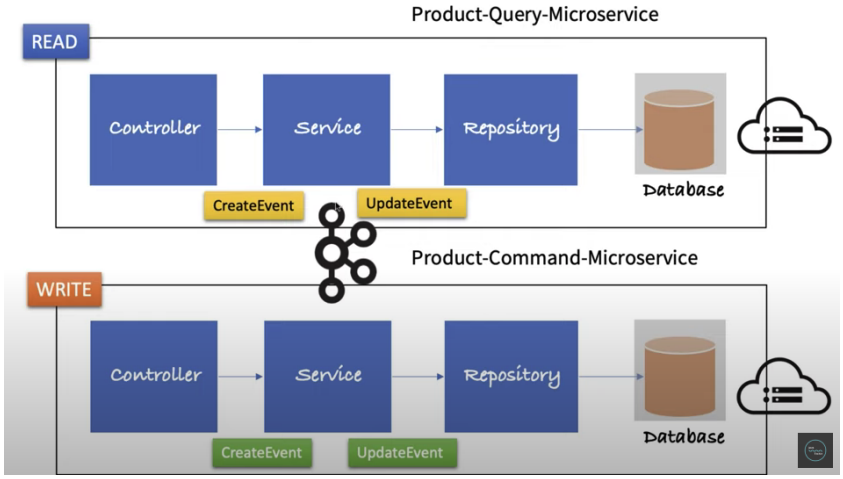




When we query. We are getting empty response.



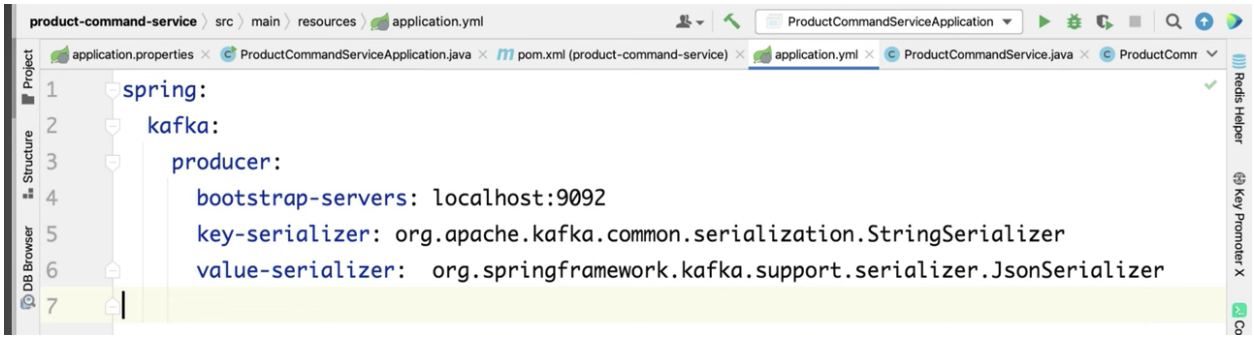
Now Kafka come into picture

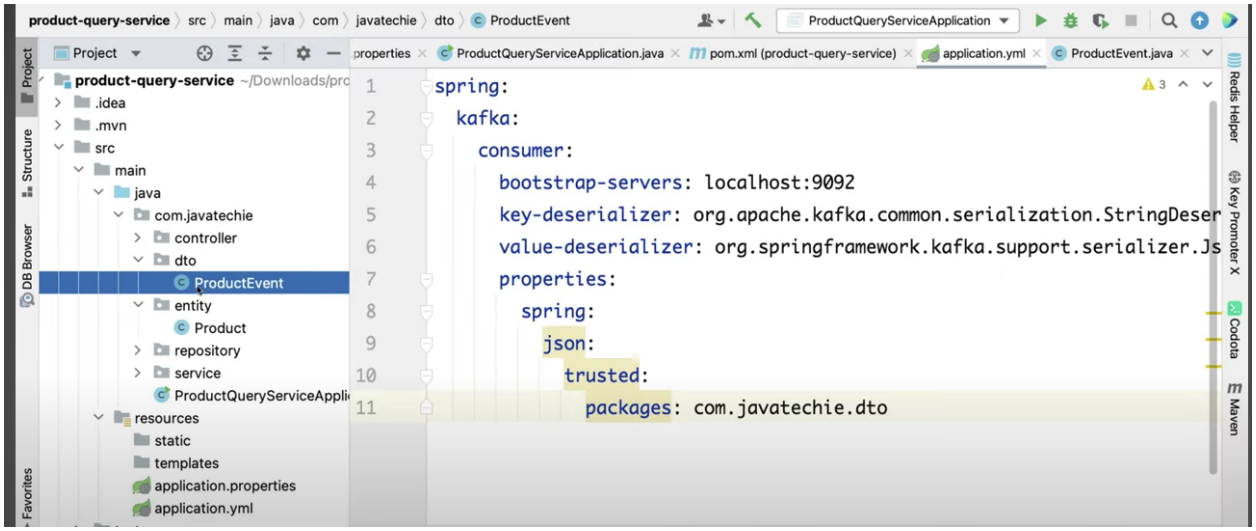


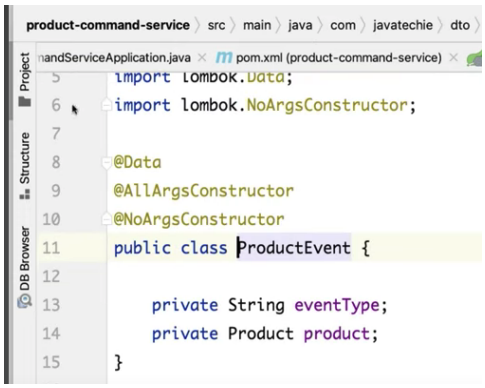
Command service should act as producer and query service should act as listener for each create and update events.

```
</dependency>
<dependency>
    <groupId>org.springframework.kafka</groupId>
    <artifactId>spring-kafka</artifactId>
</dependency>
```

Add this dependency in the both services

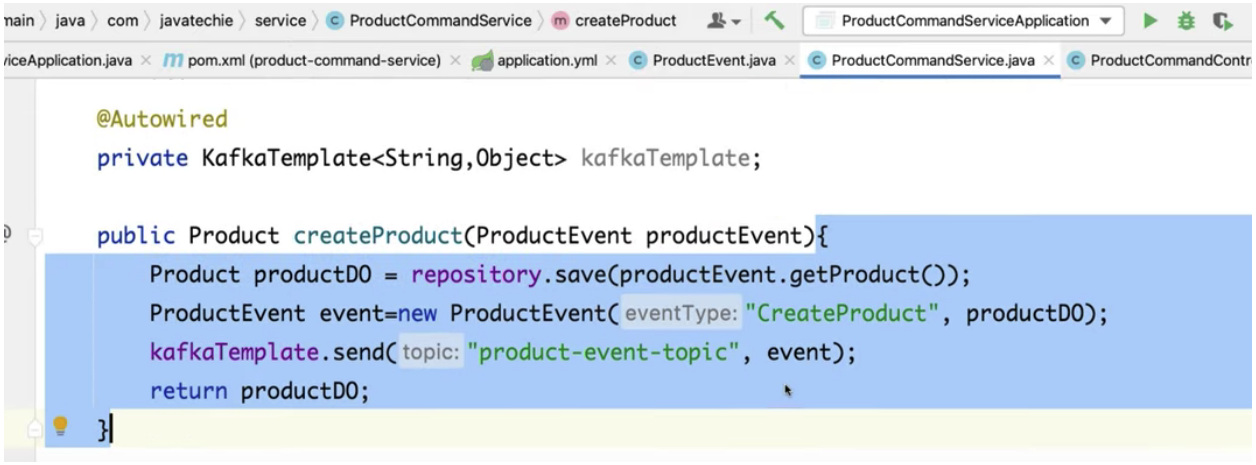






Create a product event class which takes the event type and what is the product.

Update service class with kafka to send product event on creation and update



Update controller class accordingly.

```java
@PutMapping("/{id}")
public Product updateProduct(@PathVariable long id, @RequestBody ProductEvent pro
    return commandService.updateProduct(id, productEvent);
}
```

Now consumer part

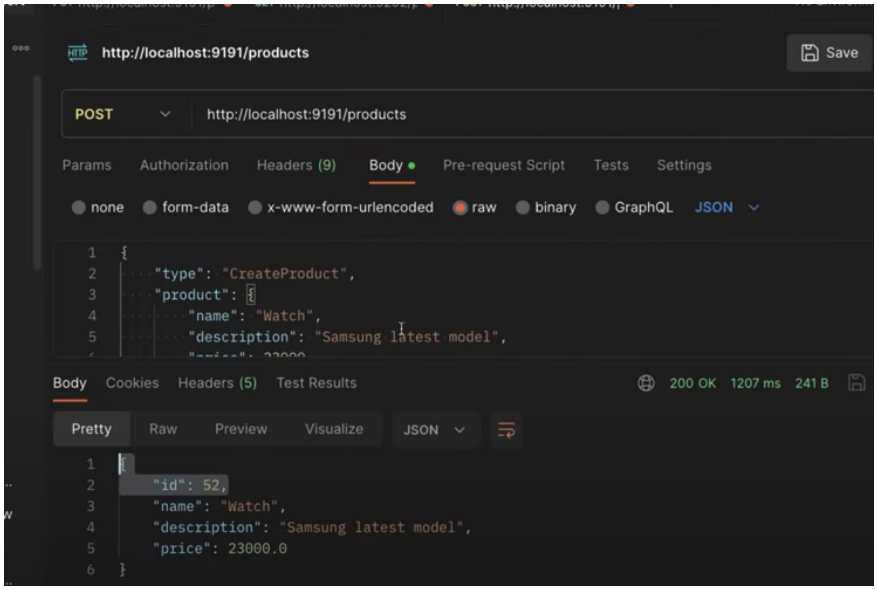
Update query service class with product events

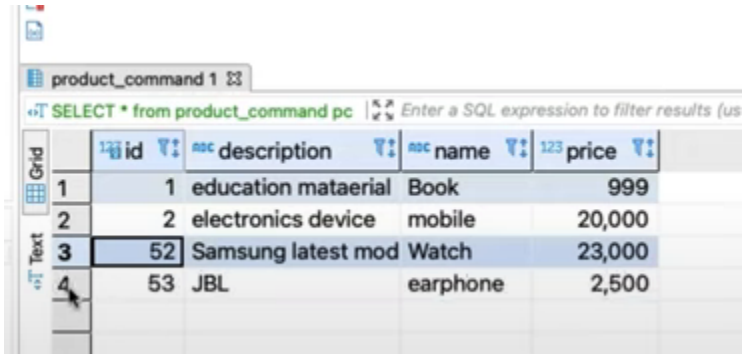
```java
@KafkaListener(topics = "product-event-topic",groupId = "product-event-group")
public void processProductEvents(ProductEvent productEvent) {
```

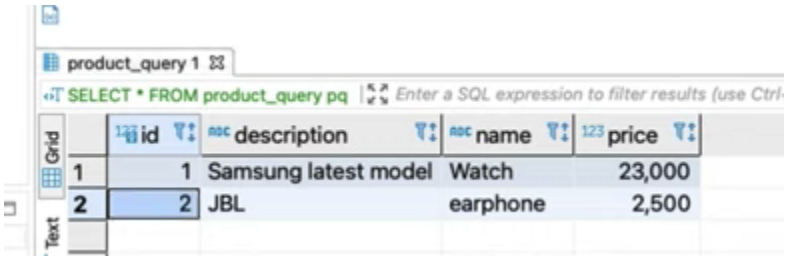
ProductQueryService.java | ProductQueryController.java | application.properties | ProductQueryServiceApplication.java | pom.xml (product-query-service)

```java
public void processProductEvents(ProductEvent productEvent) {
    Product product = productEvent.getProduct();
    if (productEvent.getEventType().equals("CreateProduct")) {
        repository.save(product);
    }
    if (productEvent.getEventType().equals("UpdateProduct")) {
        Product existingProduct = repository.findById(product.getId()).get();
        existingProduct.setName(product.getName());
        existingProduct.setPrice(product.getPrice());
        existingProduct.setDescription(product.getDescription());
        repository.save(existingProduct);
    }
```

Now add some products







Because id is not in sync just delete the records and start test the producer with update and create events.