# Using stripe in spring

Sure, let's walk through an example of integrating the Stripe payment gateway into a Spring Boot application.

1. **Sign Up and Obtain Credentials**: Sign up for a Stripe account and obtain your API keys (publishable key and secret key) from the Stripe Dashboard.

2. **Add Dependency**: Add the Stripe Java library dependency to your `pom.xml` file:

```xml
<dependency>
    <groupId>com.stripe</groupId>
    <artifactId>stripe-java</artifactId>
    <version>20.84.0</version> <!-- Replace with the latest version -->
</dependency>
```

3. **Configure API Keys**: Store your Stripe API keys in the `application.properties` file:

```properties
stripe.api.key=sk_test_your_secret_key
```

4. **Create Payment Service**: Create a Spring service to handle payment processing logic:

```java
import com.stripe.Stripe;
import com.stripe.model.Charge;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;

import java.util.HashMap;
import java.util.Map;

@Service
public class PaymentService {

    @Value("${stripe.api.key}")
    private String stripeApiKey;

    public void processPayment(String token, int amount, String currency) {
        Stripe.apiKey = stripeApiKey;
```

```
        Map<String, Object> params = new HashMap<>();
        params.put("amount", amount);
        params.put("currency", currency);
        params.put("source", token); // Token obtained from client-side Stripe.js

        try {
            Charge charge = Charge.create(params);
            // Handle successful payment
        } catch (Exception e) {
            // Handle payment failure
        }
    }
}
```

5. **Controller Endpoint**: Create a Spring MVC controller to handle payment requests:

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class PaymentController {

    @Autowired
    private PaymentService paymentService;

    @PostMapping("/charge")
    public String charge(@RequestParam("token") String token,
                         @RequestParam("amount") int amount,
                         @RequestParam("currency") String currency) {
        paymentService.processPayment(token, amount, currency);
        return "Payment processed successfully";
    }
}
```

6. **Frontend Integration**: On the frontend side, integrate Stripe.js to securely collect payment information and obtain a payment token.

7. **Testing**: Test the payment integration in a sandbox or test environment provided by Stripe before deploying to production.

8. **Error Handling**: Implement error handling and exception management in the service and controller to handle payment failures and other issues.

This example demonstrates a basic integration of Stripe into a Spring Boot application. Depending on your requirements, you may need to implement additional features such as handling webhooks, managing customer information, or supporting recurring payments. Refer to the Stripe documentation for more details on advanced integration options and best practices.