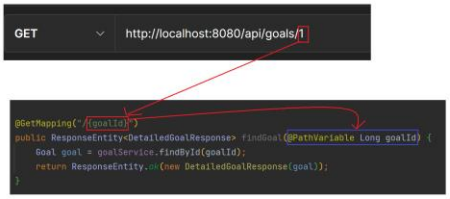
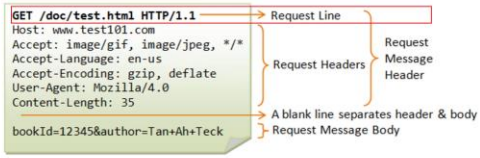
# Recebendo Requests - Spring Web

Os **requests que recebemos dependem do contrato que os endpoints possuem**. As vezes podemos retornar dados sem esperar nada de quem está fazendo a requisição, mas as vezes **precisamos que algumas informações sejam passadas.**

# PathVariable - Spring Web

As informações passadas podem ser através de **PathVariables**, quando usamos essa anotação estamos dizendo que é obrigatório que o **client passe a informação pelo caminho**:

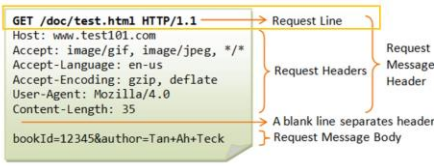



# RequestParam - Spring Web

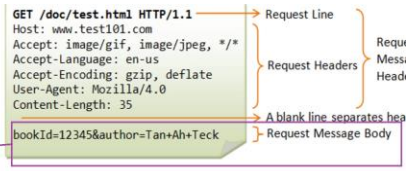
Outra forma de receber parâmetros é usando o **RequestParam,** a diferença é que esse tipo de parâmetro **é opcional**. Eles são passados após a **"?"** no caminho e podem ser separados com **"&"** a cada novo parâmetro.

Eles são bem interessantes de se utilizar quando **entregamos um endpoint** que **pode ter vários parametros** para moldar a resposta, um bom exemplo é um **endpoint que entrega o resultado paginado**, podemos fazer com que a **resposta seja de acordo com os parâmetros escolhidos do cliente:**

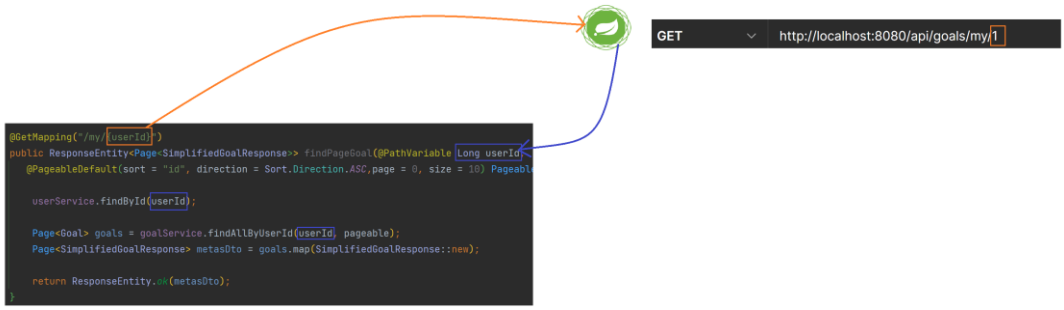

# RequestBody - Spring Web

Quando as **informações enviadas são extensas ou precisam ser fortemente protegidas**, elas são enviadas pelo **corpo da requisição**. Essas informações são conhecidas como ***payload***, é muito mais complicado interceptar informações pelo payload da requisição do que pela url, além de possuir mais espaço para enviar objetos complexos:

# Validação e conversão de parâmetros - Spring Web

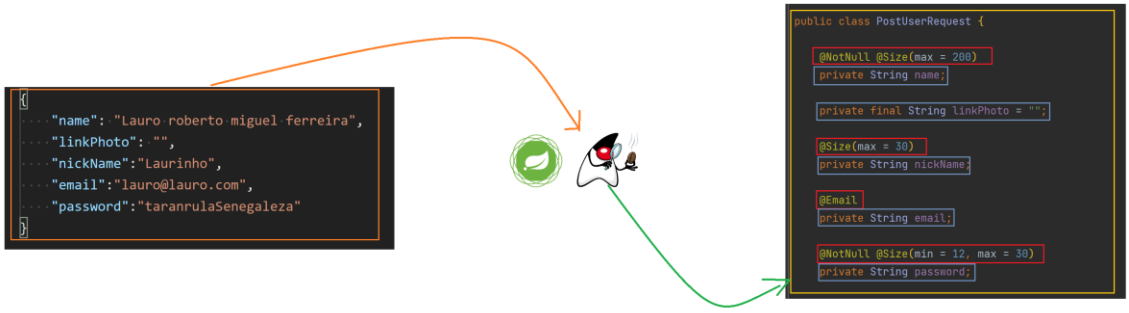
O **Spring** **consegue automaticamente converter os parâmetros passados pelo cliente**, tanto na *URI como no Body*, para os **tipos específicos do Java**, ou melhor, **ele tenta fazer essa conversão e na maioria das vezes funciona:**




# Validação com Spring validation – Spring Web

A **conversão automática para variáveis de URL do Spring serve muito bem**, afinal de contas são poucos parâmetros e uma validação manual pode ser tranquila de se fazer. Mas quando queremos **pegar o valor de uma requisição pelo body** isso muda um pouco, geralmente estamos pegando um objeto complexo e com **vários atributos.**

Nesse tipo de cenário podemos usar o **Spring Validation,** com ele podemos anotar os **atributos** da representação do body e definir **restrições com anotações**, por exemplo, *tamanho, se é maior que 0, se não é branco ou nulo etc....*:




O método do controller que vai usar essa validação precisa indicar isso com **@Valid** no parâmetro.

Outra dica para facilitar a conversão de determinados atributos é usar o **@JsonFormat** caso sua api trabalhe com Json, com essa **anotação** podemos especificar o formato esperado da data, **tornando fácil para o Spring converter o atributo em uma classe específica de Data do Java:**