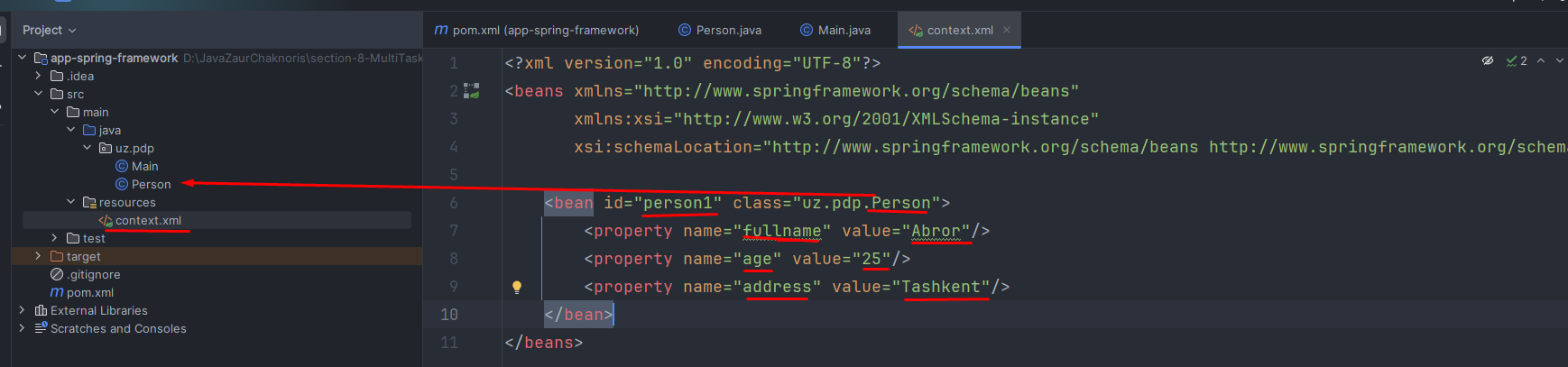
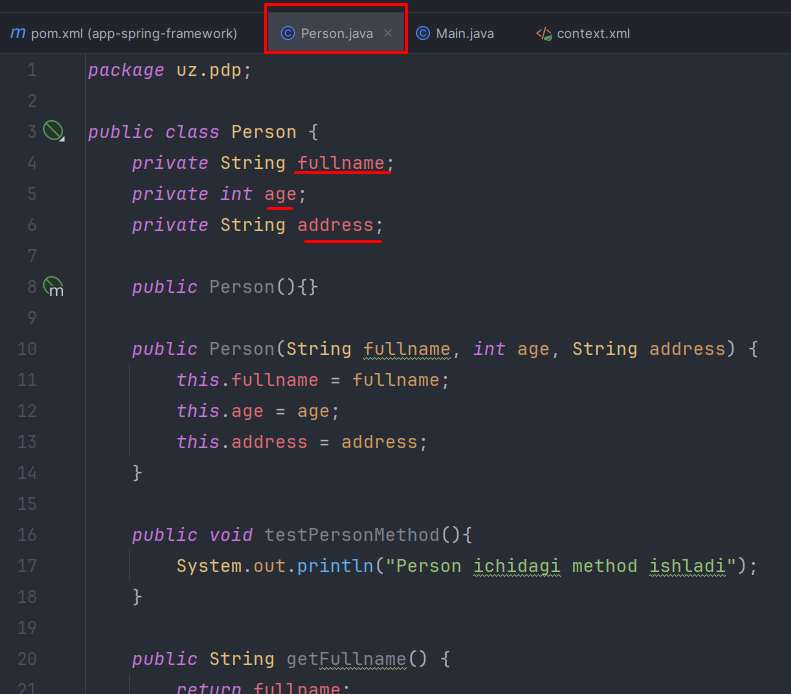
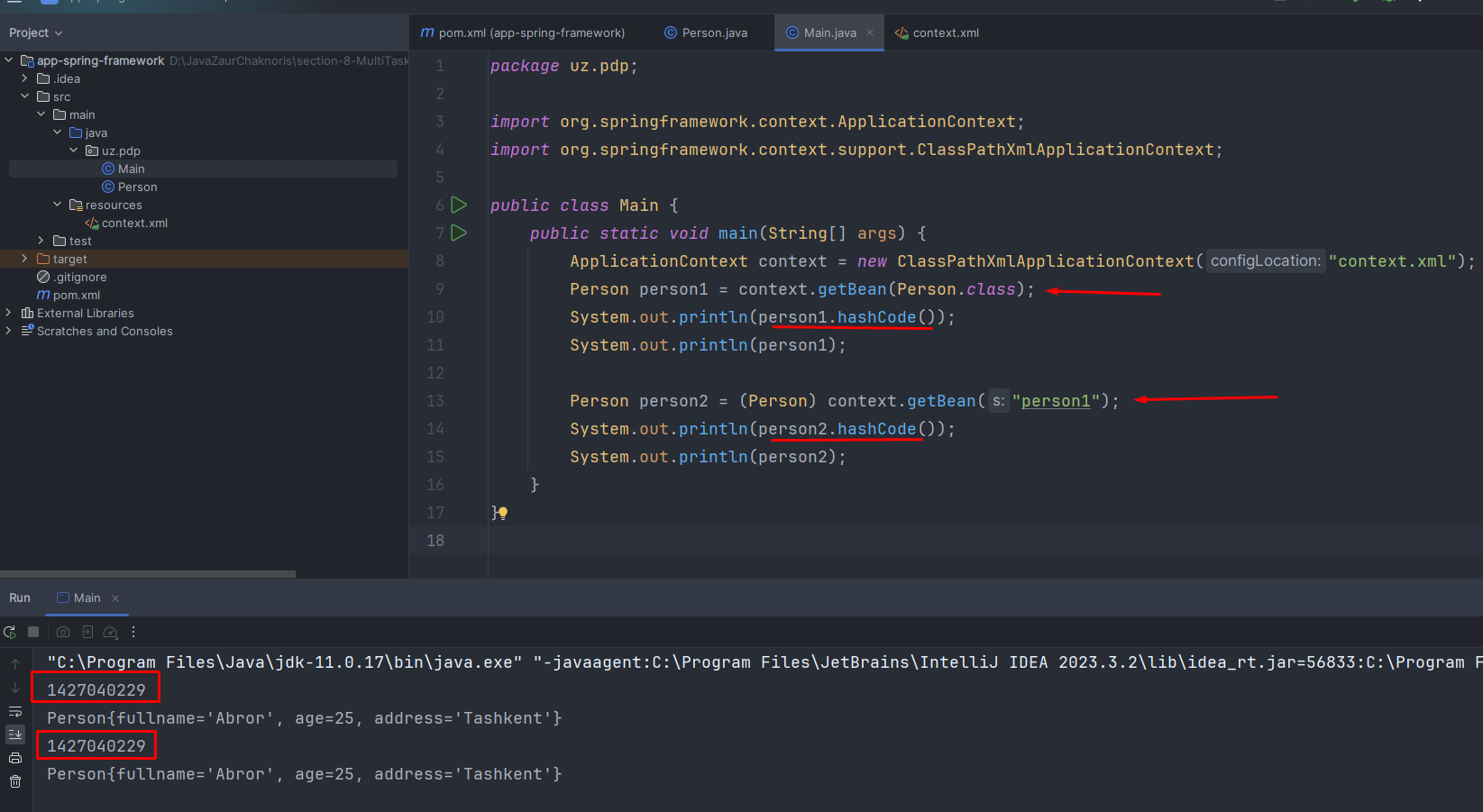
Bean ni yaratishni 2 xil usuli bor. Bittasi context.xml dan foydalanishdir. Xmlda bean tegi ochib, unga id beramiz. Bu yerda id yagona bolishi kerak xml faylda. Fullname,age va address propertylar Person.java classdagi fieldlardir.



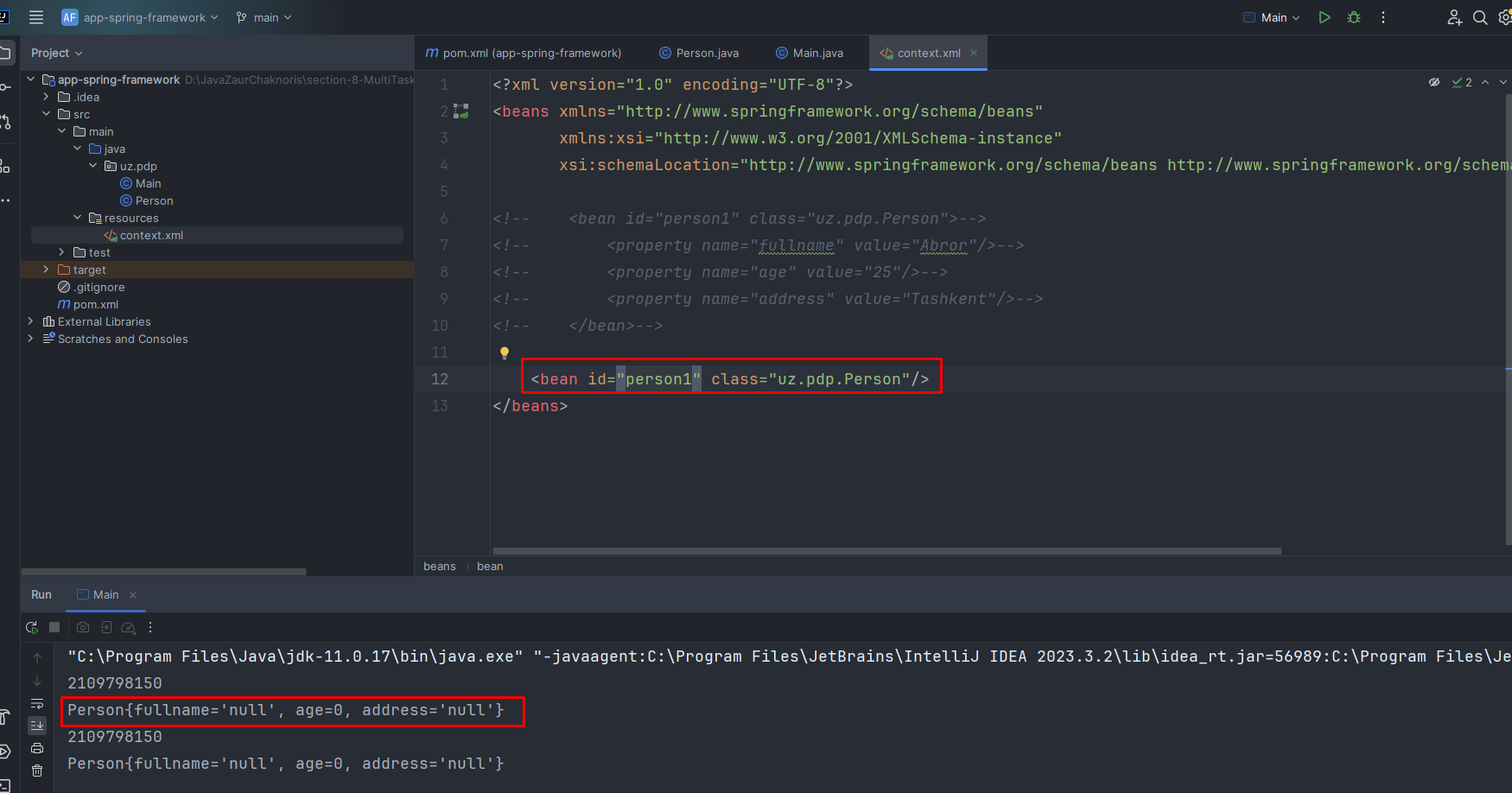
Person class:



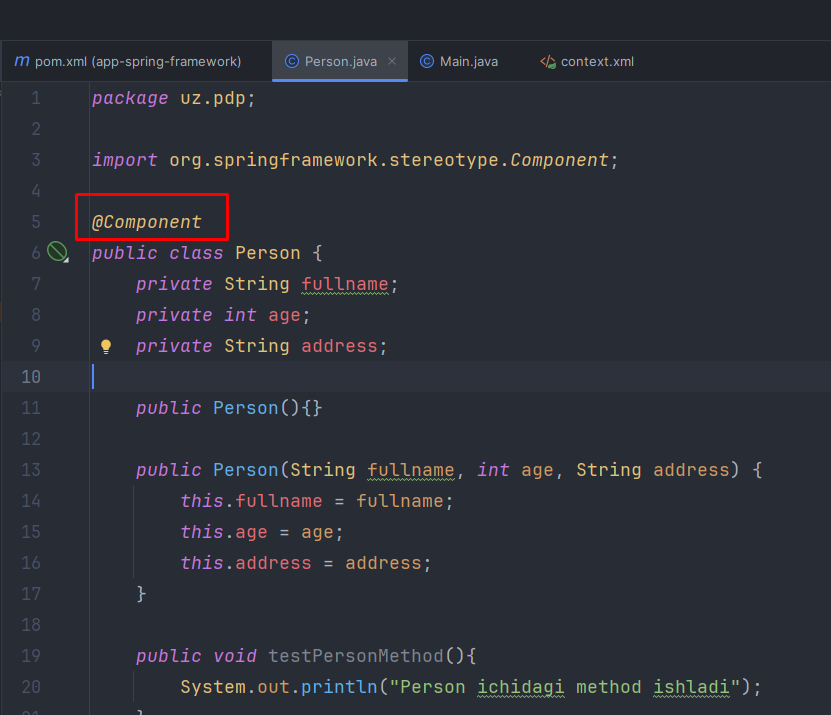
Endi keling Main classda shu beanlarni chaqirib ko’raylik. Bean ni istasak 9-qatorda chaqirgandek Person.class qilib chaqirish mumkin, xohlasak 13-qatorda chaqirgandek chaqirish mumkin, farqi yo’q. E’tibor bergan bo’lsangiz biz Person classidan 2 marta object oldik, ya’ni person1 va person2 object. Lekin bitta object yaratildi buni consoledagi hashcode dan ham bilish mumkin. Sababi bean dagi objectlar faqat bir marta yaratiladi xattoki istalgancha marta chairsak ham. Bu narsa har safar object yaratib memoryni ortiqcha ishlatmaslik uchun qilingan:



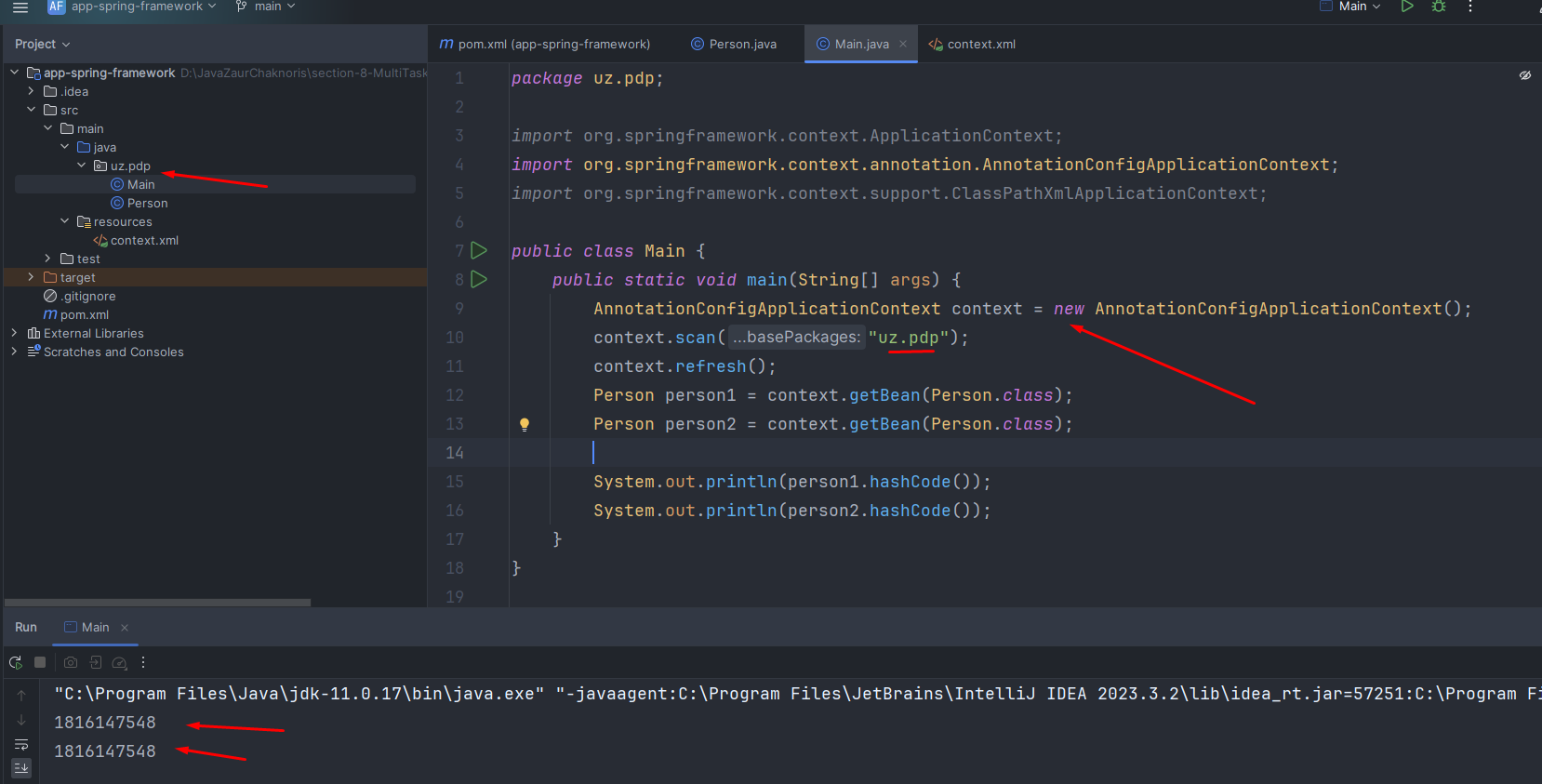
Agar istasak hech qanday boshlang’ich qiymat berilmagan object yaratish mumkin:



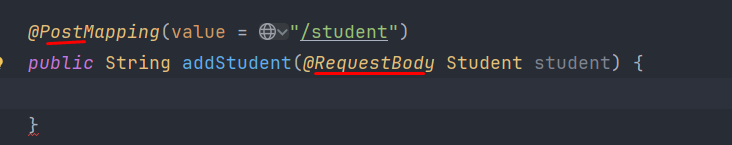
Bean ni yaratishni 2-usuli bu anotatsiyadan foydalanishdir. Person classga @Component annotatsiyani yozamiz. Bu anotatsiya classni bean sifatida e’lon qilyapti degani:



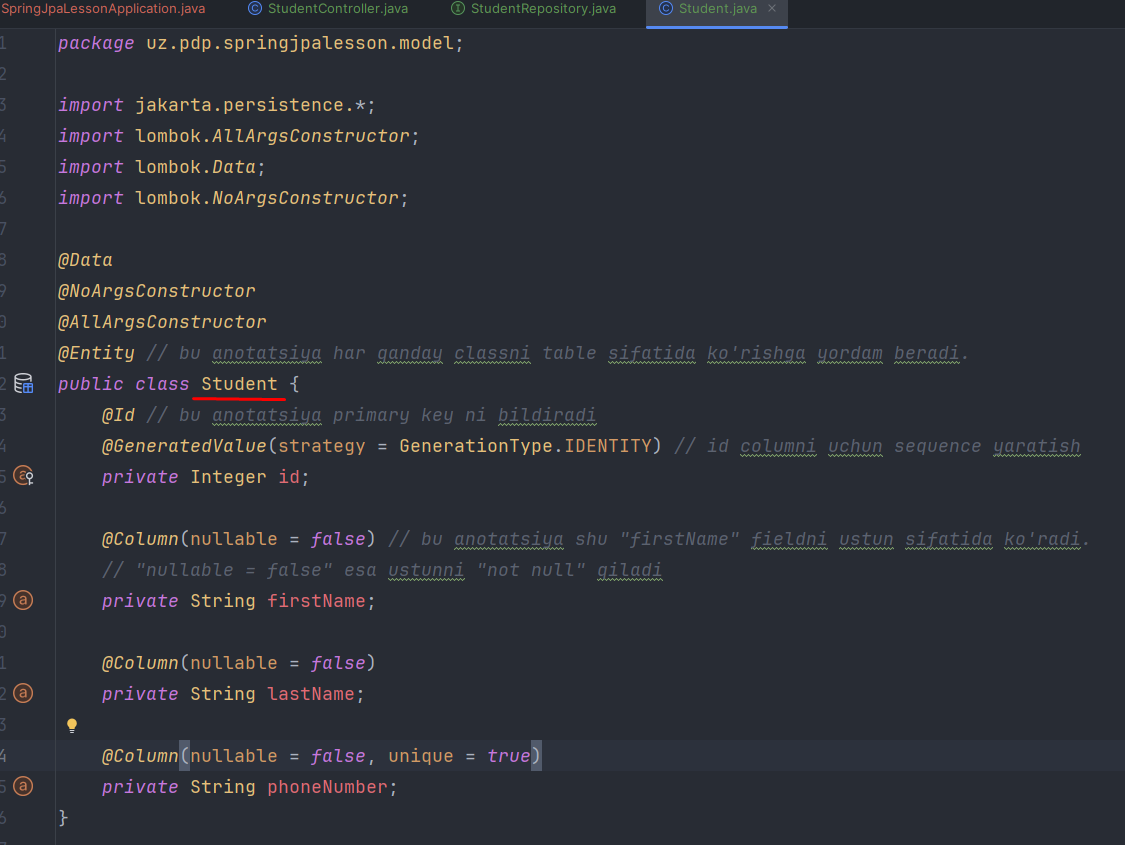
Bu holatda ham person1 va person2 ni hashcode bir xil bo’lyapti:



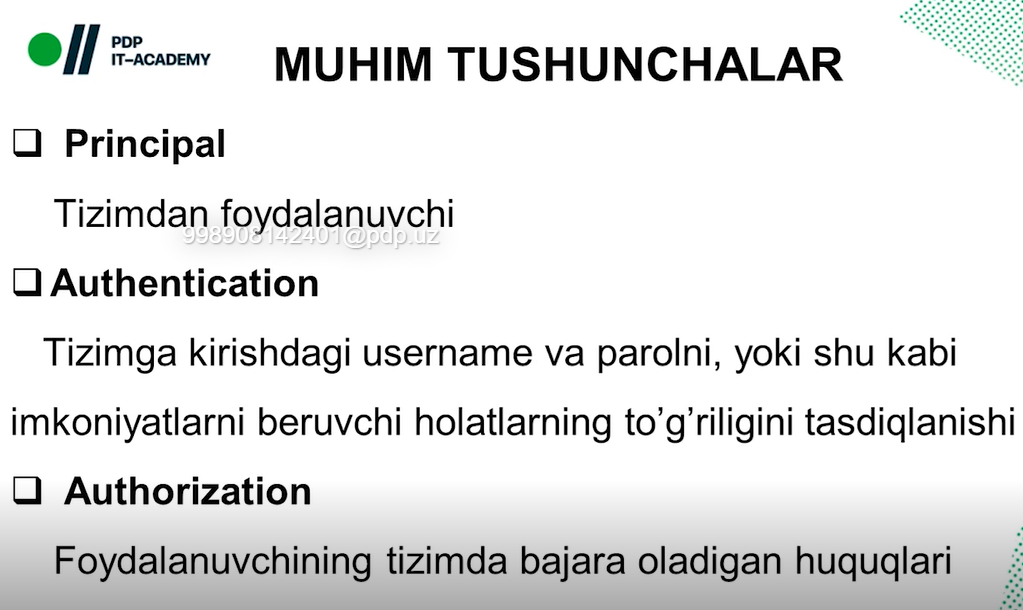
@RequestBody ann.si POST method orqali kelgan JSON objectini javaning Student classiga o’girib beradi va uni student o’zgaruvchisiga o’zlashtiradi:

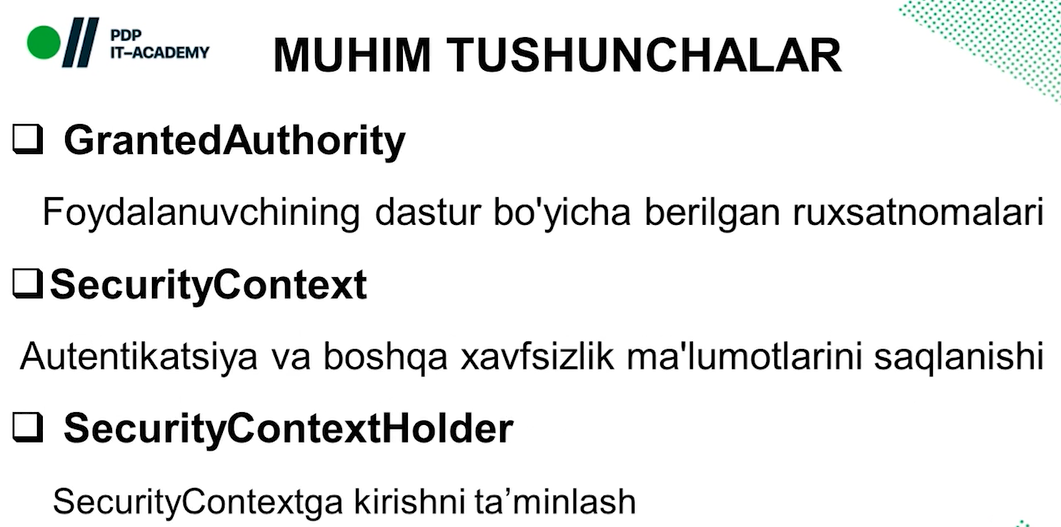


Student classi:

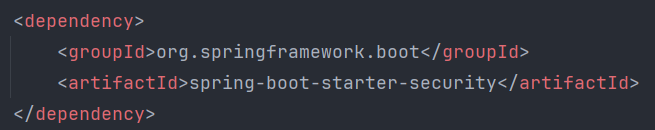


**Spring security**





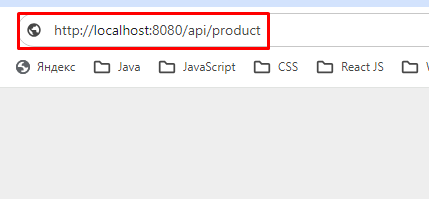
Spring securityda ishlash uchun pastdagi library kerak bo’ladi:



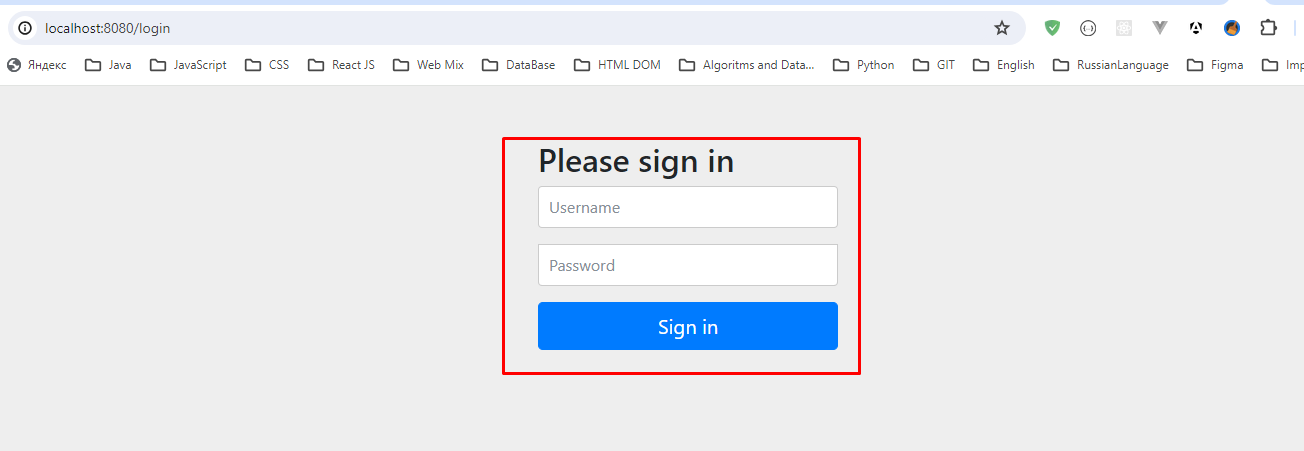
Default holatda biz security ishlatmasak, istalgan API ga so’rov istalgan method(GET, POST, PUT, DELETE) ga so’rov jo’natish mumkin. Lekin yuqoridagi libraryni ulasak, u holda biz bunday qila olmaymiz. Masalan biz pastdagi API ga browserdan so’rov jo’natib ko’ramiz.



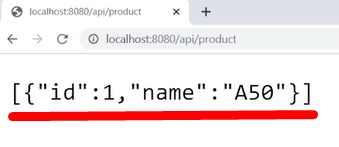
Browserdan shu API ga so’rov jo’natyapmiz:



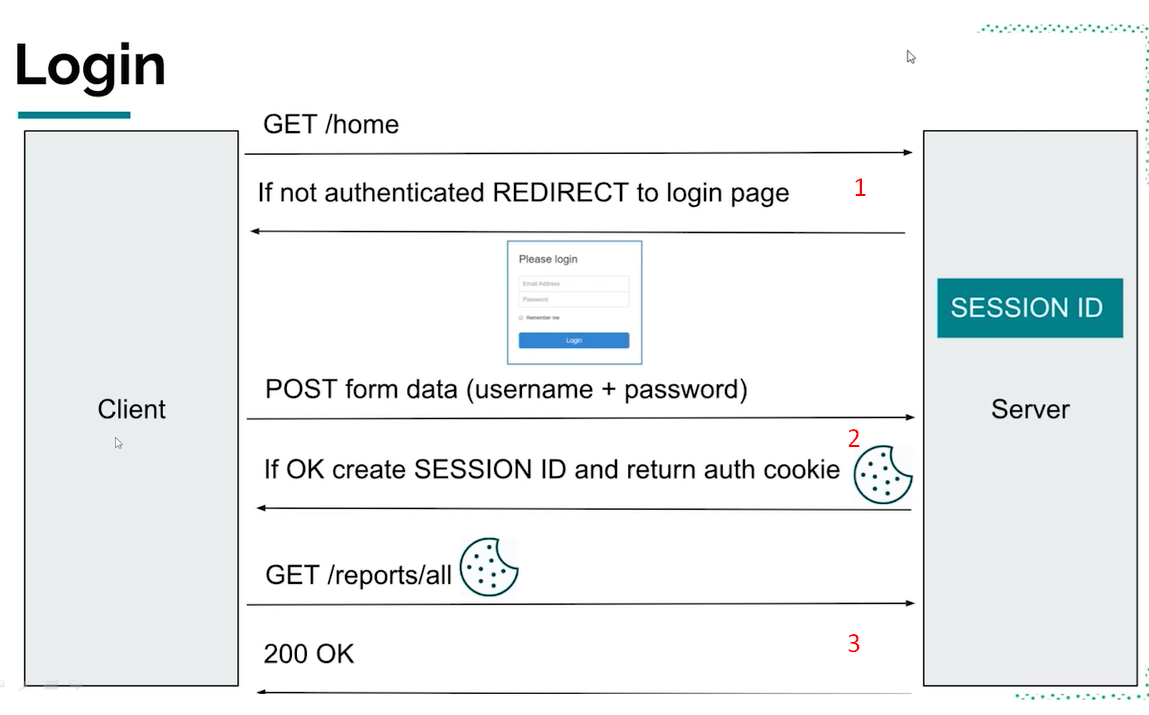
Lekin bizni login page ga redirect qilib yuboradi. Bu turdagi authenticated esa “FORM BASED AUTHENTICATED” deyiladi, ya’ni “formaga asoslangan authenticated” deyiladi. Bu browserdan forma orqali kirishni nazarda tutiladi:



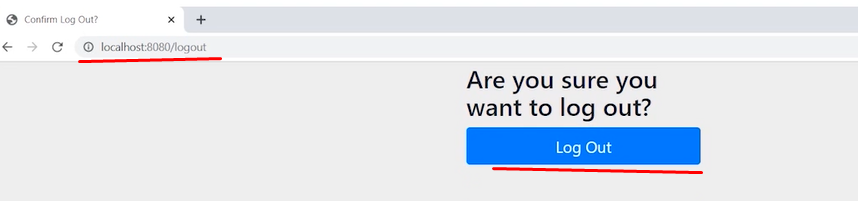
Endi biz 2-marta so’rov jo’natsak, bizga data lar keladi:

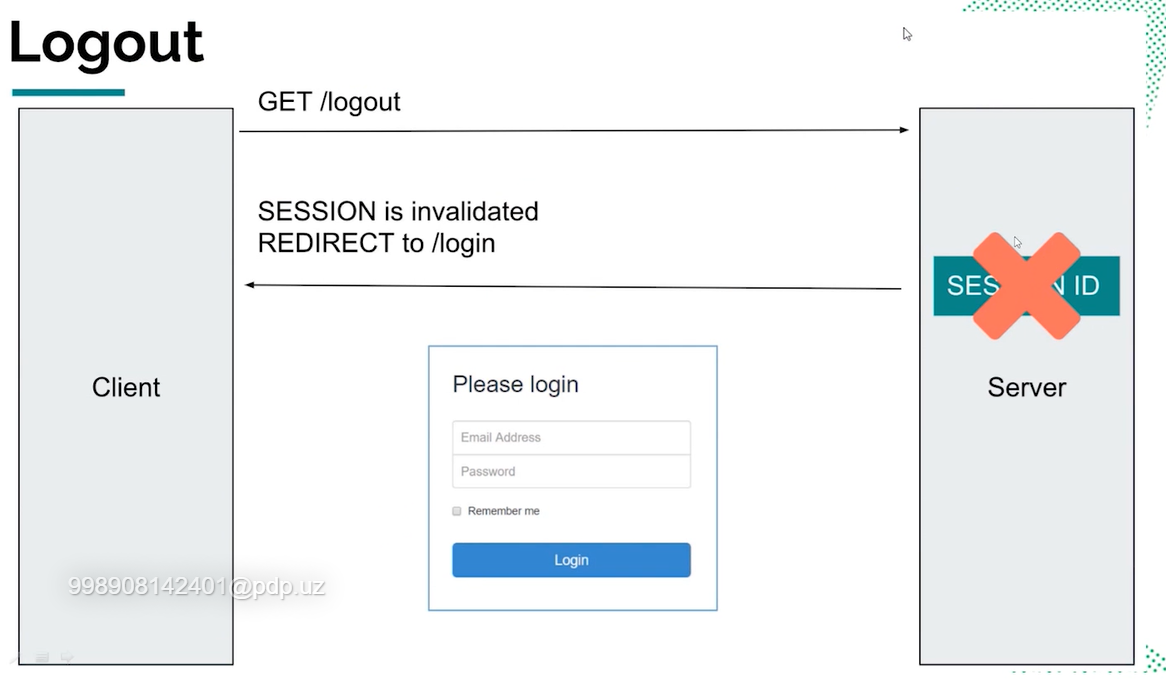


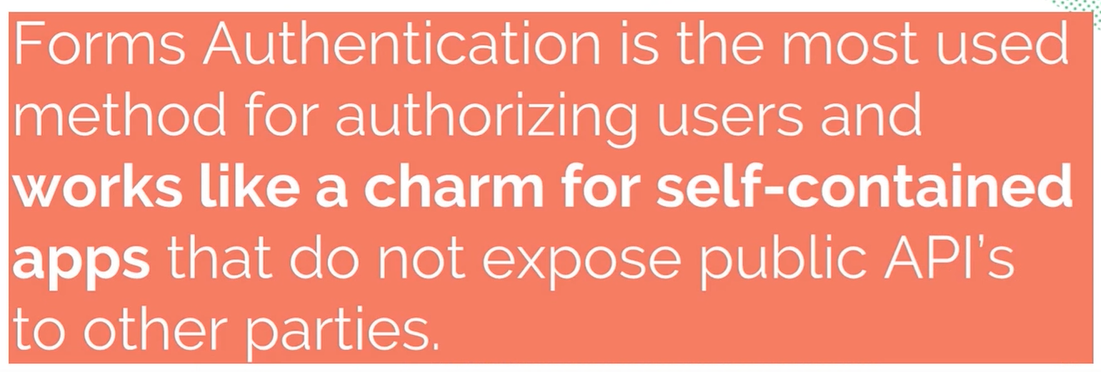
Yuqoridagi holatni esa pastdagi rasmda visual ko’rinishi. 1-bosqichda biz deylik **GET/home** API ga so’rov jo’natamiz va bizni login pagega redirect qildi. Endi 2-bosqichda esa biz **POST** qilib username va passwordni jo’natamiz. Passwordni console da ko’rib olamiz, o’zi generatsiya qiladi, username esa **“user”** bo’ladi. Shunda biz fieldlarni to’ldirib jo’natsak, server bizga SessionID generatsiya qilib, uni cookie orqali clientga jo’natadi. 3-bosqichda esa biz API larga shu cookie ni jo’natib so’rov jo’natamiz. Agar serverga restart bersak, sessionID qaytadan yangilanadi, bu esa sessionID eskiradi. Keyin browserdan so’rov jo’natsak, eskirganligi uchun kirib bo’lmaydi, yana qaytadan formadan username va passwordni terib kirish kerak:

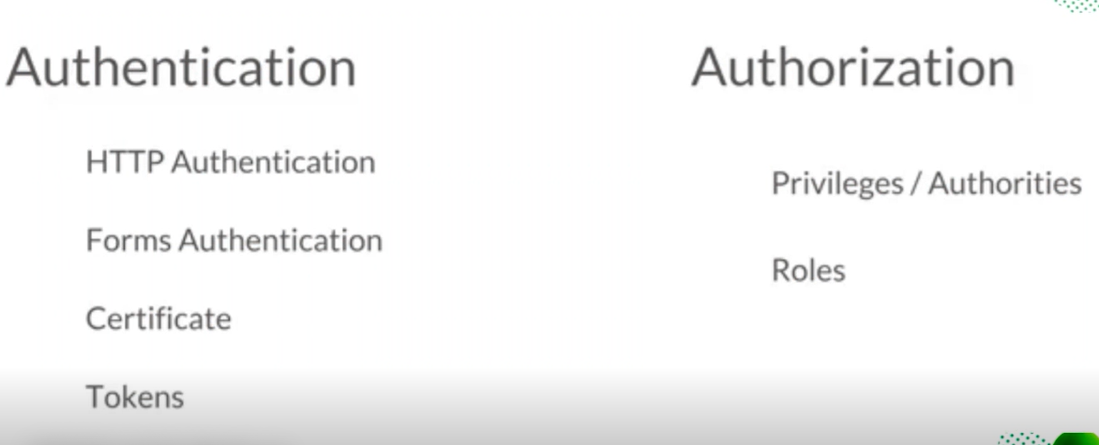


Agar logout qilib chiqib ketmoqchi bo’lsak, pastdagi url ga o’tamiz va log out ni bosib chiqib ketamiz:

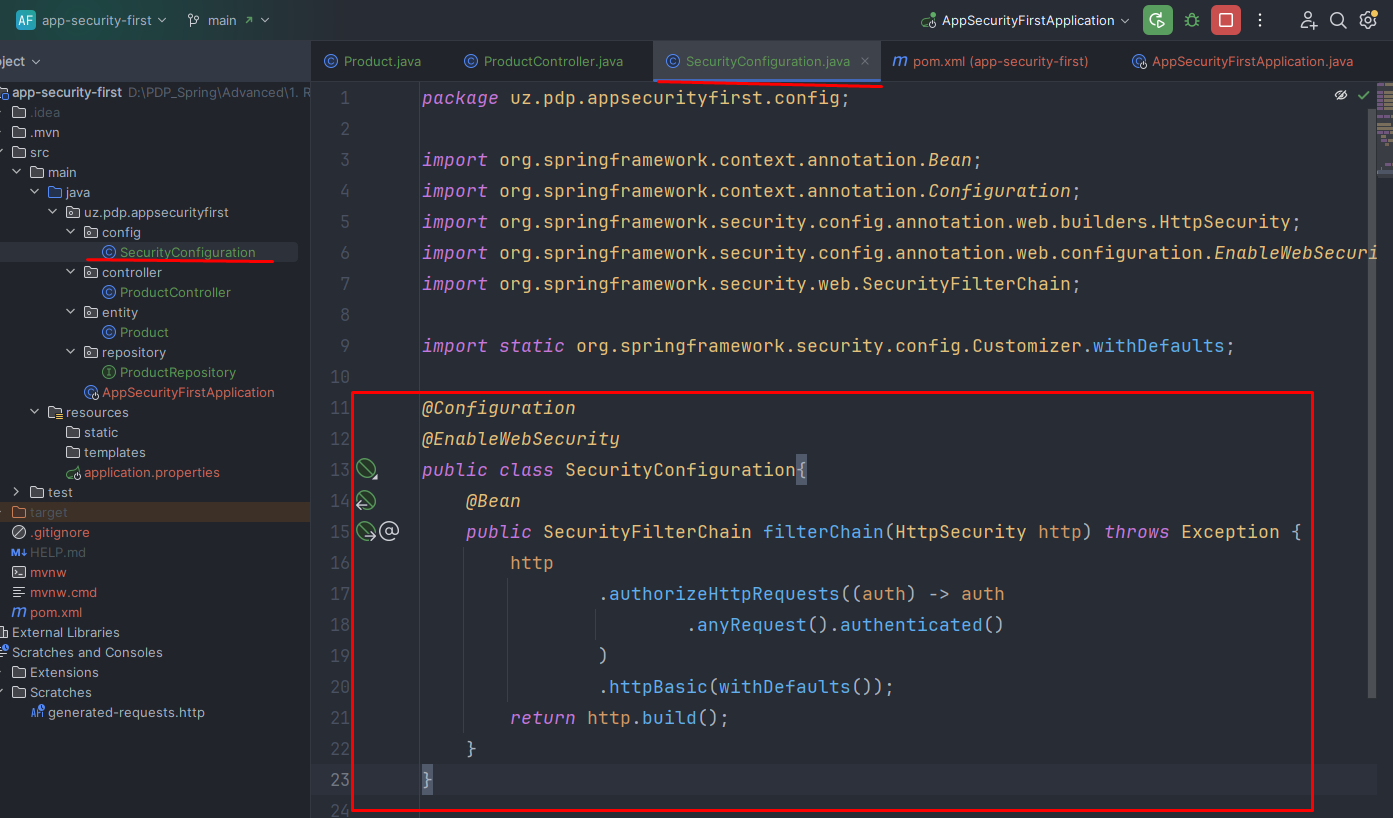




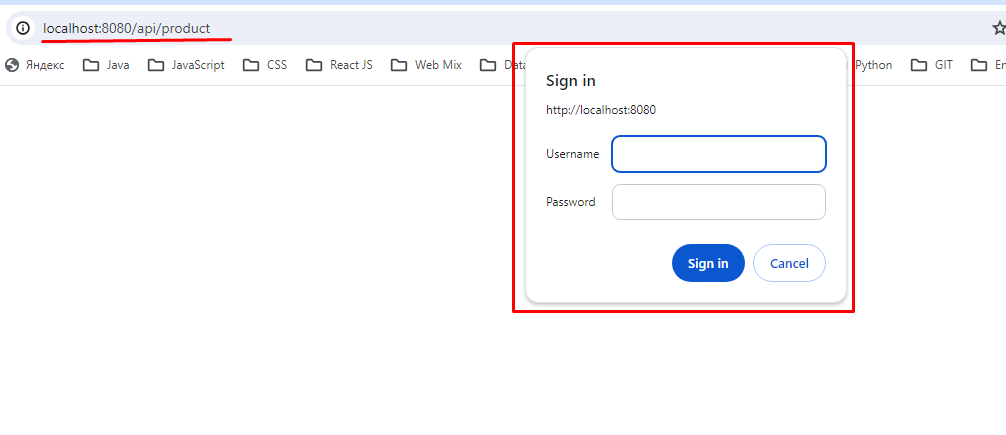




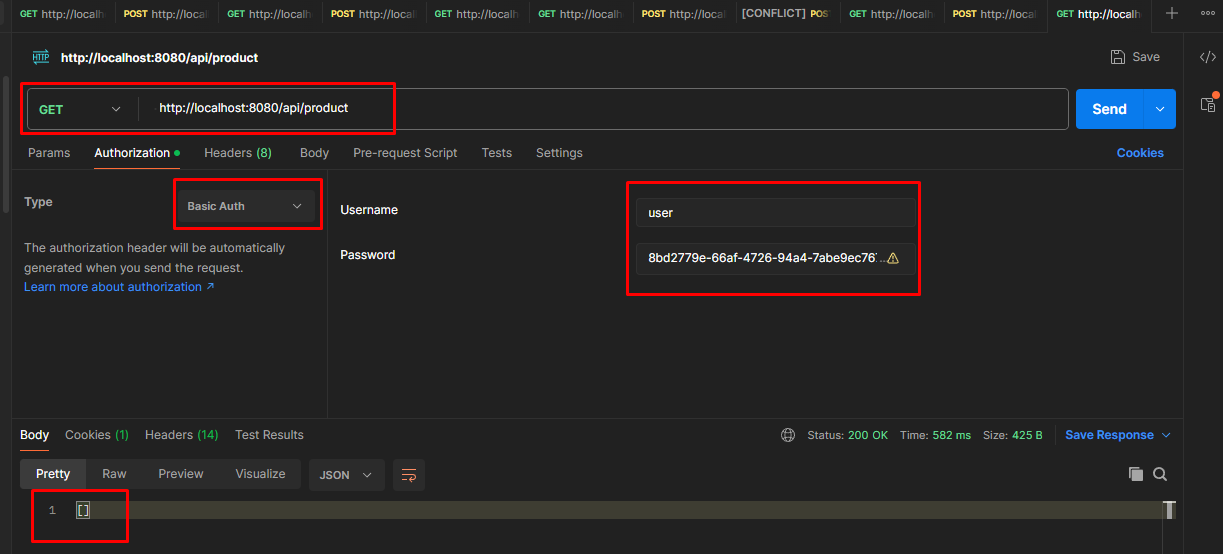
Endi esa **“BASIC AUTHENTICATED“** ni ko’ramiz. Buning uchun esa biz maxsus configuration faylni yozishimiz kerak:



Endi esa yana API ga so’rov jo’natsak, bu safar username va password formada emas, balki alert da chiqadi. Chunki bu form based auth. Emas balki basic auth. hisoblanadi.

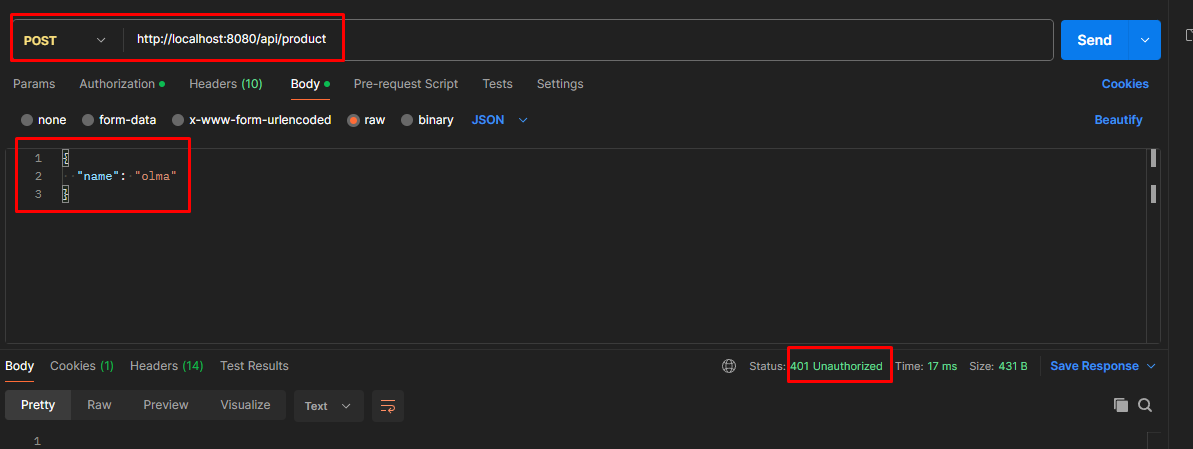


Istasak alert dagi polyalarni to’ldirishimiz mumkin, yoki Postman dan ham jo’natish mumkin.

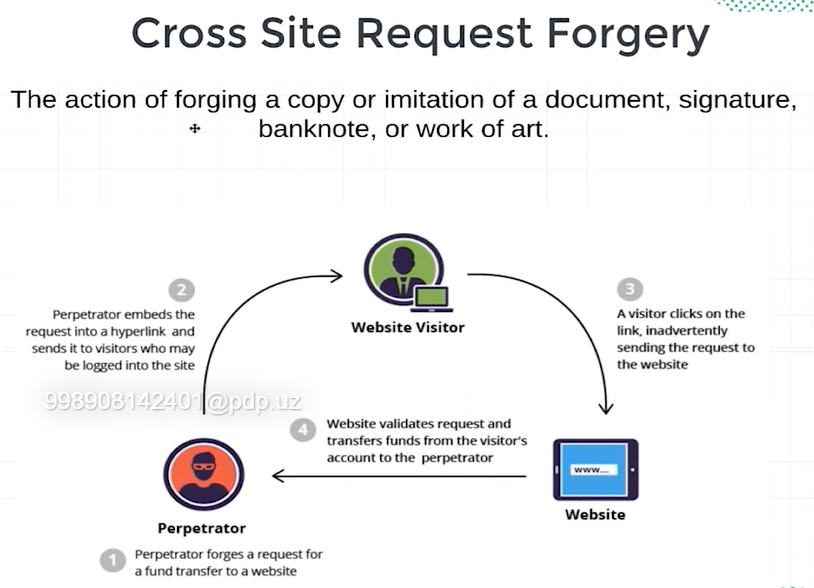


Basic auth.ni form based authdan farqi shundaki, form based auth.da faqat biz browserdan jo’nata olamiz username va passwordni, chunki forma ga asoslangan.

Endi yuqordagi holatda biz GET so’rovini yuborib ma’lumotni oldik, bo’sh array ekan. Statusi 200 OK ekan. Lekin biz POST, PUT, DELETE kabi methodlardan foydalanib ma’lumotni o’zgartirmoqchi bo’lsak, bizga 401 statusni qaytaradi, ya’ni qo’ymaydi o’zgartirgani:



Nega biz bunday statusni



**Role Based Authorization**

Authorization da 2 ta asosiy tushuncha bor. 1) Privilages – bu huquqlar degani. Bu biror ishni bajara olishga bo’lgan huquq. 2) Roles – rollardir(lavozim). Misol bilan tushunaylik. Omborxona bor deylik unda 2ta ishchi ishlaydi, biri director(role bu DIRECTOR) va oddiy ishchi(role bu WORKER). Demak bizda 2 ta role bor DIRECTOR va WORKER. Endi ularni qila oladigan ishlari(huqulari) ham har xil bo’ladi. Masalan DIRECTOR hamma mahsulotlarni ko’ra oladi, o’chira oladi, update qila oladi va qo’sh oladi. WORKER esa faqat mahsulotlarni qo’sha oladi xolos.

