****

**Apache POI** Java kutubxonasi boʻlib, **Word**, **Excel** va **PowerPoint** kabi *Microsoft* *Office* hujjatlarini **yaratish**, **oʻqish** va **oʻzgartirish** imkonini beradi. U ko'pincha Spring ilovalarida foydalanuvchi so'rovlariga javoban hujjatlarni **dinamik** ravishda yaratish uchun ishlatiladi.

"**Dinamik**" degani, *oldindan belgilangan yoki statik bo'lishdan farqli o'laroq, biror narsa ish vaqtida yaratilgan yoki yaratilganligini anglatadi*. Spring ilovasida hujjatlarni **dinamik** ravishda yaratish kontekstida, bu hujjat foydalanuvchi so'roviga yoki boshqa hodisaga javoban tezda yaratilishini anglatadi va hujjatning mazmuni foydalanuvchi kiritishi yoki boshqa omillar asosida ish vaqtida aniqlanadi. .

Masalan, Spring ilovasi ma'lum bir vaqt oralig'idagi savdo ma'lumotlari hisobotini o'z ichiga olgan Word hujjatini yaratish uchun **Apache POI**-dan foydalanishi mumkin. Foydalanuvchi hisobotning boshlanish va tugash sanalarini kiritishi mumkin va Spring ilovasi ushbu ma'lumotlardan o'sha vaqt oralig'idagi savdo ma'lumotlarini o'z ichiga olgan Word hujjatini dinamik ravishda yaratish uchun foydalanadi.

Bundan farqli o'laroq, statik hujjat oldindan belgilangan va ish vaqtida o'zgarmaydigan hujjat bo'ladi. Masalan, dasturiy ta'minot ilovasi uchun foydalanuvchi qo'llanmasi bir marta yaratilgan va dasturiy ta'minotning yangi versiyasi mavjud bo'lmaguncha o'zgarmaydigan statik hujjat bo'lishi mumkin.

**Intervyyularda Apache POI ga oid quyidagi misollar berilishi mumkin:**

1.Microsoft Office fayllarini (Word, Excel, PowerPoint) qanday o'qish va yozish mumkin?

2.Apache POI-da qanday xatoliklar ko'rsatish mumkin?

3.Apache POI-ni boshqa kutubxonalardan farqli qanday xususiyatlari bor?

4.XLSX va XLS formatidagi fayllar orasidagi asosiy farqlar nima?

5.Qanday sabablarda faylning formatini o'zgartirish kerak bo'lishi mumkin?

6.POI kutubxonasini ishlatish orqali qaysi amallar bajarilishi mumkin?

7.Microsoft Excel yoki Word dokumentidan maydonlarni yuklash va ularga qiymatlar yozish mumkinmi?

8.Apache POI kutubxonasidan foydalanishda dastlab bajarilishi kerak bo'lgan qadamlar nimalar?

9.Qanday sabablarda faylni ochishda yoki yaratishda xatolik yuz beradi?

**Javoblar**

**1.Microsoft Office fayllarini (Word, Excel, PowerPoint) qanday o'qish va yozish mumkin?**

**Apache POI** kutubxonasini ishlatib, Java dasturlari yordamida **Microsoft Office** fayllarini (**Word**, **Excel**, **PowerPoint**) **o'qish** va **yozish** mumkin.

Bu kutubxona orqali **Microsoft Office** fayllarini **o'qish**, **yozish** va ulardan tashqari xususiyatlar (masalan**, maydon nomlari**, **maydon turlari**, **qiymatlar**, **jadvallar**, **rasm** va **texst yozyuvlar**, shuningdek, yagona bir belgini **boshqa formatdan** **boshqa formatga konvertatsiya qilish**) bajarish mumkin.

**Excel fayllariga misol** uchun, **POI** kutubxonasida **XSSFWorkbook**, **HSSFWorkbook** va **SXSSFWorkbook** klasslari mavjud. Ushbu klasslar orqali **Excel** fayllarini o'qish, yozish va ulardan tashqari amallarni bajarish mumkin.

**Word fayllariga misol** uchun, **POI** kutubxonasida **XWPFDocument** klassi mavjud, shuningdek **PowerPoint** fayllari uchun **XSLFSlideShow** klassi ishlatiladi.

Bunday ko'rsatmalarga qarab, dasturchilar kutubxonadan foydalanib, har qanday **Microsoft Office** fayllari bilan ishlashlari mumkin.

**# Apache POI kutubxonasi faqat java uchun ishlatiladimi?**

Ha, Apache POI kutubxonasi faqat Java uchun ishlatiladi.

**Apache POI** kutubxonasi Java dasturlash tilida **Microsoft Office** fayllarini **o'qish** va **yozishni** **osonlashtirish** uchun yaratilgan. *Bu kutubxona faqat Java dasturlash tilida ishlatiladi va boshqa dasturlash tillari uchun mos qilib yaratilmagan*.

Bundan tashqari, **Apache POI** kutubxonasi faqat **Microsoft Office** fayllariga oid formatlarga qo'shimcha yordam berish uchun mo'ljallangan. Masalan, **Word** **(.docx)** va **Excel** **(.xlsx)** fayllarini o'qish va yozishni osonlashtirish uchun ishlatiladi.

**2.Apache POI-da qanday xatoliklar ko'rsatish mumkin?**

**Apache POI** kutubxonasida bir nechta xatoliklarni ko'rsatish mumkin. Xatoliklar yaratilgan dasturda, faylning shakli va boshqa sabablarga ko'ra farqli bo'lishi mumkin. Quyidagi xatoliklar amaliy ishda ko'rish mumkin:

Malumotlar faylga yozila olmadimi yoki xato yozildimi?

Faylni ochishda **xatoliklar** (masalan, **fayl yo'q**, **fayl shakli xato**, yoki **fayl hajmi katta** bo'lishi) chiqishi mumkin.

**Microsoft Office** dasturlarining eski versiyalari yoki yangi versiyalariga bog'liq qo'shimcha xatoliklar mumkin.

Fayllar yordamida amalga oshirilgan operatsiyalar soni katta bo'lsa, dastur to'xtash yoki javob bermay qolishi mumkin.

Fayl formati farq qilganda xatolar paydo bo'lishi mumkin.

**Apache POI** kutubxonasi va **Microsoft Office** dasturlari orasidagi mos kelishuv o'zgarishi mumkin.

**3.Apache POI-ni boshqa kutubxonalardan farqli qanday xususiyatlari bor?**

Apache POI Java dasturlash tillari kutubxonasi boʻlib, **Excel**, **Word**, **PowerPoint**, **Outlook** va **Visio** fayllarini **oʻqish**, **yozish** va **tahrirlash** imkonini beradi. Bu kutubxona boshqa kutubxonalardan quyidagi xususiyatlari bilan farq qiladi:

**1.Platforma mosligi**: **Apache POI** Java dasturlash tilida yozilgan, shuning uchun u har qanday operatsion tizimda ishlaydi.

**2.Barcha Microsoft Office formatlarini qo'llab-quvvatlash**: **POI kutubxonasi** **Microsoft Office 97**-dan beri yaratilgan barcha formatlarni (**XLS, XLSX, DOC, DOCX, PPT, PPTX, VSD, MSG**) qo'llab-quvvatlaydi.

**3.O‘qish va yozish**: **POI kutubxonasi** fayllarni o‘**qish**, **yozish** va **tahrirlash** imkoniyatlarini taqdim etadi. Misol uchun, siz **Excel** faylini **o'qishingiz**, ma'lumotlarni **qo'shishingiz** va uning **formatini** **o'zgartirishingiz** mumkin.

**4.Uzluksiz rivojlanish:** **Apache POI** rivojlanishda davom etmoqda, shuning uchun bu kutubxona yangi funksiyalar va metodlarni joriy qilish uchun ko'plab imkoniyatlarga ega.

**5.Ochiq manba**: **Apache POI** ochiq kodli tarjima, foydalanish va foydalanishni yoqish orqali bir qadam oldinda.

**6.Xavfsizlik**: **Apache POI** **Microsoft Office** fayllari uchun **oʻzgarishlarni saqlash** va **fayl yopilganda avvalgi holatga qaytish** kabi **xavfsizlik** va **ruxsatlarni boshqarish** funksiyalarini taqdim etadi.

**4.XLSX va XLS formatidagi fayllar o'rtasidagi asosiy farqlar qanday?**

**XLS** va **XLSX** formatlari **Microsoft Excel** fayl formatlaridir. Bundan tashqari, **XLS** formati **Excel 2003** va undan oldingi versiyalarida, **XLSX** formati esa **Excel 2007** va undan keyingi versiyalarida ishlatilgan. Ular orasida quyidagi asosiy farqlar mavjud:

**1.Fayl kengligi**: **XLS** **formatidagi fayllar kengligi cheklangan** edi, shuning uchun ularga qo'shimcha ma'lumotlarni kiritishda cheklangan cheklovlar mavjud edi. **XLSX fayllari esa kengligi bo'yicha cheklanmagan** va shuning uchun ularga juda ko'p ma'lumotlarni kiritish mumkin.

**2.Fayl yozish**: **XLS** **formatidagi fayllar ikkilik formatda yozilgan**, shuning uchun ularni **tahrirlash** va **o'qish** **qiyin** bo'lishi mumkin. **XLSX fayllari esa XML formatida yozilgan**, **shuning uchun ularni tahrirlash va o'qish osonroq**.

**3.Fayl hajmi**: **XLS formatidagi fayllar katta** bo'lishi mumkin, shuning uchun ularni **yuborish** va **saqlash** **katta disk yoki xotirani talab qilishi mumkin**. **XLSX** **formatidagi fayllar hajmi ham kichik bo'lishi mumkin, shuning uchun ularni saqlash va yuborish oson.**

**4.Komponentlarning o'zaro ishlashi**: **XLS formatidagi fayllar bir nechta komponentlardan foydalanishni talab qilishi mumkin**. **XLSX formatidagi fayllar esa komponentlar orasidagi hamkorlikdan yaxshiroq foydalanish imkonini beradi**.

**Qisqacha:** **XLS** va **XLSX** **formatlari Excel uchun fayl formatlaridir**. **XLSX formati** XML formatida yozilgan va o'lchami kichik bo'lishi mumkin. XLS formatidagi fayllar ikkilik formatda yoziladi va hajmi katta bo'lishi mumkin.

**5. Qaysi sabablarga ko'ra fayl formatini o'zgartirish zarur?**

Fayl formatini o'zgartirishning ba'zi sabablari:

**Faylni ko'p vaqt talab qiluvchi dasturlar yordamida ochish**: *Fayl formati dastur tomonidan qo'llab-quvvatlanmasligi mumkin*. *Bunday holda*, *fayl formatini o'zgartirish kerak bo'lishi mumkin, masalan, faylni Excelda ochish uchun uni XLS formatidan XLSX formatiga o'zgartirishingiz mumkin*.

**Foydalanuvchi talablari:** *Fayllarni boshqa foydalanuvchilar bilan baham ko'rish uchun fayl formatini o'zgartirish kerak bo'lishi mumkin*. Masalan, foydalanuvchi tomonidan Microsoft Word dasturida yaratilgan faylni **Macintosh kompyuterida ochish uchun uni DOC dan DOCX ga o'zgartirish mumkin.**

**Kompyuter yadrolarining cheklovlari:** *fayl formati yadrolar bilan bog'liq bo'lishi mumkin. Bunday holda, fayl formatini o'zgartirishingiz kerak bo'lishi mumkin, masalan, faylni PDF formatidan Word formatiga o'zgartirishingiz mumkin, shuningdek, faylni ZIP formatidan RAR formatiga o'zgartirishingiz mumkin.*

**Kengaytirilgan funksiyalar**: Fayl formatini oʻzgartirish faylning kengaytirilgan funksiyalariga kirishni ochishi mumkin. Masalan, Microsoft Word da yaratilgan DOC fayliga joylashtirilgan tasvir, video yoki audio elementlardan foydalanish uchun faylni DOCX formatiga aylantirish mumkin.

**Ma'lumotlarni saqlash xavfsizligi**: o'z fayllarini saqlash xavfsizligini ta'minlash uchun foydalanuvchilar fayl formatini o'zgartiradilar. Bunday holda, masalan, faylni **TXT**-dan **PDF**-ga o'zgartirishingiz kerak bo'lishi mumkin, shuningdek, faylni **XLS**-dan **XLSX**-ga aylantirishingiz kerak bo'lishi mumkin.

**Shunday qilib, fayl formatini o'zgartirish bir necha sabablarga ko'ra kerak bo'lishi mumkin, ammo bu jarayon fayl ma'lumotlarini o'zgartirmaydi.**

Bu xatoliklar, **Apache POI** kutubxonasi yordamida fayllar ustida amaliy ishlar bajarilgan paytda chiqishi mumkin. Xatoliklarni aniqlash va tuzatish uchun, loyihada xatolar tizimini ko'chirish va fayllar ustida amaliyotlarni qayta tekshirish yaxshi xizmat qilishi mumkin.

**real hayotda tushuntiring:**

Haqiqiy hayotda hujjatlarni dinamik yaratish turli stsenariylarda sodir bo'lishi mumkin. Mana bir nechta misollar:

Hisob-fakturalarni yaratish: Biznes hisob-fakturalarni dinamik ravishda yaratish uchun Apache POI bilan Spring ilovasidan foydalanishi mumkin. Ilova billing tizimidan ma'lumot olishi va har bir mijoz uchun hisob-fakturani, jumladan, mijozning ismi, manzili va qarz miqdori kabi ma'lumotlarni yaratishi mumkin.

Hisobotlarni yaratish: Spring ilovasi foydalanuvchidan kiritilgan ma'lumotlarga asoslangan hisobotlarni yaratishi mumkin. Masalan, kompaniya ma'lum vaqt davrlari, mintaqalar yoki mahsulotlar uchun savdo ma'lumotlarini ko'rsatadigan hisobotlarni yaratish uchun savdo hisoboti vositasidan foydalanishi mumkin.

Shartnomalarni yaratish: Yuridik sohada Spring ilovasi shartnomalarni dinamik ravishda yaratish uchun Apache POI-dan foydalanishi mumkin. Ariza advokat yoki mijozdan ma'lumot olishi va kelishilgan shartlar asosida maxsus shartnoma tuzishi mumkin.

Umuman olganda, hujjatlarni real hayotda dinamik ravishda yaratish biznes jarayonlarini soddalashtirish, vaqt va resurslarni tejashga yordam beradi. Bu maxsus hujjatlarni tez va samarali yaratishga imkon beradi va turli sohalarda va ilovalarda foydalanish mumkin.

Spring ilovalarida **Apache POI nima** uchun ishlatilishining ba'zi sabablari:

**Shaxsiy hujjatlarni yarating: Apache POI** yordamida siz foydalanuvchi kiritishiga asoslangan dinamik tarkibga ega maxsus Word hujjatlarini yaratishingiz mumkin. Bu tezkor hisobotlar, hisob-fakturalar yoki boshqa turdagi hujjatlarni yaratish kerak bo'lganda foydalidir.

Mavjud hujjatlarni o'qing va o'zgartiring: **Apache POI** mavjud **Word** hujjatlarini **o'qish** va **o'zgartirish** imkonini beradi. Bu hujjat mazmunini yangilash yoki undan ma'lumot olish kerak bo'lganda foydalidir.

O'zaro platformalar mosligi: **Apache POI** *Java kutubxonasidir*, ya'ni u Java-ni qo'llab-quvvatlaydigan har qanday platformada ishlashi mumkin. Bu Apache POI-dan foydalanadigan Spring ilovalarini turli muhitlarda joylashtirishni osonlashtiradi.

Open-source va bepul: **Apache POI** ochiq manbali kutubxona bo'lib, undan foydalanish bepul. Bu uni Spring ilovalarida hujjatlarni yaratish uchun tejamkor yechimga aylantiradi.

Umuman olganda, **Apache POI** Java-da **Microsoft Office** *hujjatlarini yaratish va boshqarish uchun kuchli vositadir*. Bu tezkor hujjatlarni yaratishi kerak bo'lgan spring ilovalari uchun mashhur tanlovdir.

Spring Boot-da Word hujjatini yaratish uchun siz quyidagi amallarni bajarishingiz mumkin:

1.Loyihangizga **Apache POI** qaramligini qo'shing: **Apache POI Microsoft Office** *hujjatlari bilan ishlash uchun Java kutubxonasi*. Loyihangizning **pom.xml** fayliga quyidagi bog'liqlikni qo'shishingiz mumkin:

**<dependency>**

**<groupId>org.apache.poi</groupId>**

**<artifactId>poi-ooxml</artifactId>**

**<version>4.1.2</version>**

**</dependency>**

2.Word hujjati shablonini yarating: Siz yangi hujjatlarni yaratish uchun asos bo'lib xizmat qiladigan shablon Word hujjatini yaratishingiz mumkin. Haqiqiy qiymatlar bilan almashtiriladigan dinamik ma'lumotlar uchun to'ldiruvchilarni qo'shishingiz mumkin.

Shablon hujjatini yuklang: Spring Boot ilovangizda shablon hujjatini FileInputStream sinfidan foydalanib yuklashingiz mumkin.

Albatta, men Spring Boot ilovasida Word faylini qanday yaratish bo'yicha bosqichma-bosqich ko'rsatma beraman.

1. Yangi Spring Boot ilovasini yarating:

Spring Initializr (https://start.spring.io/) yoki o'zingiz yoqtirgan IDE-dan foydalanib, yangi Spring Boot ilovasini yaratishingiz mumkin. Ushbu misolda siz Spring Initializr-dan foydalanyapsiz deb o'ylayman. Mana qadamlar:

https://start.spring.io/ saytiga o‘ting

Quyidagi variantlarni tanlang:

Loyiha: Maven loyihasi

Til: Java

Spring Boot: 2.6.3

Guruh: com.misol

Artefakt: so'z yaratuvchisi

Qadoqlash: banka

Java: 11

"Bog'liqliklarni qo'shish" tugmasini bosing va quyidagi bog'liqliklarni qo'shing:

Bahor veb

Apache POI

Loyihani ZIP fayl sifatida yuklab olish uchun "Yaratish" tugmasini bosing.

2. Word faylini yaratish uchun kod qo'shing:

Endi sizda yangi Spring Boot ilovasi mavjud bo'lib, Word faylini yaratish uchun kod qo'shishingiz mumkin. Buni qanday amalga oshirishga misol:

1.

**@RestController  
public class WordGeneratorController {  
 @GetMapping("/generate-word")  
 public String generateWord() {  
 XWPFDocument document = new XWPFDocument();  
 XWPFParagraph paragraph = document.createParagraph();  
 XWPFRun run = paragraph.createRun();  
 run.setText("Hello, World!");  
  
 try {  
 FileOutputStream out = new FileOutputStream("document.docx");  
 document.write(out);  
 out.close();  
 document.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 return "Error generating Word file";  
 }  
  
 return "Word file generated successfully";  
 }  
}**

**2.**

**@GetMapping("/generate-word")  
@ResponseBody  
public String generateWord() throws IOException {  
 *// Yangi Word hujjatini yaratish* XWPFDocument document = new XWPFDocument();  
  
 *// Yangi paragraf yaratish* XWPFParagraph paragraph = document.createParagraph();  
  
 *// Paragrafga matn qo'shish* XWPFRun run = paragraph.createRun();  
 run.setText("XWPFParagraph Apache POI kutubxonasi tomonidan taqdim etilgan va Apache POI tavsifli yozilgan Microsoft Word fayllarini boshqarish uchun mo'ljallangan");  
  
  
 *// Hujjatni diskda saqlash* FileOutputStream out = new FileOutputStream(new File("document.docx"));  
 document.write(out);  
 out.close();  
  
 return "Word hujjati yaratilgan va diskda saqlanadi.";  
}**

Ushbu kod **WordGeneratorController** klassini yagona metod **GenereWord**() bilan belgilaydi. **/generate-word** so'nggi nuqtasiga **HTTP GET** so'rovini yuborganingizda, metod yangi Word hujjatini yaratadi, "**Salom, dunyo!**" matni bilan paragraf qo'shadi. va uni loyiha katalogidagi **document.docx** nomli faylga saqlaydi.

3. Ilovani ishga tushiring va oxirgi nuqtani sinab ko'ring:

Spring Boot dasturini **WordGeneratorApplication** sinfining asosiy metodini bajarish orqali ishga tushirishingiz mumkin. Ilova ishga tushirilgandan so'ng, **curl** yoki **veb-brauzer** kabi vositadan foydalangan holda **/generate-word** so'nggi nuqtasiga **HTTP GET** so'rovini yuborishingiz mumkin. Misol uchun, agar siz ilovani **8080** **portida** local sifatida ishga tushirayotgan bo'lsangiz, **veb-brauzerni** ochib, **http://localhost:8080/generate-word** manziliga o'tishingiz mumkin. Agar hamma narsa to'g'ri ishlayotgan bo'lsa, brauzeringizda "**Word fayli muvaffaqiyatli** **yaratildi**" degan xabarni ko'rishingiz kerak va loyiha katalogida **document.docx** nomli yangi fayl yaratilishi kerak.

Bo'ldi shu! Siz hozirgina **Apache POI** kutubxonasidan foydalanib **Word faylini** **yaratuvchi Spring Boot ilovasini yaratdingiz**. **POI** kutubxonasi tomonidan taqdim etilgan turli klasslar va metodlardan foydalanib, **Word** faylining mazmuni va formatini kerakli tarzda sozlashingiz mumkin.

* **Menda "404 not found " xatolik chiqdi.**

Agar siz /generate-word so'nggi nuqtasiga kirishga urinayotganda "404 topilmadi" xatosiga duch kelsangiz, oxirgi nuqta Spring Boot ilovangizda to'g'ri sozlanmagan bo'lishi mumkin.

Muammoni hal qilishga urinib ko'rishingiz mumkin bo'lgan bir nechta narsalar:

1.Ilova loglarini tekshiring:

Spring Boot ilovasini ishga tushirganingizda, u ilova konteksti va ta'sir etilayotgan so'nggi nuqtalar haqidagi ma'lumotlarni jurnalga kiritishi kerak. **/generate-word** so'nggi nuqtasi bilan bog'liq muammoni ko'rsatishi mumkin bo'lgan har qanday xato xabarlari yoki ogohlantirishlarni qidiring.

2.Yakuniy nuqta mappingni tasdiqlang:

**WordGeneratorController** klassingizdagi @**GetMapping** izohi **/generate-word** so'nggi nuqtasini to'g'ri ko'rsatayotganiga ishonch hosil qiling. Oxirgi nuqta yo'lining imlo va formatlanishini ikki marta tekshiring.

3.Ilova portini tekshiring:

odatda, Spring Boot ilovalari **8080** **portda** **ishlaydi**. Ilovangiz to'g'ri portda ishlayotganiga va HTTP so'rovini to'g'ri portga yuborayotganingizga ishonch hosil qiling. Siz server.port xususiyatini **application.properties** yoki **application.yml** faylingizga qo'shish orqali ilova portini sozlashingiz mumkin.

4.Boshqa so'nggi nuqtani sinab ko'ring:

Agar **/generate-word** so'nggi nuqtasi hali ham ishlamasa, boshqa so'nggi nuqta (masalan, /salom) yaratib ko'ring va undan oddiy xabarni qaytaring. Bu sizning ilovangiz ishlayotganligini va oxirgi nuqtalar to'g'ri ko'rsatilayotganligini tekshirishga yordam beradi.

5.Loyiha tuzilishini tekshiring:

**WordGeneratorController** klassingiz to'g'ri paketda joylashganligiga va Spring Boot ilovasi tomonidan to'g'ri skanerdan o'tkazilayotganligiga ishonch hosil qiling. Odatiy bo'lib, Spring Boot ilovalari paketdagi komponentlarni (masalan, kontrollerlar) va uning @SpringBootApplication izohi belgilangan kichik paketlarini skanerlaydi.

Agar ushbu qadamlarning hech biri muammoni hal qilmasa, iltimos, ilovangiz sozlamalari va xato xabarlari yoki siz ko‘rayotgan jurnallar haqida batafsil ma’lumot bering.

**XWPFDocument** bu **Apache** **POI** kutubxonasidagi klass boʻlib, xotiradagi **Word** hujjatini ifodalaydi. Bu Java-da Word hujjatlarini dasturiy ravishda yaratish, o'qish va o'zgartirish usulini taqdim etadi.

**XWPFDocumentni** **Word** hujjatidagi barcha tarkib uchun konteyner sifatida tasavvur qiling. Hujjatga **paragraflar**, **jadvallar**, **rasmlar** va **boshqa elementlarni** **qo'shish**, **o'zgartirish** yoki **olib tashlash** imkonini beradi. Ushbu elementlarga formatlash uslublari va xususiyatlarini ham qo'llashingiz mumkin.

**XWPFDocument**-dan foydalanib, siz Java ilovangizda dinamik ravishda Word hujjatlarini yaratishingiz mumkin. Misol uchun, siz tezda yaratilishi kerak bo'lgan hisobot, xat yoki boshqa turdagi hujjatlarni yaratishingiz mumkin.

**XWPFParagraph** bu Word hujjatidagi paragrafni ifodalovchi **Apache POI** kutubxonasidagi klass. Abzats matnning boshqa bloklaridan qator uzilishi bilan ajratilgan matn blokidir. Paragrafni o'ziga xos formatlash uslublari va xususiyatlariga ega bo'lgan matn uchun konteyner sifatida tasavvur qiling.

**XWPFParagraph**-dan foydalanib, Word hujjatida yangi paragraflar yaratishingiz va ularga matn qo'shishingiz mumkin. Shuningdek, siz matnga **shrift o'lchami**, **shrift** **rangi**, **chekinish** va **tekislash** kabi formatlash uslublari va xususiyatlarini qo'llashingiz mumkin. Bir nechta matn bloklari bilan to'liq Word hujjatini yaratish uchun **XWPFDocument** ob'ektiga bir nechta paragraflar qo'shishingiz mumkin.

Misol uchun, agar siz sarlavha va ba'zi matnli Word hujjatini yaratmoqchi bo'lsangiz, sarlavha uchun **XWPFParagraph** ob'ektini va matn uchun boshqa **XWPFParagraph** ob'ektini yaratishingiz mumkin. Keyin ushbu paragraflarni **XWPFDocument** ob'ektiga qo'shishingiz va kerakli formatga ega to'liq hujjat yaratishingiz mumkin.

**ByteArrayOutputStream** - bu Java tilidagi klass bo'lib, **u xotiradagi bayt massiviga ma'lumotlarni yozish usulini ta'minlayd**i. Bu diskdagi fayl o'rniga xotiradagi buferga ma'lumotlarni yozish imkonini beradi. **ByteArrayOutputStream**-ga yozilgan ma'lumotlar bayt massivida saqlanadi.

**ByteArrayOutputStreamni** xotirada ikkilik ma'lumotlarni saqlaydigan konteyner sifatida tasavvur qiling. Unga **matn**, **tasvir**, **audio** yoki **boshqa har qanday ikkilik format** kabi har qanday turdagi ma'lumotlarni yozishingiz mumkin. **ByteArrayOutputStream**-ga ma'lumotlarni yozganingizdan so'ng, **toByteArray**() metodi yordamida ma'lumotlarni bayt massivi sifatida qaytarib olishingiz mumkin.

**ByteArrayOutputStream** ko'pincha ikkilik ma'lumotlarni tarmoq orqali yuborish, faylga saqlash yoki keyinchalik qayta ishlashdan oldin xotirada vaqtincha saqlash kerak bo'lganda ishlatiladi. Spring ilovasida Word hujjatini yaratish kontekstida siz ByteArrayOutputStream-dan hujjat mazmunini bayt massiviga yozish uchun foydalanishingiz mumkin va keyin bayt massivini foydalanuvchiga javob sifatida qaytarishingiz mumkin.

**FileOutputStream** Java tilidagi klass boʻlib, **diskdagi** **faylga maʼlumotlarni yozish usulini taʼminlaydi.** U fayl tizimida fayl yaratish yoki uning ustiga yozish va unga ma'lumotlarni yozish imkonini beradi.

**Disk**, hisoblash nuqtai nazaridan, raqamli ma'lumotlarni doimiy ravishda saqlash uchun ishlatiladigan jismoniy qurilmaga ishora qiladi. Bu ma'lumotlar va dasturiy ilovalar uchun uzoq muddatli saqlashni ta'minlaydigan ichki yoki tashqi apparat komponenti bo'lishi mumkin. Disklar qattiq disklar (**HDD**), qattiq disklar (SSD**),** floppi disklar va **CD/DVDlar** kabi turli shakllarda bo'lishi mumkin.

Ma'lumotlar diskka yozilsa, u qurilma ichidagi bir yoki bir nechta platalarda saqlanadi, ularga keyinroq kirish va o'qish mumkin. Ma'lumotlar ikkilik raqamlar ko'rinishida saqlanadi, ular kompyuterning disk boshqaruvchisi tomonidan o'qilishi va yozilishi mumkin.

Disklar zamonaviy hisoblash tizimlarining muhim tarkibiy qismidir, chunki ular katta hajmdagi ma'lumotlarni doimiy saqlashning ishonchli usulini ta'minlaydi. Ular operatsion tizimlarni, dasturiy ilovalarni, media fayllarni, hujjatlarni va keyinchalik kirish kerak bo'lgan boshqa turdagi raqamli tarkibni saqlash uchun ishlatiladi.

**FileOutputStream**-ni kompyuteringizning qattiq diskidagi yoki xotira qurilmasidagi faylga ma'lumotlarni yozish vositasi sifatida tasavvur qiling. Siz uni faylga **matn**, **tasvir** yoki **boshqa har qanday ikkilik format kabi istalgan turdagi maʼlumotlarni yozish uchun ishlatishingiz mumkin**. Faylga ma'lumot yozganingizdan so'ng, siz uni o'chirmaguningizcha yoki yangi ma'lumotlar bilan qayta yozmaguningizcha u diskda saqlanadi.

**FileOutputStream** *ko'pincha ma'lumotlarni diskda doimiy saqlash kerak bo'lganda, masalan, dasturingizning natijasini saqlashingiz yoki tasvir yoki hujjatlar kabi foydalanuvchi tomonidan yaratilgan tarkibni saqlashingiz kerak bo'lganda ishlatiladi.* Spring ilovasida Word hujjatini yaratish kontekstida siz hujjat mazmunini diskdagi faylga yozish uchun FileOutputStream-dan foydalanishingiz mumkin va keyin foydalanuvchiga fayl yo'li yoki mazmuni bilan javob qaytarishingiz mumkin.

-------------------------------------------------------------------------------------------------------------

1.Pom.xml faylingizga Apache POI bog'liqliklarini qo'shing:

**<dependency>**

**<groupId>org.apache.poi</groupId>**

**<artifactId>poi</artifactId>**

**<version>5.1.0</version>**

**</dependency>**

**<dependency>**

**<groupId>org.apache.poi</groupId>**

**<artifactId>poi-ooxml</artifactId>**

**<version>5.1.0</version>**

**</dependency>**

2. Word hujjatini yaratuvchi va uni diskda saqlaydigan boshqaruvchi klassida yangi metod yarating:

**@Controller  
public class WordController {**

**@GetMapping("/generate-word")  
public ResponseEntity<byte[]> generateWordDocument()**

**throwsIOException {  
  
 *// Yangi Word hujjatini yarating* XWPFDocument document = new XWPFDocument();  
  
 *// Yangi paragraf yarating va ishga tushiring* XWPFParagraph paragraph = document.createParagraph();  
 XWPFRun run = paragraph.createRun();  
  
 *// Ishga bir oz matn qo'shing* run.setText("Hello, World!");  
  
 *// Hujjatni ByteArrayOutputStream-ga saqlang* ByteArrayOutputStream baos = new ByteArrayOutputStream();  
 document.write(baos);  
  
 *// ByteArrayOutputStreamni bayt massiviga aylantiring* byte[] bytes = baos.toByteArray();  
*// Yangi HttpHeaders ob'ektini yarating va kontent turi va joylashuvini o'rnating* HttpHeaders headers = new HttpHeaders();  
 headers.setContentType(MediaType.APPLICATION\_OCTET\_STREAM);**

**headers.setContentDispositionFormData("attachment", "document.docx");  
  
*// Bayt massivi va HttpHeaders bilan ResponseEntity ni qaytaring* return ResponseEntity.ok()  
 .headers(headers)  
 .body(bytes);  
 }  
}**

3.**generatorWordDocument**() metodi chaqirilganda, u yangi **XWPFDocument** ob'ektini **yaratadi** va unga **XWPFRun** ob'ekti yordamida biror matn qo'shadi. *Hujjat keyinchalik bayt massiviga aylantiriladigan* **ByteArrayOutputStreamga** **saqlanadi**. Nihoyat, **ResponseEntity** ob'ekti javobning kontent turi va joylashuvini o'rnatish uchun bayt massivi va ba'zi **HttpHeaders** bilan qaytariladi.

4. **/generate-word** so'nggi nuqtasiga **GET** so'rovi yuborilganda, **generatorWordDocument**() metodi chaqiriladi va Word hujjati **document.docx** nomli fayl qo'shimchasi sifatida qaytariladi.

Ushbu misolda Word hujjati xotiraga **ByteArrayOutputStream** sifatida saqlanganligini unutmang. Hujjatni diskda saqlamoqchi bo'lsangiz, **ByteArrayOutputStream** o'rniga FileOutputStream obyektidan foydalanishingiz va **FileOutputStream**.**write**() metodi yordamida hujjatni diskdagi faylga yozishingiz mumkin.

**Spring bootd agi ObjectNode funksiyasi nimadan iborat?**

Spring Boot-da **ObjectNode** - bu **Jekson** kutubxonasi orqali **JSON** obyektlarini yaratish uchun mo'ljallangan Java klassi. **ObjectNode** klassi boshqa **JSON** **obyektlarini yaratish**, **tahrirlash** va **ular bilan** **ishlash** imkoniyatini beradi.

**ObjectNode** Java kodidagi **JSON** ob'ektlariga oson kirishni ta'minlaydi, bu **JSONni** *Java ob'ektlariga aylantirishni* va J*ava ob'ektlarida* **JSON** *bo'lishini osonlashtiradi.* Bu sizga Spring Boot-da **JSON**-ni qulay boshqarish imkonini beradi, masalan, **RESTful** xizmatlarida yoki fayllarni yashirish va yuklash va boshqa imkoniyatlar.

Masalan, Spring Boot-da **ObjectNode**-dan shunday foydalanishingiz mumkin:

**ObjectMapper objectMapper = new ObjectMapper();  
ObjectNode personNode = objectMapper.createObjectNode();  
  
personNode.put("name","John");  
personNode.put("age",30);  
  
ObjectNode addressNode = objectMapper.createObjectNode();  
addressNode.put("city","New York");  
addressNode.put("country","USA");  
  
personNode.set("address",addressNode);  
  
System.*out*.println(personNode.toString());**

Bu kod **JSON** obyektini yaratadi, "**name**" va "**age**" nomli maydonlar qo'shadi va ularning qiymatlarini qo'yadi. Keyin siz ushbu **JSON** obyektini o'zgartirishingiz, uni boshqa **JSON** obyektiga o'girishingiz yoki **JSON**-fayllarga **saqlashingiz** mumkin.

Ushbu koddagi **ObjectMapper** classi, **JSON** ma'lumotlarini **o'qish** va **yozish** uchun kerakli funksiyalarni taqdim etadi. **createObjectNode** metodi, **JSON** obyektini yaratish uchun ishlatiladi. **put** metodi, **JSON** obyektiga *yangi xususiyatlar qo'shish uchun foydalaniladi*. **set** metodi esa, *bir JSON obyektini boshqa bir JSON obyektiga o'rnatish uchun ishlatiladi*.

Yuqoridagi kod natijasida quyidagi **JSON** ma'lumotlari chiqadi:

**{  
"name": "John",  
"age": 30,  
"address": {  
"city": "New York",  
"country": "USA"  
}  
}**

**APACHE POI ga oid eng ko'p ishlatiladigan masalalar**

APACHE POI Java dasturlash tilida Excel, Word, va PowerPoint hujjatlari bilan ishlash uchun yordam beradigan bir kutubxonadir. Bu kutubxona yordamida siz Excel fayllarini yoki Word hujjatlari yaratishingiz, o'qishingiz, tahrir qilishingiz va saqlashingiz mumkin. APACHE POI ko'plab ishlar uchun qo'llaniladi. Masalan:

**1.Excel fayllarini o'qish va yozish.**

**2.Excel-da qator va ustunlarni ochish, qisqartirish va boshqa tuzatishlar.**

**3.Excel-da xaridorlar to'plamiga qarab hisob kitoblarni yaratish.**

**4.Excel-da ko'p jadvallarni biriktirish.**

**5.Excel-da funktsiyalar va hisob kitoblari yaratish.**

**6.Word hujjatlari ochish, o'qish, va yozish.**

**7.PowerPoint presentatsiyalarni yaratish, o'qish, va yozish.**

**APACHE POI ko'pincha kompaniyalarda, huquqiy yuridik shaxslarda, va boshqa sohalarda ishlatiladi, jumladan, ma'lumotlar bazalarini yaratish, ma'lumotlar bo'yicha hisobotlar tayyorlash, va hujjatlarni to'plam qilish uchun.**

**1.Excel fayllarini o'qish va yozish.**

Excel faylidan **ma'lumotlarni o'qish** va **yozish** uchun, **APACHE POI** kutubxonasidan foydalanish mumkin. Quyidagi Java kodi Excel faylidan malumot oqish uchun misol ko'rsatadi:

**public class ExcelExample {  
 public static void main(String[] args) {  
 try {  
 *// Excel faylni ochish***

**FileInputStream file=new FileInputStream(new File("test.xlsx"));  
  
 *// Workbook yaratish* Workbook workbook = new XSSFWorkbook(file);  
  
 *// Hujjatni tanlash* Sheet sheet = workbook.getSheetAt(0);  
  
 *// Qatorni tanlash* Row row = sheet.getRow(0);  
  
 *// Qatorni tanlash* Cell cell = row.getCell(0);  
  
 *// Qator va qatorni chiqarish* System.*out*.println(cell.getStringCellValue());  
  
 *// Ma'lumotlarni yozish* row.createCell(1).setCellValue("Hello");  
 file.close();  
  
 *// Yangi ma'lumotlarni saqlash* FileOutputStream outFile =new FileOutputStream(new File("test.xlsx"));  
 workbook.write(outFile);  
 outFile.close();  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}**

**FileInputStream** Java dasturlash tilida **fayllarni o'qish** uchun foydalaniladigan klassdir. Ushbu klass faylni ochib, **ma'lumotlarni o'qish** uchun yordam beradi.

Quyidagi kodda **FileInputStream** yordamida fayl ochiladi:

**FileInputStream file = new FileInputStream("file.txt");**

Ushbu kodda "**file.txt**" nomli fayl ochiladi va **FileInputStream** klassidan obyekt yaratiladi.

**FileInputStream** yordamida o'qish uchun esa quyidagi kodlar ishlatiladi:

**int i;  
 while ((i = file.read()) != -1) {  
 System.*out*.print((char) i);  
 }**

Ushbu kod faylning **barcha ma'lumotlarini o'qib chiqaradi**. Faylning oxiriga kelganda **file.read()** metodi -1 qiymatini qaytaradi va tsiklni to'xtatadi.

**FileInputStream** obyekti ishini tugatish uchun **close()** metodidan foydalanish kerakdir:

**file.close();**

**FileInputStream** va uning yordamida ochilgan fayl hammasi o'zgartirish uchun ochiq bo'lishidan keyin yopilishi kerakdir.

**Workbook**, **APACHE POI** kutubxonasining bir qismidir va **Excel** hujjatlari **yaratish**, **o'qish** va **yozish** uchun yordam beradi. **Workbook**, bir yoki bir nechta Excel fayllarini o'z ichiga oladi va ularni hujjat obyektiga aylantiradi. **Workbook**, **bir** yoki **bir nechta Excel** hujjatlari va shularning har biri uchun **ro'yxatni saqlash** uchun mo'ljallangan ma'lumotlar bazasini (spreadsheet-elektron jadval) yaratish uchun yordam beradi.

**APACHE POI kutubxonasida ikki turdagi Workbook mavjud:**

**XSSFWorkbook**: **Bu klass Excel fayllari uchun 2007 yilga qadar (.xlsx) va yangi formatlarni qo'llab-quvvatlaydi.**

**HSSFWorkbook**: **Bu klass esa Excel fayllari uchun 2003 yildan oldingi formatlarni (.xls) qo'llab-quvvatlaydi.**

**Sheet** **APACHE POI** kutubxonasining bir qismidir va **Excel hujjatlari** **yaratish**, **o'qish** va **yozish** uchun yordam beradi. **Sheet** bir hujjatning bitta ro'yxati, yoki "**sheet**"dir. **Excel hujjatida bir nechta sheet bo'lishi mumkin, va har bir sheet o'ziga xos nomi bilan ajratiladi.**

APACHE POI kutubxonasi ikki turdagi Sheet mavjud:

**XSSFSheet**: **Bu klass Excel fayllari uchun 2007 yilga qadar (.xlsx) va yangi formatlarni qo'llab-quvvatlaydi.**

**HSSFSheet**: **Bu klass esa Excel fayllari uchun 2003 yildan oldingi formatlarni (.xls) qo'llab-quvvatlaydi.**

**Cell**, Excel hujjatida **qator** va **ustunlarning** kesishmasidagi bir o'zgaruvchidir. Bitta qatordagi ma'lum bir ustunni yoki qator va ustunning kesishmasidagi maydonchani tashkil etadi. Bu maydon turli xil turlarga ega bo'lishi mumkin, masalan, **raqam**, **matn**, **sanani** ko'rsatish va **hokazo**.

**APACHE POI**-da **Cell** ni yaratish uchun, yaratilgan qatordan foydalanish kerak. Masalan, quyidagi kod yaratilgan qatordan **Cellni** yaratadi va "Hello, World!" qiymatini qo'yadi:

**Row row = sheet.createRow(0);  
Cell cell = row.createCell(0);  
cell.setCellValue("Hello, World!");**

Bu kod **qatorni yaratadi**, 0-qiymatli ustunni tanlaydi va unga "**Hello, World**!" qiymatini qo'yadi. 0-qiymatli ustunga mos keluvchi **Cell** yaratiladi va uning qiymati ham "**Hello, World**!" qiymatiga tenglanadi.

Cellga qiymat berish uchun **setCellValue**() metodidan foydalaniladi. Qiymat turi, yo'qsa, o'zgaruvchining turiga mos kelishi kerak.

**setFontHeightInPoints qanday vazifani bajaradi?**

**setFontHeightInPoints** metodi **APACHE POI**-da **Fontni yaratish** va **tahrirlash** uchun ishlatiladi. Ushbu metoddan foydalanib **Fontning o'lchamini belgilaymiz.**

Font o'lchami, tipik ravishda, "**point**" deb ataluvchi o'lchov birligidir. **Bir "point" 1/72 dyumga teng**. **Ya'ni, "12 point"lik Font "1/6 dyum" yoki "16 pixel" bo'ladi.**

Shu sababli, quyidagi kod "**Calibri**" nomli Fontni yaratadi va o'lchamini 14 pointga o'zgartiradi:

**Workbook workbook = new XSSFWorkbook();  
Font font = workbook.createFont();  
font.setFontName("Calibri");  
font.setFontHeightInPoints((short) 14);**

**HorizontalAlignment** - bu **APACHE POI**-da ishlatiladigan bir **enum** (**katta harflardan iborat o'zgaruvchi nomi**)dir. Ushbu **enum** yordamida Excel-cellelarning matnni qayerga joylashtirish kerakligini aniqlaymiz.

**HorizontalAlignment** ning uchta qiymati mavjud: **LEFT**, **CENTER**, va **RIGHT**. Shu sababli, quyidagi kod **CENTER** qilish uchun yaratilgan **cellStyle** yaratadi:

**Workbook workbook = new XSSFWorkbook();  
CellStyle cellStyle = workbook.createCellStyle();  
cellStyle.setAlignment(HorizontalAlignment.CENTER);**

**2.Excel-da qator va ustunlarni ochish, qisqartirish va boshqa tuzatishlar.**

***//Excel-da qator va ustunlarni ochish, qisqartirish va boshqa tuzatishlar.*public class ExcelExample2 {  
 public static void main(String[] args) throws IOException {  
  
 *// Yangi Workbook yaratish* Workbook workbook = new XSSFWorkbook();  
  
 *// Yangi Sheet-varaq yaratish* Sheet sheet = workbook.createSheet("MySheet");  
  
 *// Qator va ustunlarni yaratish* Row row = sheet.createRow(0);  
 Cell cell1 = row.createCell(0);  
 Cell cell2 = row.createCell(1);  
 cell1.setCellValue("Hello");  
 cell2.setCellValue("World");  
  
 *// Qatorlarni qisqartirish* CellStyle style = workbook.createCellStyle();  
 Font font = workbook.createFont();  
 font.setBold(true);  
 font.setFontHeightInPoints((short) 14);  
 style.setFont(font);  
 row.setRowStyle(style);  
  
 *// Ustunlarni qisqartirish* sheet.autoSizeColumn(0);  
 sheet.autoSizeColumn(1);  
 CellStyle headerStyle = workbook.createCellStyle();  
 headerStyle.setAlignment(HorizontalAlignment.*CENTER*);  
 Row headerRow = sheet.createRow(0);  
 Cell headerCell1 = headerRow.createCell(0);  
 Cell headerCell2 = headerRow.createCell(1);  
 headerCell1.setCellValue("Column 1");  
 headerCell2.setCellValue("Column 2");  
 headerCell1.setCellStyle(headerStyle);  
 headerCell2.setCellStyle(headerStyle);  
  
 *// Hujjatni saqlash* FileOutputStream fileOut = new FileOutputStream("workbook.xlsx");  
 workbook.write(fileOut);  
 fileOut.close();  
  
 *// Workbookni yopish* workbook.close();  
 }  
}**

Bu kod yangi Excel hujjat yaratadi va unda "**MySheet**" nomli Varaq yaratadi. Ushbu sheet'da 0-qiymatli qator yaratiladi va uning 0 va 1-qiymatli ustunlari "**Hello**" va "**World**" so'zlariga mos keladi. **Qator va ustunlarni qisqartirish** uchun **CellStyle** yaratiladi, font yaratiladi va qatorga ushbu stil qo'yiladi. Sheet ustunlarini qisqartirish uchun autoSizeColumn metodi ishlatiladi. Sheetga ustunlar uchun headerRow yaratiladi va uning 0 va 1-qiymatli ustunlariga "Column 1" va "Column 2" qiymatlari mos keladi. Ustunlar ustiga "HorizontalAlignment.CENTER" usuli orqali qisqaqtirish qo'yiladi. Keyinchalik hujjat "workbook.xlsx" nomi bilan saqlanadi.

**3.Excel-da xaridorlar to'plamiga qarab hisob kitoblarni yaratish**

Xaridorlar to'plamiga qarab hisob kitoblarni yaratish uchun **APACHE POI** kutubxonasisidan foydalanish mumkin. Bu kutubxonasi yordamida Java dasturlari o'qish, yozish va tahlil qilish uchun Excel fayllariga kirish imkoniyatiga ega bo'ladi.

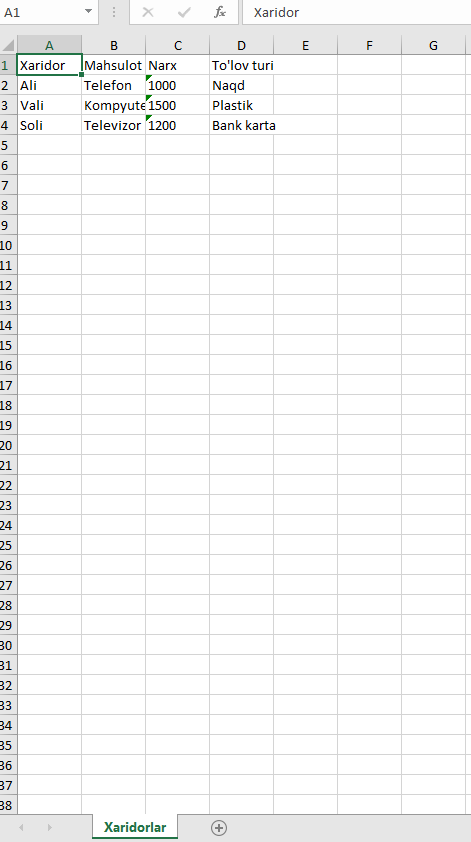
Quyidagi kodlar orqali Java dasturida Excel faylini yaratish va ma'lumotlarni kiritish mumkin:

***//Excel-da xaridorlar to'plamiga qarab hisob kitoblarni yaratish*public class ExcelFile {  
 public static void main(String[] args) {  
 *// Excel faylini yaratish* XSSFWorkbook workbook = new XSSFWorkbook();  
  
 *// Yangi Excel jadvali yaratish* XSSFSheet sheet = workbook.createSheet("Xaridorlar");  
  
 *// Ma'lumotlarni kiritish* Object[][] xaridorlar = {  
 {"Xaridor", "Mahsulot", "Narx", "To'lov turi"},  
 {"Ali", "Telefon", "1000", "Naqd"},  
 {"Vali", "Kompyuter", "1500", "Plastik"},  
 {"Soli", "Televizor", "1200", "Bank karta"},  
 };**

**//** **Ma'lumotlarni jadvalga kiritish:  
 int rowNum = 0;  
 System.*out*.println("Excel fayliga yozish boshlandi...");  
 for (Object[] xaridor : xaridorlar) {  
 *// Yangi qator yaratish* Row row = sheet.createRow(rowNum++);  
 int colNum = 0;  
 for (Object field : xaridor) {  
 *// Qatorga ma'lumotlarni kiritish* Cell cell = row.createCell(colNum++);  
 if (field instanceof String) {  
 cell.setCellValue((String) field);  
 } else if (field instanceof Integer) {  
 cell.setCellValue((Integer) field);  
 }  
 }  
 }  
  
 try {  
 *// Excel faylini saqlash* FileOutputStream outputStream = new FileOutputStream("Xaridorlar.xlsx");  
 workbook.write(outputStream);  
 workbook.close();  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 System.*out*.println("Excel fayliga yozish yakunlandi!");  
 }  
}**

Ushbu kod yordamida Xaridorlar.xlsx nomli Excel fayli yaratiladi va ma'lumotlar kiritiladi. Faylni saqlash yordamida, fayl diskga saqlanadi va Excel dasturida ochilishi mumkin.

**Natija:**



**4.Excel-da ko'p jadvallarni biriktirish.**

***//Excel-da ko'p jadvallarni biriktirish.*public class MultipleSheetsExample {  
 public static void main(String[] args) throws IOException {  
 Workbook workbook = WorkbookFactory.*create*(true);  
 Sheet sheet1 = workbook.createSheet("Sheet1");  
 Sheet sheet2 = workbook.createSheet("Sheet2");  
  
 *// Sheet1-da ma'lumotlarni kiritish* Row row1 = sheet1.createRow(0);  
 Cell cellA1 = row1.createCell(0);  
 cellA1.setCellValue("Sheet1dagi ma'lumotlar");  
 Cell cellB1 = row1.createCell(1);  
 cellB1.setCellValue("Sana");  
 Cell cellC1 = row1.createCell(2);  
 cellC1.setCellValue("Narx");  
  
 Row row2 = sheet1.createRow(1);  
 Cell cellA2 = row2.createCell(0);  
 cellA2.setCellValue("1");  
 Cell cellB2 = row2.createCell(1);  
 cellB2.setCellValue("01.01.2022");  
 Cell cellC2 = row2.createCell(2);  
 cellC2.setCellValue("1000");  
  
 Row row3 = sheet1.createRow(2);  
 Cell cellA3 = row3.createCell(0);  
 cellA3.setCellValue("2");  
 Cell cellB3 = row3.createCell(1);  
 cellB3.setCellValue("02.01.2022");  
 Cell cellC3 = row3.createCell(2);  
 cellC3.setCellValue("1500");  
  
 *// Sheet2-da ma'lumotlarni kiritish* Row row4 = sheet2.createRow(0);  
 Cell cellA4 = row4.createCell(0);  
 cellA4.setCellValue("Sheet2dagi ma'lumotlar");  
 Cell cellB4 = row4.createCell(1);  
 cellB4.setCellValue("Ism");  
 Cell cellC4 = row4.createCell(2);  
 cellC4.setCellValue("Yoshi");  
  
 Row row5 = sheet2.createRow(1);  
 Cell cellA5 = row5.createCell(0);  
 cellA5.setCellValue("1");  
 Cell cellB5 = row5.createCell(1);  
 cellB5.setCellValue("Ali");  
 Cell cellC5 = row5.createCell(2);  
 cellC5.setCellValue("30");  
  
 Row row6 = sheet2.createRow(2);  
 Cell cellA6 = row6.createCell(0);  
 cellA6.setCellValue("2");  
 Cell cellB6 = row6.createCell(1);  
 cellB6.setCellValue("Vali");  
 Cell cellC6 = row6.createCell(2);  
 cellC6.setCellValue("25");  
  
 *// Sheet1 va Sheet2 sarlavhalarini biriktirish* CellRangeAddress mergedRegion1 = new CellRangeAddress(0, 0, 0, 2);  
 sheet1.addMergedRegion(mergedRegion1);  
 *// Faylni saqlash* FileOutputStream fileOut = new FileOutputStream("ko'p\_jadval.xlsx");  
 workbook.write(fileOut);  
 fileOut.close();  
  
 System.*out*.println("Fayl muvaffaqiyatli yaratildi.");  
 }  
  
}**

**WorkbookFactory** **Apache POI** kutubxonasidagi klass hisoblanadi, bu Microsoft Office hujjatlarni boshqarish uchun Java API laridan tashkil topgan. Ma'lumki, **WorkbookFactory** Excel jadvalini ifodalaydigan Workbook klassining obyektini yaratish uchun ishlatiladi.

**WorkbookFactorydan** foydalanishning afzali shundaki, u foydalanuvchining fayl turi ni aniqlashga zarur bo'lmagan xolda .**xls** va .**xlsx** fayllarni o'z ichiga oladi. Bu, turli xil Excel fayllari bilan ishlash uchun kod yozishni osonlashtiradi.

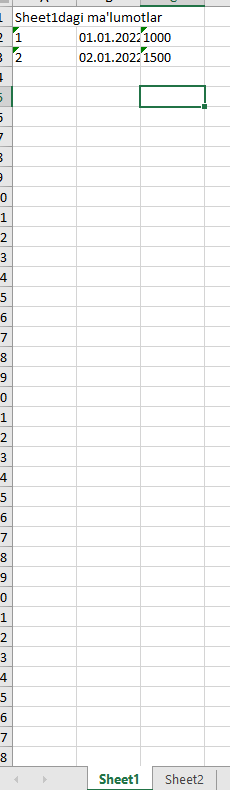
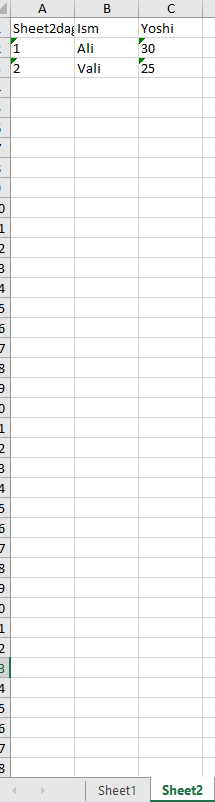
**WorkbookFactory** yordamida Workbook obyektini yaratish uchun odatda **InputStream** obyektini **create**() metodiga uzatishingiz kerak, bu metod faylning tarkibini ko'chirib olib, Workbook obyektini yaratadi. Quyidagi misolda bu haqda ko'proq ma'lumot olishingiz mumkin:

**CellRangeAddress**, Apache POI kutubxonasidagi bir class va Excel jadvalidagi bir nechta maydonlar (cell) diapazonini belgilaydi. Bu class, Excel jadvalidagi bir nechta maydonlarga (cell) bir nechta qator va ustunlardan (**row** va **column**) iborat diapazonlarni belgilashda foydalaniladi.

**CellRangeAddress** classiga ularning bosh va oxirgi koordinatalarini, yani boshlang'ich va oxirgi qator va ustun indekslarini uzatish orqali yaratiladi.

Misol uchun, agar biz **A1** tomonidan **B3** gacha bo'lgan diapazonni belgilamoqchi bo'lsak, quyidagi kodni ishlatishimiz mumkin:

**public class Example {  
 public static void main(String[] args) {  
 CellRangeAddress range = CellRangeAddress.valueOf("A1:B3");  
 *// yoki CellRangeAddress range = new CellRangeAddress(0, 2, 0, 1);* System.*out*.println("Birinchi qator indeksi: " + range.getFirstRow());  
 System.*out*.println("Oxirgi qator indeksi: " + range.getLastRow());  
 System.*out*.println("Birinchi ustun indeksi: " + range.getFirstColumn());  
 System.*out*.println("Oxirgi ustun indeksi: " + range.getLastColumn());  
 }  
}**

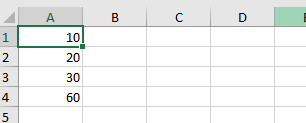
**5.Excel-da funktsiyalar va hisob kitoblari yaratish**

Apache POI kutubxonasi Excel elektron jadval funksiyalari va hisob-kitoblarini yaratish qobiliyatiga ega. Buning yordamida siz odatda jadvaldagi maydonlarda ishlaydigan Excel funksiyalaridan foydalanishingiz mumkin.

Funktsiyalardan foydalanish uchun Apache POI kutubxonasidagi **org.apache.poi.ss.usermodel.formula** paketida mavjud bo'lgan **FormulaEvaluator** classidan foydalanishingiz kerak. **FormulaEvaluator** obyekti maydon qiymatlarini aniqlashtirish va funksiya qiymatlarini hisoblash uchun ishlatiladi.

Quyidagi kodda yana bir jadval tuziladi va A1, A2 va A3 maydonlari har bir raqam bilan kiritiladi. A4 maydoniga **SUM** funksiyasidan foydalanib, A1, A2 va A3 qiymatlarining yig'indisini hisoblaydigan formula yozamiz. Keyinchalik, A4 maydonining qiymatini hisoblash uchun FormulaEvaluator classini chaqiramiz:

***//Excel-da funktsiyalar va hisob kitoblari yaratish*public class ExcelExample3 {  
 public static void main(String[] args) throws IOException {  
 Workbook workbook = new XSSFWorkbook();  
 Sheet sheet = workbook.createSheet("My Sheet");  
 Row row1 = sheet.createRow(0);  
 Cell cell1 = row1.createCell(0);  
 cell1.setCellValue(10);  
 Row row2 = sheet.createRow(1);  
 Cell cell2 = row2.createCell(0);  
 cell2.setCellValue(20);  
 Row row3 = sheet.createRow(2);  
 Cell cell3 = row3.createCell(0);  
 cell3.setCellValue(30);  
 Row row4 = sheet.createRow(3);  
 Cell cell4 = row4.createCell(0);  
 cell4.setCellFormula("SUM(A1:A3)");  
  
 *// Formulani baholang* FormulaEvaluator evaluator = workbook.getCreationHelper().createFormulaEvaluator();  
 evaluator.evaluateFormulaCell(cell4);  
  
 *// Write the output to a file-Chiqishni faylga yozing* FileOutputStream fileOut = new FileOutputStream("workbook.xlsx");  
 workbook.write(fileOut);  
 fileOut.close();  
  
 *// workbookni yoping* workbook.close();  
 }  
}**

****

**FormulaEvaluator**, **Apache POI kutubxonasida Excel jadvalidagi formulalar va funksiyalar uchun hisob-kitob yoki qiymatlarni hisoblash uchun ishlatiladigan classdir.**

**FormulaEvaluator** classi, Excel jadvalidagi har bir maydon uchun formulani hisoblash uchun yordam beradi. Bu class jadvalidagi har bir maydon uchun **funksiyalar**, **qiymatlar** va **hisob-kitoblarni** **hisoblash** uchun integratsiyalashtirilgan hisob-kitob jarayonini ishga tushiradi.

**FormulaEvaluator** classining yordamidagi **evaluateFormulaCell**(**Cell cell**) orqali, bir maydonning formulasi hisoblanadi va maydon qiymati o'zgartiriladi. Masalan, bir maydonni qiymatini hisoblash uchun **"SUM(A1:A3)"** formulani yozishimiz mumkin. Bu formulani hisoblash uchun **FormulaEvaluator** **evaluateFormulaCell** yordamiga uzatilgan maydonni o'zlashtiradi va yangi qiymatni yozadi.

**getCreationHelper** bu metod Workbook obyektini yaratishga mo'ljallangan yordamchi obyektni olish uchun ishlatiladi, shuningdek, yaratilgan Workbook obyektida **formulalar** va **shakllar** **kiritish** uchun kerak bo'lgan yordamchi obyektlarni ham yaratish uchun foydalaniladi.

**6.Word hujjatlari ochish, o'qish, va yozish**

**Apache POI** kutubxonasini foydalanib, Word hujjatlari bilan ishlash mumkin emas. Apache POI, faqatgina Microsoft Office formatida bo'lgan Excel hujjatlari bilan ishlashni o'rganishga mo'ljallangan.

Agar siz Microsoft Word hujjatlari bilan ishlashni xohlasangiz, Apache POI kutubxonasining o'rniga **Apache POI XML** kutubxonasidan foydalanishingiz mumkin. **Apache POI XML**, **OpenXML** formatida bo'lgan Microsoft Office hujjatlari bilan ishlashni o'rganishga mo'ljallangan.

Lekin, Apache POI kutubxonasidan foydalanish orqali, Microsoft Word hujjatlari ochish, o'qish va yozish uchun so'zlar kutubxonasi yoki boshqa tegishli kutubxona yordamida ishlatilishi kerak. Misol uchun, Apache POI kutubxonasidan foydalanish orqali, Microsoft Word hujjatini PDF formatida eksport qilish mumkin.

**7.PowerPoint presentatsiyalarni yaratish, o'qish, va yozish**

**PowerPoint** presentatsiyalarni yaratish, o'qish va yozish uchun Apache POI kutubxonasidan foydalanishingiz mumkin. Bu kutubxona, Microsoft Office PowerPoint formatida bo'lgan hujjatlarni o'qish va yozishni osonlashtiradi.

Quyidagi misol kod orqali, Apache POI kutubxonasidan foydalanib, PowerPoint presentatsiyasini yaratish, o'qish va yozish mumkin:

**public class PowerPointExample {  
 public static void main(String[] args) {  
 *// PowerPoint presentatsiyasini yaratish* XMLSlideShow ppt = new XMLSlideShow();  
  
 *// Yangi slide yaratish* XSLFSlide slide = ppt.createSlide();  
  
 *// Yangi paragraf yaratish va matn yozish* XSLFTextShape title = slide.createTextBox();  
 title.setText("Assalomu alaykum, dunyo!");  
  
 *// PowerPoint presentatsiyasini saqlash* try (FileOutputStream out = new FileOutputStream("presentation.pptx")) {  
 ppt.write(out);  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 *// PowerPoint presentatsiyasini o'qish* try (FileInputStream fileInputStream = new FileInputStream("presentation.pptx")) {  
 XMLSlideShow readPpt = new XMLSlideShow(fileInputStream);  
 XSLFSlide readSlide = readPpt.getSlides().get(0);  
 XSLFTextShape readTitle = (XSLFTextShape) readSlide.getShapes().get(0);  
 String text = readTitle.getText();  
 System.*out*.println(text);  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 *// PowerPoint presentatsiyasiga yangi slide qo'shish* XSLFSlide newSlide = ppt.createSlide();  
 XSLFTextShape newTitle = newSlide.createTextBox();  
 newTitle.setText("Bugungi kunda havo yaxshi!");  
  
 *// PowerPoint presentatsiyasini saqlash* try (FileOutputStream out = new FileOutputStream("presentation.pptx")) {  
 ppt.write(out);  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}**

Ushbu kod **XMLSlideShow** *classi yordamida yangi PowerPoint presentatsiyasi yaratishni,* **createSlide***() metodi yordamida yangi slayd yaratishni va* **createTextBox***() metodi yordamida yangi matn maydonchasi yaratishni o'rganadi.* **setText***() metodi yordamida matn yoziladi.*

PowerPoint presentatsiyasini saqlash uchun **write**() metodi va **FileOutputStream** yordamchi classi ishlatiladi. PowerPoint presentatsiyasini o'qish uchun esa **XMLSlideShow** classi yordamida **getSlides**() va **getShapes**() metodi ishlatiladi.

Bu misol kod poi-ooxml va poi-ooxml-schemas kutubxonalarini kerakli xolda qo'shishni talab qiladi.

Apache Kafka, tizimlar orasida xabarlar vaqtli tarzda almashishni ta'minlaydigan yuqori ko'rsatkichli bir distributiv ma'lumotlar bazasi (Distributed Data Streaming Platform) hisoblanadi.

Kafka asosan, ko'p holatda ma'lumotlar almashishida ishlatiladi, masalan, IoT (Internet of Things) tizimlarida, veb tizimlarda, qidiruv tizimlarida, qo'shimcha tahlil loyiha uchun ma'lumotlarni o'qish va boshqalarda. Ushbu xususiyatlar tufayli, Kafka engil ma'lumotlar almashishiga mos keladi:

Xavfsiz va yuqori ko'rsatkichli ma'lumot almashishni ta'minlash;

Barcha tizimlarda ishlatilishi mumkin bo'lgan qulay API;

To'liq ruxsatnoma asosida ochiq-kodli dasturlarni ishga tushirish imkoniyatini beradi;

Biroq, haqiqatda, ko'p miqdorda ma'lumot almashishini osonlashtirish uchun yuqori darajada yuzlab joriy etilgan xususiyatlar.

Kafka ma'lumotlarini topshiruvchi (producer) va ma'lumotlarini qabul qiluvchi (consumer) tashqi tizimlar ko'rsatadi. Producer va consumer'lar ko'rsatkichli xabarlar almashishini ko'rsatadilar, shuningdek, shu tizim ichidagi komponentlar orasida ham aloqa yaratishni ta'minlaydi.

Kafka, foydalanuvchilar tomonidan ko'rsatilgan holda, yozuvlarni ma'lumot bazasida saqlab turishiga, qidiruv tizimlariga bog'liq bo'lishiga yoki tahlil loyihalari uchun javoblar topishiga imkon beradi. Tizim ishlayotganda, Kafka kelayotgan ma'lumotlarni saqlab turadi va ularni istalgan vaqtda tahlil qilish uchun mavjud bo'ladi. Ushbu xususiyatlar tufayli, Kafka tizimlari bo'lishi kerak bo'lgan muhim vazifalarni bajarishi mumkin, masalan:

Ma'lumotlarni to'plash va o'zaro almashishni ta'minlash;

Ma'lumotlar bazasini qidiruv qilish tizimlari uchun ishlatish;

Ma'lumotlarni to'plab, so'rovlar topish uchun ishlatish.

Kafka, avtomatlashtirilgan tizimlar (microservices), Internet of Things (IoT) tizimlari, tahlil loyihalari, hujjat omborlari va boshqa ko'p miqdorda qo'shimcha muhim