HNBTecajnica(v3)

HNBTecajnica je web-aplikacija namijenjena prezentiranju tečajnih lista sa stranica HNB-a na čitljiv i korisniku razumljiv način. Proces je maksimalno automatiziran te korisnik treba samo unijeti datume početka i završetka. Aplikaciju je najlakše pokrenuti iz IDE-a ili direktno iz JAR datoteke.

Program započinje automatskim otvaranjem HTML forme koju korisnik treba ispuniti. HTML forma je prethodno napravljena i nalazi se u resources/template direktoriju. Za to je zadužena WebBrowser klasa, a nakon unosa datuma početka i završetka u formu, poziva se WebScraping klasa koja ima ulogu Managera. Ona prije daljnjeg izvršenja programa prvo provjerava jesu li uneseni datumi u odgovarajućem [YYYY-MM-DD] formatu.

Manager klasa preuzima podatke s HNB-ove stranice za svaki datum u intervalu datuma početka i završetka (uključivo). Pritom se koristi prilagođena klasa CustomDeserializer za ispravnu deserijalizaciju podataka iz JSON-a. Prvo se kreira nova tablica u H2 bazi podataka, a zatim se podaci spremaju u bazu, za što je zadužena klasa ExchangeRepository. Preko Managera se također šalju parametri konekcije klasi zaduženoj za generiranje reporta.

Nakon što se svi podaci spreme u bazu, Manager poziva ReportService klasu kojoj je zadatak generiranje PDF dokumenta sa svim tečajnim vrijednostima iz baze. Klasa koristi prethodno napravljene report i subreport datoteke iz resources direktorija.

Nakon uspješnog generiranja reporta, program se vraća u Browser klasu koja korisniku preko novog HTML dokumenta (također iz resources/template) obznanjuje da je novi PDF dokument uspješno napravljen. Nakon što to obavi, program se preko nove dretve automatski sam zatvara.

Glavni nedostatak je korištenje lokalne baze podataka u server načinu, zbog čega instanca baze mora biti otvorena kako bi se program mogao spojiti na nju i uspješno raditi. To bi se moglo riješiti korištenjem baze u embedded načinu ili, još bolje, spajanjem na remote bazu podataka.

Buduće mogućnosti uključuju korištenje remote baze koja bi sadržavala sve podatke s HNB-ovih stranica. Korisnik bi unosio željene datume te dobivao informacije o tečajnim listama samo za one datume koji ga zanimaju. Korištenjem remote baze eliminirala bi se i potreba za posjedovanjem lokalne kopije baze podataka, uz minimalne promjene postojećeg source koda. Još jedna mogućnost je unapređenje estetskog izgleda user interfacea kako bi proces korištenja aplikacije bio još lakši te kako bi se maknuo uvjet obaveznog korištenja [YYYY-MM-DD] formata. Također, isti princip rada aplikacije mogao bi se primijeniti i na druge djelatnosti, poput znanosti, sporta, telekomunikacija i drugih financijskih institucija.

JasperReports funkcionalnosti u programu

JasperReports je alat kojim se sadržaji iz nekog izvora podataka mogu izvesti u širok broj različitih datotečnih formata, između ostalog u obliku PDF datoteka što radi ovaj program. Alat je osobito koristan kad se radi o dinamičnom stvaranju sadržaja, pri čemu se najčešće čitaju podaci iz proizvoljne baze podataka te se zapisuju u dokument. Osnovna jedinica u JasperReportu je sam report, koji konceptualno predstavlja dokument kojeg nam je cilj generirati. Glavne vrste podataka u JasperReportu su Fields i Parameters. U njima su sadržani objekti koji se žele prikazati u generiranom dokumentu. Parameter se zadaje šabloni iz programskog koda, bilo direktno od strane korisnika ili preko nekog procesa u samom programu. Najlakše ih je zadati tako što se u programu prilikom poziva za punjenje šablone preda Map objekt koji kao value sadrži vrijednost koja se želi predati parametru, a kao key ime tog parametra. Ime obavezno mora biti isto kao ono koje je zadano u samoj šabloni reporta. Field je vrijednost koja se najčešće dobiva kao rezultat neke SQL operacije nad željenom bazom, iako se i parametar može predati Fieldu kao vrijednost, što se u ovom slučaju i napravilo i o čemu će biti riječi kasnije. Glavna razliku između njih je ta što kod parametara korisnik ima direktniju ulogu u vidu zadavanja vrijednosti. Također, vrijednosti Fieldova se u šabloni adresiraju formatom $F{*ime\_fielda*}, dok se parametri adresiraju s $P{*ime\_parametra*}.

Još jedna važna stvar kod JasperReporta su subreportovi. Oni su identični reportu, ali su sami podređeni nekom reportu. Drugim riječima, oni predstavljaju „dokument u dokumentu“. Posebno su korisni kad nam je cilj generirati dokument gdje se ponavlja isti segment podataka, npr. iste Field kategorije, ali i kad imamo neki element za kojeg želimo da se pojavi samo jednom, npr. naslov dokumenta. U tom slučaju će glavni report kao element imati taj naslov, a u sebi će integrirati subreportove koji će imati odgovarajuće Fieldove. Upravo je to slučaj u našem programu.

Prije same izrade reporta potrebno je u rubrici *Data Adapters* u odjeljku *Repository Explorers* definirati instancu baze iz koje želimo da šablona čita podatke. U našem slučaju to je *Database JDBC Connection*, a parametri te konekcije su identični kao oni zadani u samom Java programu. Pri izradi novog reporta određeni adapter se može specificirati, što će šabloni reći iz koje baze će čitati podatke. Naravno, pritom treba imati na umu koje su tablice i drugi objekti prisutni u bazi s kojom radimo.

Pregledajmo prvo subreport. Sastoji se od naslova i tablice od 9 stupaca, od čega njih 7 odgovaraju kategorijama tečajne liste iz baze podataka, dok su prvi i zadnji stupac direktno integrirani u subreportu. Prvi stupac služi kao oznaka rednog broja retka u tablici, počevši od 1, a to je postignuto jednostavnim korištenjem funkcije REPORT\_COUNT koja je integrirana u sam JasperReports alat. Zadnji stupac predstavlja *Lorem ipsum* i on je naprosto paragraf ispisan u tekstualnom području tog stupca te je sadržaj isti za svaki element tablice. Svi ostali stupci su kategorije tečajne liste iz same baze podataka koje je u programu prethodno napunjena s novim vrijednostima. Kako bi se do podataka došlo, potrebno je napraviti SQL upit. Pri izradi novog reporta može se odabrati unaprijed definirani oblik reporta (npr. sa ili bez tablice, vertikalne ili horizontalne orijentacije i sl.), pri čemu se također može definirati SQL upit koji će se koristiti za dohvat podataka iz baze, kao i kategorije koje će se u šabloni koristiti u obliku Fieldova. Upravo to je i napravljeno u izradi (sub)reportova. Nakon što se generira predefinirani dizajn, jednostavnim *drag and droppom* iz odjeljka *Outline* se tablica napunila ciljanim podacima. SQL upitom su se u Fieldove smjestili svi podaci o tečajnim listama na određeni datum. Sada je lako vidljivo zašto se baš ovakav način rada koristio za izradu subreporta, jer je to proces kojeg želimo ponoviti za svaki datum kojeg imamo u specificiranom intervalu. Pri tome su kao parametar svakom subreportu predani redom svi datumi iz intervala te je taj isti parametar korišten u SQL upitu. Upravo je taj parametar datuma pojedinom subreportu došao direktno iz reporta, čime se vraćamo u situaciju kad Field svoju vrijednost ne dobiva direktno iz baze već preko nekog parametra.

Što se reporta tiče, rješenje je jednostavnije. Estetski gledano, sastoji se samo od naslova i subreport odjeljka. Naslovu su preko parametara iz programa poslani datumi početka i završetka pretrage. Zatim se obavio SQL upit koji je iz tablice vratio sve jedinstvene datume iz dohvaćenih tečajnih lista. Ti su se datumi spremili u jedini Field glavnog reporta. Taj se Field prenio u svaki subreport kao njihov parametar, što je rezultiralo time da je svaki subreport redom imao sve datume iz intervala. Sada se možemo vratiti u subreport i vidjeti da kod njega rezultat SQL upita ovisi o datumu tečajnice, odnosno za svaki datum subreport će vratiti odgovarajuću tečajnu listu. Na kraju će se dokument sastojati od naslova koji ima dinamično generirani dio te onoliko subreportova koliko je datuma u unesenom intervalu. Upravo to je tražena funkcionalnost aplikacije.

Prije nego što se subreport krene koristiti u nekom reportu, on se mora kompajlirati!

Za realizaciju funkcionalnosti potrebna je neka baza podataka, u ovom slučaju korištena je H2 baza. Također, potreban je JasperReports uređivač u kojem se također može uređivati XML kod direktno kako bi se što više dala prilagoditi šablona koja se koristi.

Korišten je vlastiti font na bazi DejaVu Sans fonta u UTF-8 načinu PDF enkodiranja. Na taj način u generiranom dokumentu moguće je prikazati dijakritičke znakove. U uređivaču je potrebno dodati željeni font u rubriku vlastitih fontova, a nakon toga eksportirati JAR i dodati ga u biblioteku Java IDE-a. Cijeli proces detaljnije je opisan na sljedećem linku: <https://medium.com/@seymorethrottle/jasper-reports-adding-custom-fonts-589b55a52e7c>

Za stvaranje report šablona korišten je uređivač Jaspersoft Studio verzije 6.19.1 i verzija korištenog JasperReport Libraryja je također 6.19.1.