**Šta je novo u verziji 7?**

**Dozvole na nivou polja**

Sada postoje dozvole za polja. Možete deklarisati polja kao hidden ili read-only za jedu rolu. Hidden polja nisu raspoloživa na klijentu i ne mogu biti promenjena od strane korisnika sa ovom rolom iz brauzera. Read-only polja su onemogućena u brauzeru i ako su promenjena neće biti sačuvane promene na serveru.

**Detalji**

Detalji više nisu poseban tip noda. Svaki nod može biti detalj nekog drugog noda ako su povezani lookup poljem. Na primer, fakture mogu biti detalji kupca, jer fakture imaju kupac lookup polje koje je povezano sa lookup nodom kupaca. Na taj način nema potrebe za pisanjem koda kao u verzji 5.4 za slučaj povezivanja *invoice\_table* kao detalja i *tracks* u demo aplikaciji.

**Izračunata polja**

Postoje izračunata polja koja su takodje bazirana na lookup poljima. Na primer, možete imati polje koje će prikazivati broj prodatih pesama bez pisanja koda.

**Broj nivoa detalja**

Neoganičeni broj nivoa detalja je podržan.

**Čitanje i pisanje**

Čitanje i pisanje podataka je promenjeno. *on\_open* i *on\_apply* hendleri dogadjaja su zastareli. Umesto njih koriste se sledeći hendleri:

* *on\_before\_open*,
* *on\_after\_open,*
* *on\_before\_apply\_record* i
* *on\_after\_apply\_record*

dogadjaja.

*on\_before\_open* hendler dogadjaja se pokreće pre nego što je *sql* zahtev izvršen i može biti korišćen za validaciju zahteva i dodatnih filtera,

*on\_after\_open* handler je pokrenut pošto je *sql* zahtev izvršen i ima dataset kao parametar koji može biti modifikovan pre nego što je poslan na klijenta,

*on\_before\_apply\_record* hendler je pokrenut pre nego što je *sql* upit, koji treba da sačuva promene na tabeli baze podataka, izvršen i može biti korišćen za validaciju podataka i izračunavanje skoro svega na isti način kao što se to radi sada na klijentu,

*on\_after\_apply\_record* hendler je pokrenut pošto je *sql* upit, koji treba da sačuva promene u tabeli baze podataka, izvršen i polje primarnog ključa je postavljeno i može biti korišćen da izvrši druge dodatne promene u bazi podataka u istoj konekciji.

Pošto su podaci sačuvani, *delta* objekat je poslan klijentu i sve promene napravljene na serveru su propagirane na klijent.

Promene na zapisu koji ima detalje su procesirane pokretanjem *on\_before\_apply\_record* hendler događaja

* na master nodu,
* za svaki detalj nod koji je bio promenjen,
* za njegove sub-detalje koji su promenjeni itd.

Posle toga u obrnutom redosledu *on\_after\_apply\_record* hendler događaja je pokrenut za

* nodove sub detalja,
* nod detalja i
* nod master.

Ovo je tačno čak iako su promene napravljene samo na detaljima. Tako su promene na dokumentu (zapis i njegovi detalji) sačuvane kao celina.

**Kod za rad sa bazama podataka ponovo napisan**

Za MSSQL i MYSQL alternativni drajveri su podržani.

**Veličina tekst polja**

Veličina tekst polja može biti promenjeno za baze podataka koje imaju tekst polja specificirane dužine. Ako je nova veličina veća, dužina polja će biti promenjena, inače, dužina polja ostaje ista ali aplikacija proverava da li je dužina teksta manja od veličine polja.

**Edit i post metode**

Edit i post metode kada se puni zapis mogu biti izostavljeni. One se implementiraju interno.

**Kopije nodova**

U App builderu kopije (klonovi) nodova trebaju biti kreirani.

**Pomeranje nodova iz grupe u grupu**

Moguće je pomerati nodove iz jedne u drugu grupu.

**Koncept Jam.py**

Možete koristiti Jam.py za izradu interfejsa opterećenih aplikacija. Ove aplikacije mogu biti izradjene na

isti način kao desktop aplikacije, bez obzira na raspoloživost koda na klijentu. To je moguće zbog pune kontrole kojim podacima korisnik na klijentu može pristupiti i koje promene može napraviti na podacima koji mogu biti implementirani na serveru.

Ova kontrola je raspoloživa zbog postavke role na elemente i polja projekta i vrlo jednostavnog načina implementacije restrikcija sa serverskim hendlerima događaja.