Koszalin, 5. marca, 11. maja Roku Pańskiego 2021.

Apollo – żeby dać coś Światu

Przyczyny powstania

Ktoś powie: „Po co pisać program, w którym nie wszystko chodzi jak w Delphi?”. A przecież właśnie o to chodzi, by nie wszystko chodziło jak w Delphi. Delphi – oddajmy szacunek tej niezwykłej aplikacji – jest pionierem pewnych rozwiązań. Tam zostały wprowadzone po raz pierwszy. Jednakże doświadczony programista wie, że jest mnóstwo uciążliwości w pisaniu programów na tym świetnym programie. Rozumiem, że Delphi i rozwiązania tam stosowane stały się standardami. Jednak pewne wzory stają się już tylko uciążliwością, gdy się poza nie nie wychodzi długo. Świat byłby się wcale nie rozwijał, gdyby wszyscy ciągle stosowali wzory, które jako narosły tradycją kanon zadusiłyby chęć zmiany. My skupmy się nad tym, czego w propozycjach tych wzorcowych programów nie ma, co zrobiono w sposób wątpliwy albo nawet, gdzie są tam błędy.

Nie twierdzimy wszakże, że nie popełniamy błędów. Ależ popełniamy. I to obecnie pewnie jest więcej błędów niż prawidłowości. Jednak człowiek się na błędach uczy a to wartość nie do przecenienia.

Czy powinniśmy zatem robić wszystko, by kody napisane w tych wzorcach chodziły w naszym programie bez jakichkolwiek poprawek? Poszliśmy drogą, że wcale nas nie obchodzi, jak to mają w Delphi i jego przyciężkiej kopii Lazarusie, Typhonie i innych środowisk potomnych od FPC, które już były. Zrobiliśmy tak, by chodziło to po naszemu. Czy lepiej? Nie nam to oceniać. Program się dopiero rozwija, choć już jest dosyć zwartym środowiskiem programistycznym dostarczającym całe bogactwo narzędzi. Idziemy swoją drogą, bo taką mamy chęć. Tak nam fantazja podpowiada.

Czy oni tam dostaną białej gorączki, gdy zobaczą u nas to, czego sami u siebie nie mają? Nie jesteśmy siostrami miłosierdzia z okładami na gorączkę a piszemy użyteczny program, czyli taki, który nas samych zadowoli, gdy go dopracujemy. Jeśli dojdziemy do etapu, że do pisania programów już nam nic nie będzie potrzebne oprócz naszego programu, to zrobiliśmy plan podstawowy.

A jak zwie się nasz program? **Nasz program to program Apollo.**

Zacznijmy od samej struktury programu. U nas **w Apollu nie ma żadnego dodatkowego pliku projektu**, jak w Delphi a tym bardziej w Lazarusie. Żadnych dodatkowych

pliczydeł w czymś tam a tym bardziej w XML-u lub jakiejś innej cholerze tego typu. Nasz kod programu jest samowystarczalny. Apollo w momencie rozruchu programu – czy to przy otwieraniu go, czy podczas wskazania z listy zakładek – automatycznie analizuje kod programu i uzupełnia sobie wszystkie niezbędne mu informacje. Czy użytkownik się z tego ucieszy? A czemu nie, skoro ma dzięki temu wszystko to, co tam jest u innych i jeszcze trochę więcej? Na dysku nie zalega jakiś tam pliczor, w którym coś jest dogryzmolone.

Jedyny plik dodatkowy do projektu to plik o nazwie projektu i rozszerzeniu *.apc (Apollo Project Configuration – plik konfiguracji Apolla), który zawiera dodatkowe opcje kompilatora, gdy są konieczne przy kompilowaniu projektu. Gdy nie jest konieczny, brak tego pliku niczego nie powoduje w Apollu i pracy nad projektem. Jest to plik opcjonalny, tylko gdy projekt wymaga dodatkowych ustawień kompilatora.

Następnie **forma, nie jest żadnym dodatkowym plikiem doklejonym do pliku Pascala** (*.pas). W ogóle nie ma pliku formy. Wszystko jest tworzone metodą jawną w samym kodzie, czyli właśnie pliku Pascala (*.pas), w procedurze GenObiects automatycznie tworzonej przez Apolla. Nikt jednak nie broni programiście grzebać w tym kodzie, jeśli zachowa podstawowe zasady obsługiwania się tą bronią, jaką jest tak specyficzna procedura. Lepiej na przykład nie stosować obecnie przyrównań do: działań (dodawania, konkatenacji i tym podobnych), stałych, zmiennych i procedur. Trzeba wartości przyrównania zostawić w wersji podstawowej. Zapis całkowity, to całkowity, łańcuchowy to łańcuchowy i tak dalej. A jak komuś przeszkadza ta procedura, bo jest długa i nudna, albo z jakichś innych względów, których tu nie wymienimy, to może ją sobie zamknąć minusikiem listingu i nie będzie mu zawadzać. Jednak ta metoda daje wgląd w samo tworzenie obiektów i to bez jakiegokolwiek inspektora obiektów, czy zarządcy albo jak chce pewien znawca małp na drzewie – ciecia. Można ingerować w tworzenie obiektu od razu tam, gdzie powstał i nie trzeba robić korekty w kodzie, która właściwie zmieni wartości dopiero po przyznaniu ich w niewidocznym pliku formy. Programista jest tu panem sytuacji a nie jakiś działający w tle mechanizm, który owszem działa sam, ale spowalnia aplikację w momencie tworzenia obiektów. Ponadto programista widzi, jak to jest zbudowane i ma dzięki temu możliwość szybkiego zdobycia większej wiedzy o zasadach budowania wszystkiego, co tworzony przez niego program wydziwia.

Ponadto same programy – a w istocie ich projekty - są otwierane w dowolnych ilościach w edytorze kodu. Na zakładkach widać od razu po ikonkach, który to: program, forma i moduł, czy inny plik Pascala a nawet inne pliki. **Wystarczy zaznaczyć daną zakładkę programu, aby Apollo od tego momentu kompilował ten wskazany program z jego projektu.** To znakomicie ułatwia pracę w środowisku, gdyż można pracować na plikach współdzielonych przez wiele aplikacji i zamiennie sprawdzać, czy nasze poprawki i pomysły odpaliły w tych programach, które są podwieszone do tych plików. To coś nazwaliśmy multiplikowością nieco metodą nieuka Comta twórcy łacińsko-greckiego zrostu słownego zwanego socjologią. Podatność na multiplikowość to multiplikalność. Czyli Apollo jest multiplikalny, bo realizuje multiplikowość projektów a w zasadzie pracy na projektach. Możemy to poprawnościowo (bo słowo purysta dla miłośników polszczyzny powinno być słowem napiętnowanym) nazwać wieloplikowością i wieloplikalnością.

Ale Anglik lub inny Anglosas, lub anglojęzyczny, nazwie to multifilnością (multifileness) i multifilialnością (multifileability). Nie zabawnie? A jaka zabawa, przy tworzeniu aplikacji na Apollu z tą jego niepospolitą cechą. Nie trzeba otwierać drugiej i entej kopii środowiska programistycznego, by pootwierać wiele projektów. Kichać na to. Otwieramy Apolla i jedziemy na tylu projektach, na ilu chcemy i mamy do tego potrzebę. Tu nam nikt nie ogranicza naszych potrzeb a szczególnie potrzeb naszej wyobraźni.

Ale, żeby nie było za słodko, Apollo ma szczupłą paletę komponentów, które w Apollu w istocie nie są żadnymi komponentami a klasami. Nie ma też metody doklejania nowych komponentów do palet. Czy będzie miał? A czemu nie? Apollo się rozwija i w fałdach jego rozłożystego stroju, te – i nie tylko te - barwy też w końcu się rozwiną. W końcu w Delfach, od których pochodzi nazwa Delphi, była świątynia Apollina – czyli Apolla. Zatem Apollo rządzi, choć niektórym to nawet do głowy nie przyjdzie.

A czemu Apollo? Mam żal do Amerykanów – to prawda – o zamknięcie programu kosmicznego Apollo tak pięknie zaczętego od myśli i słów pierwszego katolickiego prezydenta Stanów Zjednoczonych Ameryki Północnej Johna Fitzgeralda Kennedy'iego (Kenediego). Czy Ameryka utraciła bezpowrotnie możliwość stanięcia na czele pochodu narodów w niczym nieograniczoną przestwór[1] Wszechświata? Tego nie wiem. Piszę program. Piszę program Apollo. Może właśnie po to, by tchnąć Ducha w tę zamierzchłą przeszłość, by kurz głupoty polityków całkiem nie zasypał pięknego pomysłu. Czy mój program może do czegoś dobrego posłużyć? Wierzę, że tak. Piszę go, by dać coś Światu. Moim naczelnym hasłem jest tu: **Apollo – żeby dać coś Światu.**

Ktoś powie: „A to pogaństwo.”. Hm, to i ruch olimpijski to typowe pogaństwo, święto hekatomby – stówołu – ku czci Apolla. Nie zastanawiajmy się nad tym. Dziś Apollo jest sługą Chrystusa, nawet jeśli ktoś chce inaczej, to będzie mu to trudno wykazać. Ale dobry sługa to dobry sługa. I takiego sługi potrzebujemy – siejącego swoje strzały zarazy po wszystkich krańcach Świata, ale także ślącego swoje służki Muzy, by opiekowały się zapędzoną – oby nie w kozi róg – Ludzkością. Czyli każdym z nas osobno i wszystkimi razem. I na tej palecie barw i na tych strunach zagrajmy, aby programistom było lepiej pracować i by ich płody były lepiej stworzone... Dzięki Apollu.

Z Bogiem.

Andrzej Marek Hendzel

1 Czy przestwór jest rodzaju żeńskiego czy męskiego? Ta przestwór czy ten przestwór. Oto dopiero pytanie.