1. Slaids

Labdien! Mani sauc Artis Pauniņš un šajā studiju semestrī es izstrādāju kvalifikācijas darbu, kurā tiek veidots .NET izsaukumu adapteris WebAppOS videi kā arī starpkoda ģeneratoru, kurš ģenerē C# kodu.

1. Slaids

Kas ir WebAppOS? Tā ir starpplatformu vide, kas ļauj izsaukt kodu, kas atrodas dažādos tīkla mezglos jeb serveros un kas rakstīts dažādās programmēšanas valodās.

WebAppOS atordas tīmekļa procesori, kas atbild par funkciju izsaukšanu un procesorā ir izsaukumu adapteri, kas ir paredzēti koda izsaukšanai. Adapteri veido katrai programmēšanas valodai atsevišķi.

1. Slaids

Šī shēma vienkāršoti izskaidro kā darbojas WebAppOS. Ir tīmekļa atmiņa, kas sinhronizē WebAppOS. Ir serveri, kuros glabājas funkcijas, kuras var izsaukt WebAppOS vidē, un lai tīmekļa procesors šīs funkcijas izsauktu, ir nepieciešams adapteris konkrētai programmēšanas valodai. Kā redzams tīmekļa procesoram pašlaik ir adapteri Javā un lua valodās, bet C# nav.

1. Slaids

Papildus adapteriem, WebAppOS ir RAAPI, kurš ir zema abstrakcijas līmeņa API tīmekļa atmiņas piekļūšanai, bet šo abstrakcijas līmeni vajag augstāku.

Tīmekļa atmiņā glabājas klases objekti ar visām tās vērtībām, bet pašlaik nav iespējams strādāt ar šiem atmiņas objektiem kā ar kādas dotas programmēšanas valodas objektiem.

Tādējādi kopā ir trīs problēmas un katra no šīm problēmām tiek risināta.

1. Slaids

Būs nepieciešams tīmekļa procesora adapteris valodai C# un .NET 5.

Būs nepieciešams augstāka abstrakcijas līmeņa API.

Un lai varētu izmantot tīmekļa procesora objektus kā programmēšanas valodas objektus (šajā gadījumā, kā C# objektus), tad tiks izstrādāta jaunas “definēšanas” valodas gramatika un kompilators, kas no jaunās valodas koda (turpmāk - jaunkods) ģenerēs starpkodu C# valodā, kurš vēl izmantos augstā līmeņa API, kuru jau pieminēju.

1. Slaids

Visiem risinājumiem tiek izmantota programmēšanas valoda C# ar .NET 5, jo .NET 5 ir atbalsts uz vairākām platformām (Platforma = OS + Procesora tips).

Jaunās valodas gramatikas un kompilatora veidošanai izmantosim ANTLR parsētāja ģeneratoru, jo tas ir labi zināms ģenerators ar atbalstu vairākās labi zināmās programmēšanas valodās, piemēram, C#, Java, JavaScript. Tiek izmantota 4. versija.

Ģeneratoram ir arī prasība, ka katru definēto klasi ģenerē savā failā, lai padarītu starpkodu uzturamāku un lai ieverotu OOP principus.

Kompilatoram tika veidotas datu struktūras, kurās glabā informāciju par jaunkodu, kuru kompilē.

1. Slaids

Ir jāveido kompilatora struktūra, kurā ir kompilēšanas funkcijas, kā arī glabā jaunkodā aprakstītās klases, asociācijas un jaunkodā pieļautās kļūdas, jo ir prasība, ka no kļūdaina jaunkoda starpkodu neģenerē.

Katrā klasē glabājas atribūti, metodes, asociāciju galapunkti, virsklase un tās apakšklases.

Asociācijās glabā divus asociācijas galapunktus – mērķi un avotu. Katram galapunktam ir lomas vārds un klase, kurai tā ir piesaistīta.

Klasē tiek definēti lauki, kas ir atribūti un metodes un tiem ir jāglabā aizsardzības līmeni, datu tipu un vārdu, bet metodēm vēl papildus glabā anotācijas, kas apraksta metodes un URL, kas pasaka, kur atrodas metodes kods (uz servera, lokāli), kādā valodā kods ir rakstīts un kā piekļūt šīs metodes kodam.

1. Slaids

Iepriekš tika pieminēts RAAPI. RAAPI ir funkcijas, kuras var izmantot, lai piekļūtu tīmekļa atmiņā esošajiem objektiem. Šīs RAAPI funkcijas kalpo par pamatu augstā līmeņa API realizācijai.

Datu iegūšanai no RAAPI vēl izmanto LINQ iebūvētās funkcijas, ar kuras palīdzību var iegūt datus un pārveidot tos par objektiem.

Gluži kā ar kompilatoru, arī šim risinājumam tika veidotas datu struktūras.

1. Slaids

Šeit galvenās struktūras ir tīmekļa atmiņa un tīmekļa klase. Tīmekļa atmiņā glabājas klases un klasēs glabājas objekti, asociāciju galapunkti un atribūti.

Atribūtos glabājas datu tips un vārds un asociāciju galapunktos glabājas lomas vārds un klases.

Netiek glabātas metodes, jo kā minēju, tas glabājas citur un klašu objektos jeb instancēs glabājas atribūtu vērtības.

1. Slaids

Adapterim ir nepieciešami divu veidu funkciju izsaukumi – lokāli un tālvadības (remote). Remote izsaukumiem tiek izmantotas RAAPI iebūvētās funkcijas. Lokāliem izsaukumiem tiek izmantota refleksija, tādējādi dinamiski izsaucam funkcijas. Kā minēju metodēm ir URL un tas tiek rakstīts formātā *protokols:lokācija:ceļš* un tā kā C# refleksija strādā ar .NET, tad šobrīd ar refleksiju var izsaukt tikai .NET funkcijas, kuras atrodas lokālā failu sistēmā.

1. Slaids

Rezultātā tika iegūts, ka WebAppOS vidē ar izstrādātajiem rīkiem var izsaukt funkcijas un ka var darboties ar tīmekļa atmiņas objektiem kā ar C# objektiem.

Veicot funkcijpunktu aprēķinus, tika aprēķināts, ka projekts iekļāvās 3 cilvēkmenešos.

Tā kā šie risinājumi tika veidoti WebAppOS, tad tie tiks iekļauti tās distribūcijā.