JANUAR 2021

Pika

SHAREPOINT

VZPOSTAVITEV PORTALA ZA SKUPNO RABO INFORMACIJ

SharePoint Online – Migracija delovnih tokov na PowerAutomate 2.del
str. 10

RAZVOJ

RAZVOJ REŠITEV PO MERI

Telerik Reporting v okolju ASP.NET Core Blazor in Entity Framework
str. 18

IZOBRAŽEVANJA

MICROSOFT URADNI TEČAJI

In se veselimo leta, ki prihaja
str. 34

VIRTUALNI DOGODEK
Ljubljana, 16. marec 2021

GLOBAL CLOUD SKILLS v-TOUR

LLPA Cloud Skills Cup
TOURNAMENT SLOVENIA

do 20% POPUST NA PRIJAVE

13-14 APRIL 2021
Bohinjska Bistrica, Slovenia
THRIVE CONFERENCE

Xnet 25let

INFRASTRUKTURA | RAZVOJ APLIKACIJ IN SPLETNIH STRANI | OFFICE TEČAJI | SHAREPOINT | ITIL | SQL | BI


11. KONFERENCA
O MODERNIH IT TEHNOLOGIJAH
THRIVE
CONFERENCE
13-14 APRIL 2021
Bohinjska Bistrica, Slovenia
do 20%
POPUST NA
PRIJAVE
Zagotovite si svojo vstopnico zdaj!
thriveconf.com

UVODNIK

Spoštovani,

Zares veliko dobrih in spodbudnih želja smo si izmenjali v preteklih tednih. Zato mi dovolite, da še sama »pristavim svoj piskerček« in vam zaželim vse dobro.

Iz vsega srca vam želim zdravja, razumevanja, sočutja, znanja, strasti, ljubezni, sreče, pozitivne energije in poleta, da boste segli po zvezdah ter osrečili sebe in vam drage.
Naj bo leto 2021 veselo in uspešno, naj se vam izponi tud kakšno drobna, skrita želja.

Hvala vsem, ki nam zaupate, da lahko sodelujemo in skupaj ustvarjamo zanimive ter uspešne projekte. Veselimo se že novih izzivov, ki bodo pisali naše skupne, uspešne zgodbe leta 2021. Mi bomo tu, ZA VAS, ko boste želeli naše sodelovanje, nasvet ali pomoč. Obenem pa vas vabimo, da nas priporočite tudi svojim poslovnim partnerjem, prijateljem. Res iskrena hvala vam za to.

Zelo počasi a vztrajno se povečuje zaupanje v live virtual tečaje pri nas in tudi število prijav. Prepričani smo, da predvsem zato, ker so izkušnje tistih, ki so se opogumili, da tak način preizkusijo, zelo dobre. Vtisi "live-virtual" udeležencev. Posebej spodbudno je tudi to, da ne gre več le za tečaje za tehnične secialiste, pač pa vedno več končnih uporabnikov širi svoje znanje v naših navideznih učilnicah.

»Zadnji tečaj iz naslova, ki ste ga opravili virtualno za Elektro Ljubljana, je navdušil naše zaposlene!« dr. Alenka Kolar, PMP Elektro Ljubljana, d.d.
(Tečaj Excel 365 nadaljevalni, live virtual, Klemen Vončina, oktober 2020)

»Predavatelj odličen (strokovno podkovan, zabaven, praktičen), Kompas Xnet sodelavke pa zelo prijazne in vedno pripravljene pomagat. Z vami sem vedno zadovoljen!« Andraž Zlobec Elektro Primorska d.d.
(Tečaj 10987 Performance Tuning and Optimizing SQL Databases, live virtual, Dejan Sarka, oktober 2020)

Prisrčno vas vabim, da se nam pridružite tudi vi, saj boste v našem koledarju tečajev, zagotovo našli nekaj zase in za svoje sodelavce. Če izpostavim le nekatere:

- DA100: Analyzing Data with Power BI - **Dejan Sarka**, 25.- 28.1.2021
- 10997: Office 365 Administration and Troubleshooting - **Miha Pihler**, 27.- 29.1.2021
- 20762 Developing SQL Databases - **Dejan Sarka**, 1. - 5.2.2021
- 10961: Automating Administration with Windows PowerShell - **Miha Pihler**, 1. - 5.2.2021
- AZ303: Microsoft Azure Architect Technologies - **Jože Markič**, 22. - 26.2.2021
- 20339-1: Planning and Administering SharePoint 2016 – **Robi Vončina**, 22. - 26.2.2021
- WS-011: Windows Server 2019 Administration – **Luka Manojlovič**, 29.3. – 2.4.2021

Tu so še **Excel, Teams, BI, Power Apps, O365, Exchange, DevOps, ITIL, Scrum** ter vrsta, posebej vam prilagojenih delavnic, z odličnimi predavatelji, ki bodo pomembno nadgradili vaše kompetence. Ne zamudite priložnosti, da si povrnete investicijo v znanje prek **spodbude za digitalno transformacijo**. Cilji javnega razpisa so: izvedba digitalne transformacije, izboljšanje kompetenc zaposlenih in povečanje dodane vrednosti. Podrobnosti na Podjetniškem skladu >

Prisluhnite, kako ocenjuje pomen digitalnih kompetenc **Christian Pedersen,** TietoEvry, Norveška, ki po svetu zaposluje več kot 24 tisoč specialistov.

Držimo pesti, da bo aprila že precej lažje in se bomo lahko ponovno družili, zato vas vljudno vabim, da si pogledate program Thrive konference in se čimprej prijavite. Zgodnje prijave so do konca februarja, zagotovite si do **20 % popust**.
V času zgodnjih prijav lahko kupite določeno število vstopnic, imena udeležencev pa sporočite kasneje. **Ne odlašajte s prijavo na najboljši tehnično – izobraževalni dogodek s tradicijo in mednarodno udeležbo.**

Povabite nas k sodelovanju na projektih, vezanih na Microsoft tehnologije, kot na primer: implementacije, nadgradnje in migracije elektronske pošte, SharePoint-a, vzpostavitev System Center orodij, migracija storitev v **O365** ali **Azure** ipd.
Na voljo imamo tudi vrsto dodatkov za SharePoint - Katalog SharePoint gradnikov >

Zdaj se že nekoliko ponavljam, a vam zares želim, da ostanete zdravi in da vas corona zaobide tako in drugače: bodite zdravi vi in vaši najbljižji.

Naj bo **Xnet vaša prva izbira, ko gre za IT rešitve in storitve.** Microsoft tehnologije so naša strast.

Čuvajte se in ostanite zdravi!
Branka Slinkar

Xnet
Leading Learning Partners Association
Member
A Proud Member of the LLPA, Microsoft's Partner of the Year
Learning
Microsoft Partner
2020 Partner of the Year Winner
Learning Award

Pika
ISSN: 1408-7863
Kompas Xnet d.o.o.
Stegne 7
1000 Ljubljana

Telefon: 01 5136 990
Fax: 01 5136 999
Email: info@kompas-xnet.si
Web: https://www.kompas-xnet.si

Direktorica
Branka Slinkar

Urednica in oblikovalka
Urška Premzl

Člani uredništva
Aleš Lipušček, Aida Kalender Avdić, Gašper Rupnik, Miha Pihler, Jože Markič, Klemen Vončina, Robert Vončina, Anja Rupnik, Petra Militarev, Dejan Sarka, Andraž Bergant, Manca Gruden

Odri so, delavcev pa ni	Anja & Gašper
Stotka je padla! Čestitamo!	Klemen
Če zavihaš rokave, potem pa gre	Aida, Manca, Petra
Ves čas so v vrsti	Robi
Tudi do službe jim pomaga	Luka
Kdaj bo prva lopata?	Miha
Monika gre k babici	Jože
Za kolo je že preveč mraz	Aleš
S »peškondo« je sigurno	Aida
Idej jim nikoli ne zmanjka	Urška
Večkrat je še tema …	Mojca
Vsakokrat je nekaj novega	Domen
ESI / SATV / C3, …	Petra
Z vsem se že spopade	Andraž
Statistika ne laže!	Dejan

KOLOFON

MICROSOFT OFFICE

8 UPORABA SPREMENLJIVK S FUNKCIJO LET
Klemen Vončina
Microsoft Office Specialist Master

SHAREPOINT

10 SHAREPOINT ONLINE – MIGRACIJA DELOVNIH TOKOV NA POWERAUTOMATE 2.DEL
Robi Vončina
MVP, MCT, MCITP, MCSA, MCTS

SQL

13 SQL SERVER SECURITY PART 11: PREDICATE-BASED ROW-LEVEL SECURITY
Dejan Sarka
MVP, MCT

RAZVOJ

18 TELERIK REPORTING V OKOLJU ASP. NET CORE BLAZOR IN ENTITY FRAMEWORK
Gašper Rupnik
MCT, MS, MCSD, MCPS

24 CHROME ORODJA ZA RAZVIJALCE - ELEMENTS
Domen Gričar
MCSD, MCSA, MCT

26 KAKO DELUJEJO PREVAJALNIKI (ANG. COMPILERS)?
Andraž Bergant

ADMINISTRACIJA

27 POWERSHELL KOTIČEK
Aleš Lipušček
MCP, MCTS, MCITP

28 MICROSOFT ENDPOINT MANAGER
Jože Markič
MCT, IT arhitekt, sistemski inženir,

31 MREŽNI ZAJEM PODATKOV (NETWORK CAPTURE)
MIHA PIHLER
MCT

DRUGO

34 IN SE VESELIMO LETA, KI PRIHAJA
Petra Militarev
Vodja izobraževanj

Predavanje je bilo odlično, na trenutke precej zahtevno, vendar pa vsekakor zelo koristno z ustrezno pripravljenimi primeri in dodatnimi nalogami. Predavatelj strokoven, poln izkušenj in praktičnih primerov, s katerimi lepo ponazori teorijo.

Vid, Gen-I

TEČAJI ZA IT STROKOVNJAKE

Počutili se boste kot v učilnici, vendar iz udobja doma/pisarne

20764

Administering a SQL Database Infrastructure

Kdaj: : 15. 2. - 19. 2. 2021
Predava: Dejan Sarka, MCT

POGLEJ VEČ

20339-1

Planning and Administering SharePoint 2016

Kdaj: : 22. 2. - 26. 2. 2021
Predava: Robi Vončina, MCT

POGLEJ VEČ

MS030

Office 365 Administrator

Kdaj: : 22. 2. - 26. 2. 2021
Predava: Miha Pihler, MCT

POGLEJ VEČ

Forensics and Incident Handling

Kdaj: : 15. 2. - 19. 2. 2021
Predava: Paula Januszkiewicz, Cqure, MVP, MCT

POGLEJ VEČ

Za vse informacije so vam na voljo na info@kompas-xnet.si ali prek tel.: 01 01 5136 990

Sproščenost, ravno pravšnji tempo (ni bilo nepotrebnega hitenja, saj je potem nemogoče uloviti predavatelja), zanimivi praktični primeri, pozivanje k vprašanjem po vsakem koraku.

g. Matjaž, Petrol

TEČAJI ZA UPORABNIKE

Spoznajte, v živo ali na daljavo, osnovne ali napredne funkcionalnost zbirk programov Microsoft Office, Microsoft Office 365.

Uvod v Excel BI

Kdaj: : 9. 2. 2021

POGLEJ VEČ

Microsoft Excel začetni

Kdaj: : 11. 3. - 12. 3. 2021

POGLEJ VEČ

Microsoft Excel nadaljevalni

Kdaj: : 10. 2. - 12. 2. 2021

POGLEJ VEČ

Microsoft Teams

Kdaj: : 5. 2. 2021

POGLEJ VEČ

Office 365 - za uporabnike

Kdaj: : 8. 3. 2021

POGLEJ VEČ

Microsoft Access začetni

Kdaj: : 8. 3. - 10. 3. 2021

POGLEJ VEČ

Za vse informacije so vam na voljo na info@kompas-xnet.si ali prek tel.: 01 01 5136 990

Uporaba spremenljivk s funkcijo LET

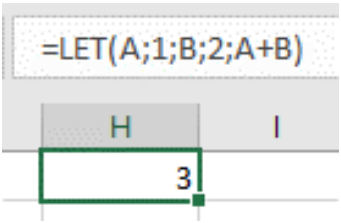
Klemen Vončina
Microsoft Office Specialist Master
klemen.voncina@kompas-xnet.si

V naboru Excelovih funkcij, ki jih imamo na voljo uporabniki Officea 365, se nahaja tudi funkcija, ki nam omogoča, da definiramo spremenljivke, ki jih nato uporabimo v neki kalkulaciji znotraj te iste funkcije. Spremenljivk torej ne določimo za uporabo povsod po Excelu, pač pa samo znotraj ene funkcije. To nam omogoča funkcija LET.

Funkcija pride zelo prav, kadar moramo napisati bolj kompleksne kalkulacije, znotraj katerih uporabljamo še druge kompleksne kalkulacije. Torej namesto da napišemo funkcijo, ki bo vsebovala ogromno drugih gnezdenih funkcij, vse tiste funkcije, ki bi jih morali gnezditi, definiramo kot spremenljivke, nato pa se v "glavni" kalkulaciji sklicujemo na imena spremenljivk. Poleg boljše čitljivosti s tem izboljšamo tudi hitrost delovanja funkcij, saj se mora vsaka spremenljivka preračunati le enkrat, medtem ko se ponavljajoče gnezdene funkcije izračunavajo vsakič znova.

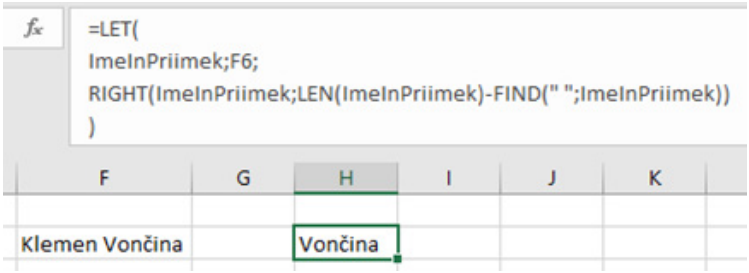
Poglejmo najprej en zelo enostaven primer:
=LET(A;1;B;2;A+B)
V tej funkciji smo navedli 2 spremenljivki:

- A, ki smo ji dodelili vrednost 1.
- B, ki smo ji dodelili vrednost 2.



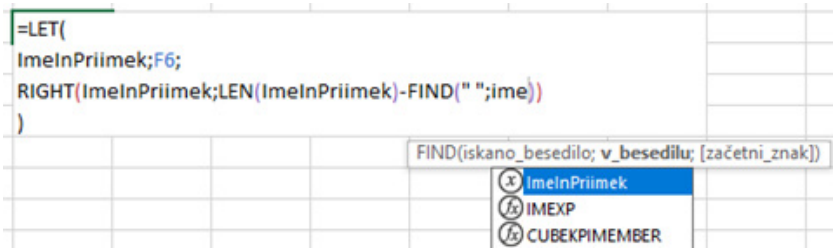
Po definiciji spremenljivk pa smo napisali formulo, ki je obe spremenljivki seštela (A+B). Rezultat je seveda 3.
Funkcijo LET torej uporabljamo tako, da najprej v parih določamo imena spremenljivk in njihovih vrednosti. Definiramo lahko bodisi samo eno spremenljivko, bodisi več. Čisto na koncu funkcije pa lahko vse definirane spremenljivke uporabimo v formuli. Zanimivo je, da pri definiranju imen spremenljivk imen ne rabimo vpisati znotraj narekovajev.

Zgornji primer je seveda sila enostaven, funkcija LET pa je namenjena bolj kompleksnim funkcijam. Uporabili bi jo lahko denimo za nekoliko izboljšano čitljivost kombinacije funkcij, o kateri sem pisal nekaj časa nazaj, in sicer RIGHT(LEN-FIND). Namesto, da se v vsaki funkciji znova sklicujemo na celico, v kateri imamo vhodne podatke, lahko



to celico definiramo kot spremenljivko. In se potem sklicujemo na ime spremenljivke. Na sliki spodaj sem spremenljivki "ImeInPriimek" dodelil vrednost, ki se nahaja v celici F6.

Prednost tovrstnega dela je tudi to, da nam Excel pri samem pisanju funkcije nenehno ponuja definirane spremenljivke, zaradi česar



je lažje tudi pisanje funkcije, saj ne rabimo ves čas "skakati" naokoli po delovnem zvezku in iskati celic z vrednosti, ki jih želimo uporabiti v funkciji.

Še dodaten namig – kako spraviti funkcijo (ali poljubno vsebino) v več vrstic znotraj iste celice (kot na zgornjih slikah)? Na mestu, kjer želimo narediti prelom vrstice, stisnemo kombinacijo Alt + Enter.

Robi Vončina
MVP, MCT, MCITP, MCSA, MCTS
robi.voncina@kompas-xnet.si

SharePoint Online – Migracija delovnih tokov na PowerAutomate 2.del

Najprej vsem srečno in zdravo v leto 2021 in upam, da bo to leto veliko bolj normalno, kot je bilo preteklo. Situacija sicer ni najbolj rožnata, a upam, da v tem primeru zdrži rek, »slab začetek, dober konec«.

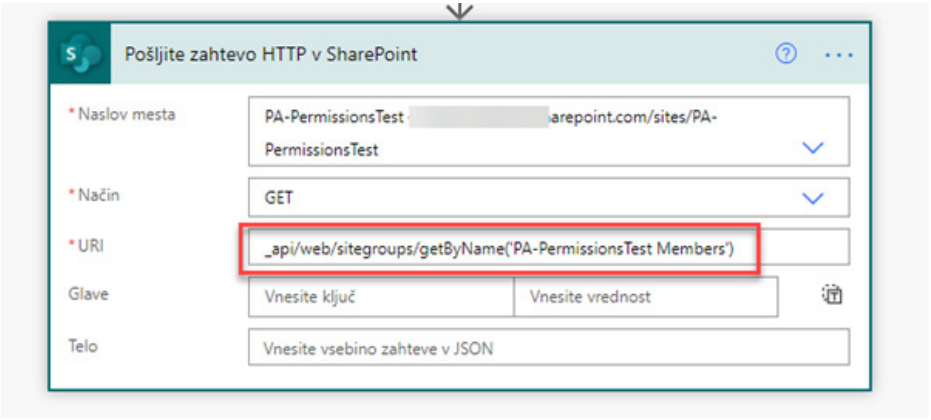
V prejšnji številki Pike, sem pričel s pisanjem članka o migraciji delovnih tokov na PowerAutomate. Predvsem je bil fokus na migraciji akcij, ki se nanašajo na nastavljanje pravic, saj so v PowerAutomate okolju, na voljo samo v omejenem obsegu in ne tako, kot smo bili vajeni na platformi SharePoint 2010 delovnih tokov. Prvi del je predstavil predvsem nastavitev pravic za uporabnike oz. »user objects«, v tej številki pa bo tematika nastavitev pravic na SharePoint skupine.

Logika

Podobno, kot je bilo za nastavljanje pravic za uporabnike, moramo tudi v tem primeru uporabiti akciji »Send HTTP request to SharePoint« in »Parse JSON«.

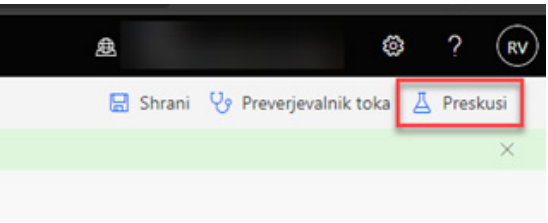
Prva se uporablja 2x, in sicer da pridobimo ID skupine, kateri bi radi dodelili pravice in nato še nastavimo pravice. Slednja se uporablja zato, da iz poizvedbe o ID-ju skupine, preberemo rezultat in ga uporabimo pri naslednje klicu za nastavitev pravic.

Za pridobitev informacije o ID-ju skupine, lahko naredite poizvedbo na REST endpoint: »_api/web/sitegroups/getByName('PA-PermissionsTest Members')«, kjer lahko ime skupine »PA-PermissionsTest Members« zamenjate s svojim imenom. Da bi lahko bil delovni tok dinamičen, lahko namesto statične vrednosti uporabite spremenljivko, ki jo lahko npr. dobite iz vrednosti polja.



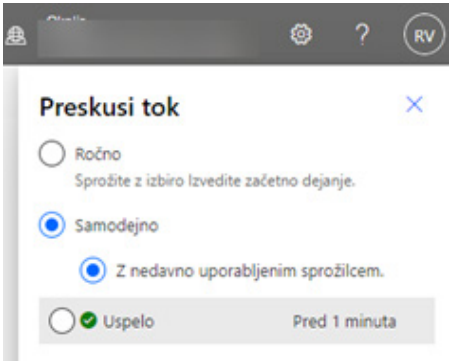
V naslednjem koraku moramo razčleniti vsebino JSON odgovora, ki ga pridobimo s to akcijo. Da bi lahko uporabili razčlenitev, moramo v to akcijo vnesti primer JSON odgovora, iz katerega se ustvari shema, ali pa shemo ročno vpisati. Ker je lahko JSON odgovor precej kompleksen, in s tem povezana tudi shema, je za nas najlažje, da testno poženemo delovni tok, in nato prekopiramo JSON odgovor iz akcije.

Testno se delovni tok požene tako, da na zgornjem desnem robu, kliknete na gumb preskusi:



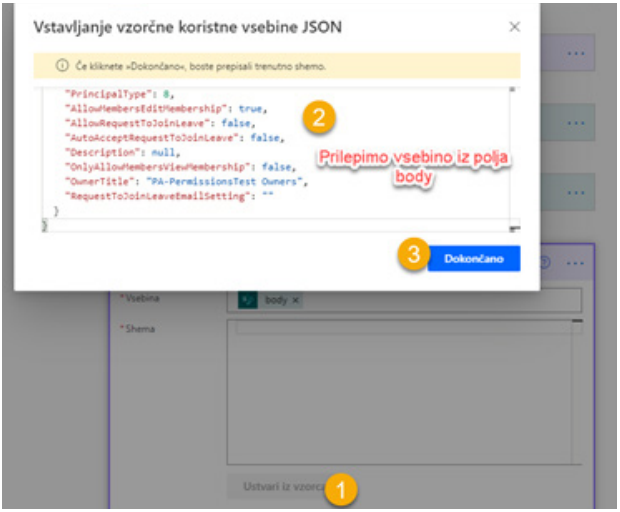
Nato imate na voljo ali ročni zagon ali samodejni zagon. Odvisno od tega, kaj je v vašem primeru sprožilec in ali se navezujete na elemente v SharePoint seznamu, bo mogoče potrebna tudi akcija ustvarjanja ali spreminjanja elementov, da bi lahko pridobili potrebne informacije za zagon toka.

V mojem primeru, je proces vezan na usvarjanje in spreminjanje elementov v SharePoint seznamu, zato moram vsaj 1x ustvariti nov element, da se sproži tok in pridobim informacije za zagon. Za vsa nadaljnja testiranja nato lahko uporabim že obstoječe informacije, kot prikazano na sliki.



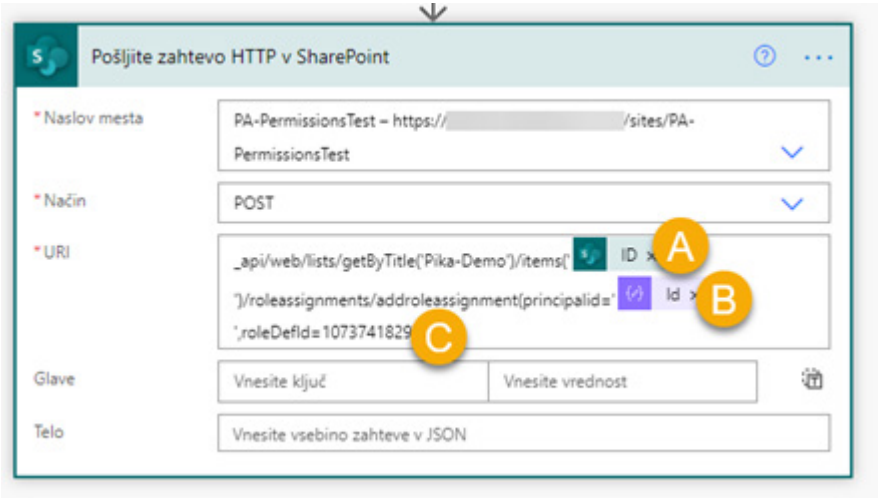
V zgodovini izvajanja delovnega toka, lahko odpremo nedavno izvajanje in poiščemo akcijo za pridobitev ID-ja SharePoint skupine. Za razčlenitev JSON strukture, si moramo prekopirate vsebino iz polja »Body«

To vsebino nato prilepimo v akcijo »Razčlenitev JSON«.



Pred nami je še zadnji korak, v katerem moramo zopet poslati HTTP poizvedbo na SharePoint, ki se glasi:

»_api/web/lists/getByTitle('Pika-Demo')/items('ID')/roleassignments/addroleassignment(principalid='Id',roleDefId=1073741829)«



A. ID elementa na katerem želimo nastaviti pravice

B. ID skupine, kateri bomo nastavili pravice. ID smo pridobili v prejšnji HTTP poizvedbi

C. RoleDefId ID nivoja pravic. Kako pridemo do tega ID-ja je bilo razloženo v prejšnji številki.

Delovni tok shranite in preskusite. S tem, smo izvedli tudi zadnje dejanje v delovnem toku za nastavitev pravic.

Upam, da vam bo to v pomoč pri izdelavi bolj kompleksnih delovnih tokov v prihodnje. V primeru, da potrebujete našo pomoč, ste vabljeni da mi pišete na naslov: robi.voncina@kompas-xnet.si.

SQL Server Security Part 11: Predicate-Based Row-Level Security

Dejan Sarka
MVP, MCT
dsarka@solidq.com

Using programmable objects for RLS protects sensitive data very well because users don't have direct access to the tables. However, the implementation of such a security might be very complex for existing applications that don't use stored procedures, and other programmable objects. This is why SQL Server 2016 and later include predicate-based RLS. A DBA creates the security filters and policies. The security policies are transparent to the application. There are two types of RLS security predicates:

- **Filter predicates** that silently filter the rows the application reads. For these predicates, no application change is needed. Note that, besides reading, filter predicates also filter the rows when an application updates or deletes the rows; this is because the application again simply does not see the filtered rows.
- **Block predicates** that explicitly block write operations. You can define them for after-insert and after-update operations, when the predicates block inserts or updates that would move a row beyond the scope of the block predicate. After-insert block predicates also apply to minimally logged or bulk inserts. You can also define block predicates for before-update and before-delete operations, when they serve as filter predicates for the updates and deletes. Note that if you already use filter predicates, before-update and before-delete predicates are not needed. You might want to change the affected applications to catch additional errors produced by block predicates.

Setting Up the Environment

For testing predicate-based RLS, I will use the same database and data as I did in my previous article, when I discussed using programmable objects for RLS. This is the code from that article that creates the demo objects.

To start testing programmable-object-based RLS, let's create a new demo database and change the context to this database:

```
USE master;
IF DB_ID(N'RLSDemo') IS NULL
CREATE DATABASE RLSDemo;
GO
USE RLSDemo;
```

The next step is to create four database users without logins. Three of them represent regular users from the sales department, and the fourth one represents the sales department manager. We will use the following program to create the databases:

```
CREATE USER SalesUser1 WITHOUT LOGIN;
CREATE USER SalesUser2 WITHOUT LOGIN;
CREATE USER SalesUser3 WITHOUT LOGIN;
CREATE USER SalesManager WITHOUT LOGIN;
GO
```

The next piece of code creates a table for the employee data. This table needs RLS:

```
CREATE TABLE dbo.Employees
(
 EmployeeId   INT        NOT NULL PRIMARY KEY,
 EmployeeName NVARCHAR(10) NOT NULL,
 SalesRegion  NVARCHAR(3)  NOT NULL,
 SalaryRank   INT        NOT NULL
);
GO
```

Now let's insert some data into the dbo.Employees table. The three rows inserted represent the three regular users from the sales department. You can check the inserted rows immediately with a query. Note that the sales region for the first two users is USA, and for the third one it is EU:

```
INSERT INTO dbo.Employees
(EmployeeId, EmployeeName, SalesRegion, SalaryRank)
VALUES
 (1, N'SalesUser1', N'USA', 5),
 (2, N'SalesUser2', N'USA', 4),
 (3, N'SalesUser3', N'EU', 6);
-- Check the data
SELECT *
FROM dbo.Employees;
GO
```

The dbo.Customers table, created with the following code, will also need RLS:

```
CREATE TABLE dbo.Customers
(
 CustomerId   INT        NOT NULL PRIMARY KEY,
 CustomerName NVARCHAR(10) NOT NULL,
 SalesRegion  NVARCHAR(3)  NOT NULL
);
GO
```

Again, let's insert some rows into this table and check them. There are two customers from the USA and two from the EU:

```
INSERT INTO dbo.Customers
(CustomerId, CustomerName, SalesRegion)
VALUES
 (1, N'Customer01', N'USA'),
 (2, N'Customer02', N'USA'),
 (3, N'Customer03', N'EU'),
 (4, N'Customer04', N'EU');
-- Check the data
SELECT *
FROM dbo.Customers;
GO
```

Simple Filter Predicates

You define predicates through a predicate function. In the body of this function, you can use other tables with the JOIN or APPLY operators. If the function is schema-bound, no additional permission checks are needed. If the function is not schema-bound, users need permissions to read the data from the joined tables. When a predicate function is schema-bound, you cannot modify the objects it refers to.

A security policy adds an RLS predicate to a table using a predicate function. The policy can be disabled. If it is disabled, users see all of the rows. A security policy also filters and/or blocks the rows for the database owners (the dbo user, db_owner database, and sysadmin server roles).

Before testing SQL Server RLS, users need permissions to read the data. The following code gives users permissions to read data from both tables in the demo database:

```
GRANT SELECT ON dbo.Employees
TO SalesUser1, SalesUser2, SalesUser3, SalesManager;
GRANT SELECT ON dbo.Customers
TO SalesUser1, SalesUser2, SalesUser3, SalesManager;
GO
```

To check the permissions, you can try to read from

the dbo.Employees table by impersonating each of the users again. All of the users see all of the rows. The following code illustrates this process:

```
SELECT * FROM dbo.Employees;
EXECUTE (N'SELECT * FROM dbo.Employees') AS USER = N'SalesUser1';
EXECUTE (N'SELECT * FROM dbo.Employees') AS USER = N'SalesUser2';
EXECUTE (N'SELECT * FROM dbo.Employees') AS USER = N'SalesUser3';
EXECUTE (N'SELECT * FROM dbo.Employees') AS USER = N'SalesManager';
GO
```

The following command creates a separate schema for security objects. It is good practice to move security objects into a separate schema. If they are in a regular schema, a DBA might inadvertently give permission to modify the security objects when giving the ALTER SCHEMA permission to users for some other reason, such as allowing them to alter the procedures in that schema. The following code illustrates this process:

```
CREATE SCHEMA Security;
GO
```

The following predicate function limits the users to seeing only their own rows in a table. The Sales department manager role can see all rows. In addition, the predicate also takes care of the dbo users, enabling these users to see all of the rows as well:

```
CREATE FUNCTION Security.EmployeesRLS(@UserName AS NVARCHAR(10))
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN SELECT 1 AS SecurityPredicateResult
 WHERE @UserName = USER_NAME()
   OR USER_NAME() IN (N'SalesManager', N'dbo');
GO
```

The next step is to create the security policy. The security policy created with the following code adds a filter predicate for the dbo.Employees table. Note that the EmployeeName column is used as the argument for the predicate function:

```
CREATE SECURITY POLICY EmployeesFilter
ADD FILTER PREDICATE Security.EmployeesRLS(EmployeeName)
ON dbo.Employees
WITH (STATE = ON);
GO
```

You can test the filter predicate by querying the dbo. Employees table again. This time, each regular user gets their own row only, while the Sales department manager and the dbo users see all of the rows. The following code illustrates this process:

```
SELECT * FROM dbo.Employees;
EXECUTE (N'SELECT * FROM dbo.Employees') AS USER = N'SalesUser1';
EXECUTE (N'SELECT * FROM dbo.Employees') AS USER = N'SalesUser2';
EXECUTE (N'SELECT * FROM dbo.Employees') AS USER = N'SalesUser3';
EXECUTE (N'SELECT * FROM dbo.Employees') AS USER = N'SalesManager';
GO
```

Note that users can still gain access to sensitive data if they can write queries. With carefully crafted queries, they can conclude that a specific row exists, for example. The salary rank for the SalesUser1 user is 5. This user might be interested if another user with salary rank 6 exists. The user can execute the following query:

```
EXECUTE (N'SELECT * FROM dbo.Employees
WHERE SalaryRank = 6')
AS USER = N'SalesUser1';
```

The query returns zero rows. The SalesUser1 did not get any information yet. However, when a query is executed in SQL Server 2016 or 2017, the WHERE predicate is evaluated before the security policy filter predicate. Imagine that the SalesUser1 tries to execute the following query:

```
EXECUTE (N'SELECT * FROM dbo.Employees
WHERE SalaryRank / (SalaryRank - 6) = 0')
AS USER = N'SalesUser1';
```

When you execute this code in SQL Server 2016 or 2017, you get error 8134, divide by zero error encountered. Now SalesUser1 knows that an employee with salary rank equal to 6 exists. SQL Server 2019 returns an empty rowset for this query as well, and thus closes another possible security hole.

Complex Filter Predicates

Now let's create another predicate function that will be used to filter the rows in the dbo. Customers table. It applies a tabular expression to each row in the dbo. Customers table to include rows with the same sales region as the sales region value for the user who is querying the tables. It does not filter the data for the sales department manager and the dbo database user. The following code illustrates this process:

```
CREATE FUNCTION Security.CustomersRLS(@
CustomerId AS INT)
RETURNS TABLE
WITH SCHEMABINDING
AS
RETURN
SELECT 1 AS SecurityPredicateResult
FROM dbo.Customers AS c
 CROSS APPLY(
  SELECT TOP 1 1
  FROM dbo.Employees AS e
  WHERE c.SalesRegion = e.SalesRegion
   AND (e.EmployeeName = USER_NAME()
      OR USER_NAME() IN (N'SalesManager', N'dbo')))
 AS E(EmployeesResult)
WHERE c.CustomerId = @CustomerId;
GO
```

The next step is, of course, to add a security policy. Note that you need to use a column from the dbo. Customers table for the argument of the predicate function. This argument is a dummy one, and does not filter the rows; the actual filter is implemented in the body of the function. The following code illustrates this process:

```
CREATE SECURITY POLICY CustomersFilter
ADD FILTER PREDICATE Security.CustomersRLS(Cus-
tomerId)
ON dbo.Customers
WITH (STATE = ON);
GO
```

The following queries test the filter predicate:

```
SELECT * FROM dbo.Customers;
EXECUTE (N'SELECT * FROM dbo.Customers') AS
USER = N'SalesUser1';
EXECUTE (N'SELECT * FROM dbo.Customers') AS
USER = N'SalesUser2';
EXECUTE (N'SELECT * FROM dbo.Customers') AS
USER = N'SalesUser3';
EXECUTE (N'SELECT * FROM dbo.Customers') AS
USER = N'SalesManager';
GO
```

The rows from the dbo.Customers table are filtered for regular users. However, note that SalesUser1 and SalesUser2 see the same rows—the rows for the customers from the USA—because the sales territory for both of them is USA. Now let's give users permissions to modify the data in the dbo.Customers table, using the following code:

```
GRANT INSERT, UPDATE, DELETE ON dbo.Customers
TO SalesUser1, SalesUser2, SalesUser3,
SalesManager;
```

Try to impersonate the SalesUser1 user, and delete or update a row that SalesUser1 does not see because of the filter predicate. In both cases, zero rows are affected. The code for this is given here:

```
EXECUTE (N'DELETE FROM dbo.Customers WHERE
CustomerId = 3')
 AS USER = N'SalesUser1';
EXECUTE (N'UPDATE dbo.Customers
 SET CustomerName =' + '''' + 'Updated' + '''' +
  'WHERE CustomerId = 3')
 AS USER = N'SalesUser1';
```

However, SalesUser1 can insert a row that is filtered out when the same user queries the data. In addition, the user can also update a row in such a way that it disappears from the user's scope. The following code illustrates this process:

```
EXECUTE (N'INSERT INTO dbo.Customers
(CustomerId, CustomerName, SalesRegion)
 VALUES(5, ' + '''' + 'Customer05' + '''' + ',' +
  '''' + 'EU' + '''' + ');'
 ) AS USER = N'SalesUser1';
EXECUTE (N'UPDATE dbo.Customers
 SET SalesRegion =' + '''' + 'EU' + '''' +
 'WHERE CustomerId = 2')
 AS USER = N'SalesUser1';
```

Now try to read the data, using the following code. The dbo user sees all of the rows, while SalesUser1 sees neither the row(s) he just inserted nor the row(s) he just updated:

```
SELECT * FROM dbo.Customers;
EXECUTE (N'SELECT * FROM dbo.Customers') AS
USER = N'SalesUser1';
```

Block Predicates

You need to add a block predicate to block the inserts and updates that would move a row outside the scope of the user performing the write operation:

```
ALTER SECURITY POLICY CustomersFilter
ADD BLOCK PREDICATE Security.CustomersRLS(Cus-
tomerId)
ON dbo.Customers AFTER INSERT,
ADD BLOCK PREDICATE Security.CustomersRLS(Cus-
tomerId)
ON dbo.Customers AFTER UPDATE;
GO
```

Try to do similar data modifications while impersonating the SalesUser1 user again:

```
EXECUTE (N'INSERT INTO dbo.Customers
(CustomerId, CustomerName, SalesRegion)
 VALUES(6, ' + '''' + 'Customer06' + '''' + ',' +
  '''' + 'EU' + '''' + ');'
) AS USER = N'SalesUser1';
EXECUTE (N'UPDATE dbo.Customers
SET SalesRegion =' + '''' + 'EU' + '''' +
'WHERE CustomerId = 1')
 AS USER = N'SalesUser1';
```

This time, you get an error for both commands. You can see that the block predicate works. Finally, you can clean up your SQL Server instance:

```
USE master;
IF DB_ID(N'RLSDemo') IS NOT NULL
    ALTER DATABASE RLSDemo SET SINGLE_USER
WITH ROLLBACK IMMEDIATE;
  DROP DATABASE RLSDemo;
GO
```

Conclusion

Row-level security can be achieved in two way: through programmable objects and with the predicate-based approach. It is up to you to select the one that suits you. It is possible also to combine both approaches and use, for example, for some special groups of users programmable objects, while use for the majority of regular uses predicate-based RLS.

Gašper Rupnik
MCT, MS, MCSD, MCPS
gasper.rupnik@kompas-xnet.si

Telerik Reporting v okolju ASP.NET Core Blazor in Entity Framework

Danes bom govoril malo bolj o Telerik Reporting-u v okolju "novega" ASP.NET Core Frameworka – ampak ne golega, zraven bo pripeta še novejša tehnologija - Blazor Framework. Več o Blazor-ju mogoče v eni izmed naslednjih izdaj Pike, tokrat se bomo osredotočili zgolj na Telerik Reporting znotraj Blazorja, saj o tem trenutno na internetu ni veliko oz nič, še tistih nekaj nasvetov ki jih morebiti dobite, so bolj out-dated oz deprecated.

Za tiste ki ne poznate, **Telerik Reporting** skupaj s svojim Designerjem, je zelo dobro orodje za enostavno izdelavo poročil / reportov, ki jih lahko uporabljate v vaših namiznih ali spletnih aplikacijah. Več o tem si lahko preberete tukaj: https://www.telerik.com/products/reporting.aspx

Blazor je platforma v velikem razcvetu za vse tiste, ki "sovražite" client-side programiranje v **JavaScriptu, TypeScriptu** ali kateremkoli od znanih framework-ov. Omogoča vam zgraditi client-side aplikacijo v C# jeziku. Obstajata dve glavni "filozofiji" uporabe – **WebAssembly** (poganja vašo C# kodo direktno v brskalniku preko WebAssembly) ali **Server** (poganja vašo C# kodo na serverju, kjer server in klient komunicirata preko SignalR poti). Več o Blazorju si lahko preberete tukaj: https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor

Torej združimo oboje zgoraj, kjer kot glavno platformo uporabimo **ASP.NET Core**, za dostop do podatkov pa **Entity Framework Core**. Torej ta članek predpostavlja, da poznate vsaj malo vse zgornje štiri platforme, sicer bi lahko ta članek zavzel celotno Piko .

V vašem ASP.NET Core projektu, v katerem imate že pripet **Blazor Server** ter nameščen **Telerik UI For Bazor** si zaželimo imeti vzpostavljen in delujoč še **Telerik Reporting**.

Za začetek ustvarimo **package.json NPM** fajl, v katerem definiramo naslednje client-side pakete, ki jih bomo uporabljali v našem projektu - Telerik Reporting na Blazorju rabi za delovanje (prikaz in klicanje funkcionalnosti preko WebAPIja) JQuery, prav tako mu dodamo default temo. Sicer malo smešno, ker pomen Blazorja je ravno v tem, da se izognemo client-side kodi. Ampak ob tem je potrebno vedeti, da brskalniki še vedno znajo zgolj client-side jezike, tako da nekako jih še vedno rabijo.

```
{
  "version": "1.0.0",
  "name": "asp.net",
  "private": true,
  "devDependencies": {
    "@types/jquery": "^3.5.4"
  },
  "dependencies": {
    "@progress/kendo-theme-default": "^4.32.0",
    "jquery": "^3.5.1"
  }
}
```

Node_modules pot nastavimo kot statične datoteke naše ASP.NET Core aplikacije, ki bodo gostovanje na /**lib** naslovu:

```
app.UseStaticFiles(new StaticFileOptions()
{
    RequestPath = "/lib",
    FileProvider = new PhysicalFileProvider(Path.Combine(env.ContentRootPath, "node_modules"))
});
```

Za delovanje kode, ki jo bom spisal v nadaljevanju, rabite na projekt namestiti spodnje dva NuGet paketa:
Telerik.Reporting.Services.AspNetCore (za izdelavo svojega WebAPIja)
Telerik.ReportViewer.Blazor (za Blazor ReportViewer kontrolo)
V _Host.cshtml datoteko v <head> tag dodamo spodaj označene script in link tage.

```
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Web</title>
    <base href="~/" />
    <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
    <link href="css/site.css" rel="stylesheet" />

    <script src="_content/Telerik.UI.for.Blazor/js/telerik-blazor.js" defer></script>
    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="/api/reports/resources/js/telerikReportViewer"></script>

    <link rel="stylesheet" href="_content/Telerik.UI.for.Blazor/css/kendo-theme-default/all.css" />
    <link rel="stylesheet" href="lib/@@progress/kendo-theme-default/dist/all.css" />
    <link href="https://fonts.googleapis.com/css?family=Open+Sans:300,400,600,700,800" rel="stylesheet">
</head>
```

Na konec <body> tag-a pa vstavite spodaj označeno skripto.

```
        <a class="dismiss">✕</a>
    </div>

    <script src="_framework/blazor.server.js"></script>
    <script src="_content/telerik.reportviewer.blazor/interop.js" defer></script>
</body>
</html>
```

Ker bomo za Telerik Reporting uporabili **svoj WebAPI**, kjer ga bomo prilagodili za delo z našim podatkovnim modelom v Entity Framework Core, je zgoraj tudi link do servisa relativen (**/api/reports/resources/js/telerikReportViewer**). V nasprotnem primeru bi lahko uporabili tudi official Telerik-ovega.
Torej, ustvariti moramo svoj WebAPI za Telerik Reporting. Ker uporabljamo ASP.NET Core (brez MVCja), je potrebno v našem projektu dodati **Controller** mapo, v kateri ustvarimo **ReportsController.cs** datoteko. Vanjo dodamo spodnjo kodo (PrevozService spodaj je injectan mikroservis, ki preko Entity Framework Core v našo aplikacijo "uvaža" podatke iz podatkovne baze):

```
[Route("api/reports")]
1 reference
public class ReportsController : ReportsControllerBase
{
    0 references
    public ReportsController(IReportServiceConfiguration reportServiceConfiguration, PrevozService prevozService)
        : base(reportServiceConfiguration)
    {
    }
}
```

Ker naša ASP.NET Core aplikacija še ni nastavljena za uporabo controllerjev, moramo v **Startup.cs** fajl dodati naslednje vrstice kode (prva slika se nanaša na dodajanje mikroservisov v **ConfigureServices** metodi, druga slika pa registracija middleware za routanje v controllerje znotraj **Configure** metode):

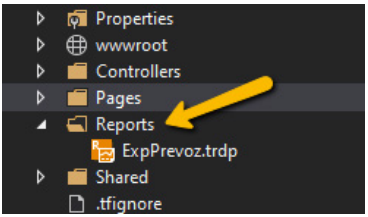
```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<SLOContext>(options =>
        options
            .UseLazyLoadingProxies()
            .UseSqlServer(
                Configuration.GetConnectionString("DefaultConnection")));

    services.AddControllers().AddNewtonsoftJson();

    services.AddRazorPages().AddNewtonsoftJson();

    services.AddServerSideBlazor().AddCircuitOptions(options => { options.DetailedErrors = true; });

    services.AddTelerikBlazor();

    services.Configure<IISServerOptions>(options =>
    {
        options.AllowSynchronousIO = true;
    });
```

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllers();
    endpoints.MapBlazorHub();
    endpoints.MapFallbackToPage("/_Host");
});
```

Ustvarimo nov Razor page z imenom **Test.razor.** V njemu pokličemo **ReportViewer** komponento (potreben tudi **using** stavek na vrhu), kjer kličemo lokalni service URL (tega, ki smo ga ravnokar ustvarili), ter povemo, da bomo poklicali report v fajlu **ExpPrevoz.trdp**, kateremu bomo podali še dva dodatna parametra, preko katerega bo report vedel, kateri podatek iz množice vseh naj vzame.

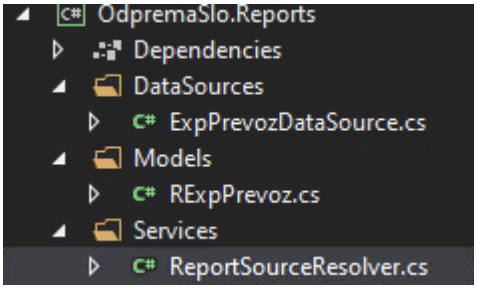
```
@page "/reports/test"
@using Telerik.ReportViewer.Blazor

<style>
    #rv1 {
        position: relative;
        width: 1200px;
        height: 600px;
    }
</style>

<ReportViewer ViewerId="rv1"
              ServiceUrl="/api/reports"
              ReportSource="@(new ReportSourceOptions()
                              {
                                  Report = "ExpPrevoz.trdp",
                                  Parameters = new Dictionary<string, object>
                                  {
                                      { "sifra", "10056" },
                                      { "plantId", (byte)13 }
                                  }
                              })"
              Parameters="@(new ParametersOptions { Editors = new EditorsOptions { MultiSelect = EditorType.ComboBox, SingleSelect = EditorType.ComboBox } })"
              ScaleMode="@(ScaleMode.Specific)"
              Scale="1.0" />
```

Report bomo kasneje ustvarili preko **Telerik Report Designer-ja**, ter ga shranili v mapo Reports znotraj našega projekta, ki jo bomo morali še kreirati.



Ker bomo v Telerik Reportu uporabili **ObjectDataSource**, moramo za to ustvariti znotraj našega Visual Studio Solutiona ločen **Class Library (.NET Standard)** projekt. Paziti morate, da je tipa **.NET Standard 2.0**, ker trenutno Telerik Reporting podpira le tega za naše ASP.NET Core aplikacije (tudi za to mora biti ločen projekt). V mojem primeru sem mu dal ime **OdpremaSlo.Reports**. V njem bom fajle grupiral v tri sklope:

- **DataSources** – v tem folderju bodo metode, ki bodo pripravljale podatke za naše Telerik Reporte preko parametrov, ki jih bodo dobile ob klicu v ReportViewer kontroli
- **Models** – v tem folderju bodo modeli, ki jih bomo importirali v naš Telerik Report Designer, preko katerega bo naš report dobil shemo podatkov
- **Services** – v tem folderju bomo ustvarili mikroservis za našo ASP.NET Core aplikacijo, kjer bo glavna stvar Resolve funkcija, ki bo znala poklicati pravi Report fajl (TRDP datoteko) in ji pripeti prave podatke (torej uporabiti pravi DataSource iz prve točke zgoraj).



Pojdimo najprej v naš **ReportSourceResolver**, kjer je njegova vsebina spodnja:

```
2 references
public class ReportSourceResolver : IReportSourceResolver
{
    private readonly PrevozService _prevozService;
    private readonly Assembly _ass;

    0 references
    public ReportSourceResolver(PrevozService prevozService)
    {
        _prevozService = prevozService;

        _ass = Assembly.GetExecutingAssembly();
    }

    0 references
    public ReportSource Resolve(string report, OperationOrigin operationOrigin, IDictionary<string, object> currentParameterValues)
    {
        var reportPackager = new ReportPackager();
        Report reportInstance = null;
        using (var sourceStream = System.IO.File.OpenRead("Reports/" + report))
        {
            reportInstance = (Report)reportPackager.UnpackageDocument(sourceStream);
        }

        Type dataSource = _ass.GetType("OdpremaSlo.Reports.DataSources." + report.Replace(".trdp", "") + "DataSource");
        MethodInfo method = dataSource.GetMethod("GetData");

        reportInstance.DataSource = method.Invoke(null, new object[] { _prevozService, currentParameterValues });

        InstanceReportSource instanceReportSource = new InstanceReportSource();
        instanceReportSource.ReportDocument = reportInstance;

        return instanceReportSource;
    }
}
```

Source resolver dobi injectan servis za povezavo do baze podatkov preko Entity Framework Core, ter preko Reflectiona vzame trenutni assembly. V Resolve metodi dobi podatek o klicanem report iz ReportViewer kontrole iz našega Test.razor fajla. Iz imena ugotovi, kateri DataSource naj vzame, ter datasourcu posreduje naprej parametri, ki so bili zraven pripeti.

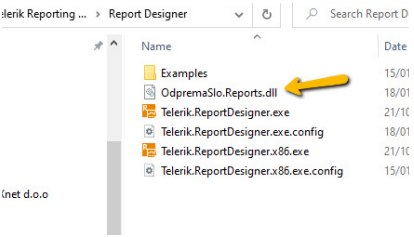
Torej **ExpPrevoz.**trdp report bo uporabil **ExpPrevoz**DataSource.cs datasource. V njem pokliče **GetData()** metodo, katera dobi podatke iz mikroservisa ter parametre, preko tega pa poišče prave podatke v bazi in jih vrne.

```
[DataObject]
0 references
public static class ExpPrevozDataSource
{
    [DataObjectMethod(DataObjectMethodType.Select)]
    0 references
    public static RExpPrevoz GetData(PrevozService prevozService, IDictionary<string, object> currentParameterValues)
    {
        ExpPrevoz prevoz = prevozService.GetPrevozDetails(currentParameterValues["sifra"].ToString(), Byte.Parse(currentParameterValues["plantId"].ToString()));

        // skopiraj samo tiste property-e, ki jih rabiš v reportu (report modelu)
        RExpPrevoz rPrevoz = new RExpPrevoz()
        {
            Sifra = prevoz.Sifra,
            Datum = prevoz.Datum,
            Prevoznik = prevoz.Prevoznik,
            Kamion = prevoz.Kamion
        };

        return rPrevoz;
    }
}
```

Ustvarimo še model podatkov **RExpPrevoz** v Models mapi:

```
3 references
public class RExpPrevoz
{
    1 reference
    public string Sifra { get; set; }
    1 reference
    public DateTime? Datum { get; set; }
    1 reference
    public string Prevoznik { get; set; }
    1 reference
    public string Kamion { get; set; }
}
```

Vse skupaj zbuildamo, ter **OdpremaSlo.Reports.dll** skopiramo na lokacijo, kjer imamo poinštaliran Telerik Report Designer:

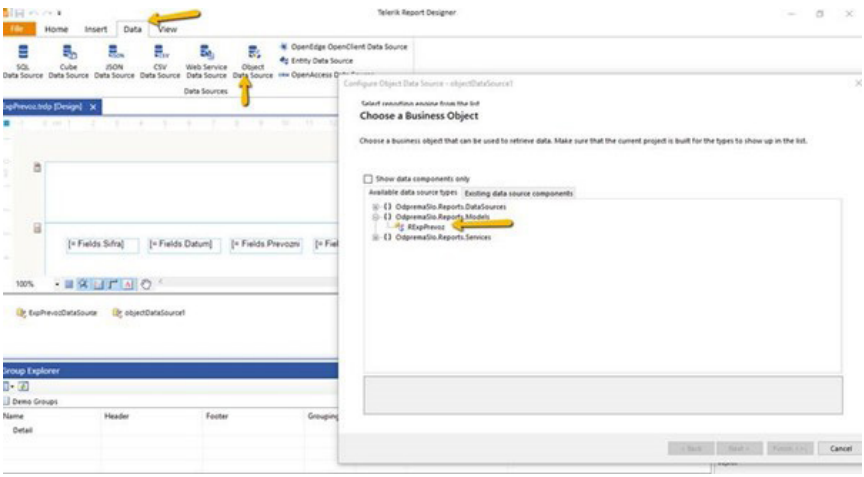


C:\Program Files (x86)\Progress\Telerik Reporting R3 2020\Report Designer

V **Telerik.ReportDesigner.exe.config** datoteki morate nastaviti, da bo le ta zagrabil zgoraj dodan assembly:

```
    <dependentAssembly>
      <!-- Required for interoperability with older versions of Telerik Reporting -->
      <assemblyIdentity name="Telerik.Reporting" culture="neutral" publicKeyToken="a9d7983dfcc261be"/>
      <bindingRedirect oldVersion="0.0.0.0-14.2.20.1021" newVersion="14.2.20.1021"/>
    </dependentAssembly>
  </assemblyBinding>
</runtime>

<connectionStrings>
  <add name="Telerik.Reporting.Examples.CSharp.Properties.Settings.TelerikConnectionString"
      connectionString="Data Source=(local);Initial Catalog=AdventureWorks;Integrated Security=SSPI"
      providerName="System.Data.SqlClient" />
</connectionStrings>

<Telerik.ReportDesigner DefaultWorkingDir="Examples">
</Telerik.ReportDesigner>

  <!-- Add assembly references -->
  <Telerik.Reporting>
      <AssemblyReferences>
          <add name="OdpremaSlo.Reports" version="1.0.0.0" culture="neutral" publicKeyToken="null" />
      </AssemblyReferences>
  </Telerik.Reporting>

  <system.diagnostics>
      <trace autoflush="true" indentsize="4">
          <listeners>
              <add name="myListener" type="System.Diagnostics.TextWriterTraceListener" initializeData="D:\Temp\T
              <remove name="Default" />
          </listeners>
      </trace>
  </system.diagnostics>
```

V Report Designerju nato lahko dodate **Object data source** kot prikazujem spodaj:



Iz tega dobite data source, kjer lahko property-e modela dodate na svoj report ter tako ustvarite **ExpPrevoz.trdp** datoteko, ki jo shranite v prej ustvarjeno **Reports** mapo.

Da bo tudi vaša ASP.NET Core aplikacija delovalo popolno, je potrebno v Startup.cs datoteki poregistrirati prej ustvarjeni ReportSourceResolver ter **ReportServiceConfiguration**, kateri bo bral podatek o temu, kateri assembly naj za report vzame iz appsettings.json datoteke:

```
services.AddScoped<UserService>();
services.AddTransient<PrevozService>();

services.AddTransient<SLOContextProcedures>();

services.TryAddSingleton<ConfigurationService>();

services.TryAddScoped<IReportSourceResolver, ReportSourceResolver>();
services.TryAddScoped<IReportServiceConfiguration>(sp => new ReportServiceConfiguration
{
    ReportingEngineConfiguration = sp.GetRequiredService<ConfigurationService>().Configuration,
    HostAppId = "OdpremaSlo.Reports",
    Storage = new FileStorage(),
    ReportSourceResolver = sp.GetRequiredService<IReportSourceResolver>()
});
```

Appsettings.json:

```
"telerikReporting": {
  "assemblyReferences": [
    {
      "name": "OdpremaSlo.Reports"
    }
  ]
},
"Serilog": {
  "MinimumLevel": {
    "Default": "Debug"
  },
```

Domen Gričar
MCSD, MCSA, MCT
domen.gricar@kompas-xnet.si

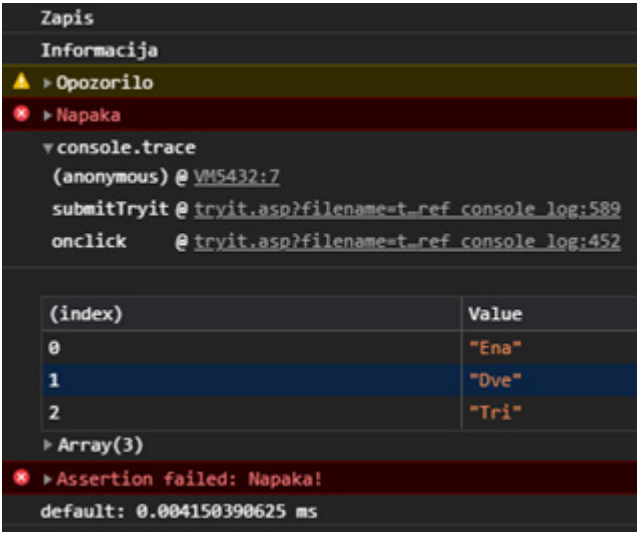
Chrome Orodja za razvijalce - Console

Pri razvijanju JavaScript-a je najenostavneje pregledovati kodo direktno v brskalniku, saj lahko sproti iščemo in popravljamo napake. Pri tem si lahko z zavihkoma Console in Sources v orodjih za razvijalce. V tem članku si bomo pogledali kako v konzolo zapisovati JavaScript sporočila in vrednosti, kako testirati kodo in kako urejati in vplivati na DOM strukturo strani. V naslednjem članku bom podrobneje opisal zavihek Sources in kako v njem razhroščujemo kodo.

```
console.log("Zapis");
console.info("Informacija");
console.warn("Opozorilo");
console.error("Napaka");
console.trace();
console.table(["Ena", "Dve", "Tri"]);
console.assert(false, "Napaka!");
console.time();
console.timeEnd();
```

Zavihek Console ima dva glavna namena, pregled sporočil in pisanje JavaScript stavkov. Sporočila so uporabna pri testiranju JavaScript kode. Ko si želimo zagotoviti da koda deluje pravilno, si lahko zapišemo komentarje v začetkih funkcij in zank, vrednosti spremenljivk, izrazov in vrnjenih vrednosti metod. Ko se JavaScript koda izvede, bodo vsi zapisi komentarjev izpisani v konzoli. Sporočila lahko zabeležimo na več različnih načinov. Najbolj pogosti so console.log(), ki zapiše običajno sporočilo v konzolo, console.info(), za zapis informativnega sporočila, console.warn(), zapiše opozorila (obarvana rumeno) in console.error(), ki zapiše napake (obarvano rdeče). V primeru, da želimo zapisati zbirke vrednosti, jih lahko zapišemo z console.table(), v obliki razpredelnice. Za zapis stack trace-a uporabimo console.trace(). console.assert() uporabimo, ko želimo zapisati napako le v primeru določenega pogoja. Če nas zanima koliko časa traja neka operacija, lahko pred začetkom izvajanja uporabimo console.time(), da začnemo merjenje časa in po zaključeni

operaciji console.timeEnd(), ki nam izpiše čas trajanja. V primeru, da želimo pobrisati vse zapise iz konzole pa uporabimo console.clear().



V zavihku lahko tudi zaganjamo JavaScript stavke, da preverimo delovanje funkcij ali za interakcijo z DOM strukturo strani. Lahko dodajamo in brišemo elemente na strani, jih oblikujemo in prestavljamo. Da poiščemo element lahko uporabimo običajne JavaScript metode, na primer document.getElementById(), ki v dokumentu poišče element po določenem id-ju. Za interakcijo z DOM strukturo strani imamo na voljo tudi Console Utilities API (deluje le v konzoli in ne v JavaScript kodi), s katerim si lahko olajšamo delo s spremenljivkami in iskanjem elementov na strani. Preko Console Utilities API imamo na voljo bližnjice za izbiro rezultatov, $_ vrne vrednost zadnjega izraza, $0-$4 pa vrnejo zadnjih pet DOM elementov, ki smo jih pregledali oziroma poiskali z JavaScriptom. $0 vrne zadnjega in $1 vrne predzadnjega. Namesto document.querySelector() funkcije lahko uporabimo skrajšano obliko $() (ki po obliki spominja na jQuery, vendar nima na voljo funkcij), ki izbere prvi element, ki ustreza selektorju. V primeru, da želimo izbrati vse elemente, ki ustrezajo izbranim pogojem uporabimo selektor $$() na primer $$('img'), ki vrne zbirko vse img elementov (ta selektor je krajša oblika document.querySelectorAll()). Selektor $x() nam vrne vse DOM elemente pri katerih se ujema XPath izraz, na primer $x('//div[img]') vrne vse div elemente, ki vsebujejo img element. Če nas zanimajo lastnosti objekta lahko uporabimo dir(), ki prikaže seznam vseh lastnosti objekta.

V prejšnjem članku sem opisal zavihek Elements, ki prikazuje DOM strukturo strani. Če želimo pregledati izbrani element v DOM strukturi lahko uporabimo funkcijo inspect(), za primer lahko vzamemo inspect(document.body), ki nas prestavi na zavihek Elements in pregleda body tag dokumenta.

Zavihek Console je pomembno orodje pri razhroščevanju JavaScript kode in nam zelo olajša razvoj in testiranje strani. Že najosnovnejše funkcije pripomorejo k enostavnejšemu testiranju aplikacij in iskanju napak na strani, če pa uporabljamo tudi naprednejše, je lahko razvoj bistveno hitrejši.

Andraž Bergant
andraz.bergant@kompas-xnet.si

Kako delujejo prevajalniki (ang. Compilers)?

Programski jeziki so bili narejeni zato, da lahko razvijalci pišejo in berejo bolj človeku razumljivo kodo. Problem nastane, ker računalniki razumejo samo strojni jezik, katerega pa ljudje zelo težko razumejo oziroma berejo. Zato prevajalniki, prevedejo kodo napisano z nekim programskim jezikom, ter jo pretvorijo v računalniku razumljivo obliko.

Kot smo že omenili se pri prevajanju pretvarja visoko nivojska koda v nizko nivojsko strojno kodo katero lahko računalnik izvrši. Zelo pomembna vloga prevajalnikov je prav tako opozarjanje razvijalcev na napake še posebno sintaksne napake.

Proces prevajanja je sestavljen iz več faz:

1. Leksikalna analiza
2. Sintaksna analiza
3. Semantična analiza
4. Generiranje vmesne kode
5. Optimizacija
6. Generiranje strojne kode

1. Prva faza prevajanja je leksikalna analiza. Tekom te faze prevajalnik razdeli izvirno kodo v delce, ki se imenujejo leksemi. Leksem je abstraktna enota določenega leksikalnega jezičnega sistema. Npr: string pozdrav = 'zdravo'; Tukaj imamo 5 leksemov:

1. string
2. pozdrav
3. =
4. 'zdravo'
5. ;

Ko razdeli kodo v lekseme, ustvari neko zaporedje žetonov, kar pa je končni produkt leksikalne analize. Zato se velikokrat v literaturi imenuje žetoniranje. Žeton je objekt, ki opiše leksem. Le ta poda informacije o namenu leksema, kot npr. ali je privzeta beseda, spremenljivka, ime, string,…

2. Med sintaktično analizo prevajalnik uporabi zaporedje žetonov, ki so bili generirani pri prvi fazi. Žetoni se uporabijo za strukturo, ki se imenuje abstraktno sintaksno drevo, ki je drevo katero predstavlja logično strukturo programa. V tej fazi prav tako prevajalnik preveri gramatično strukturo izvorne kode in sintaktične napake. Vsaka sintaktična napaka povzroči prekinitev prevajanja na kar nas opozori oziroma nam izpiše v konzoli tudi prevajalnik.

3. V semantični fazi prevajalnik uporabi abstraktno sintaktično drevo za detekcijo semantičnih napak, kot so npr. pripis napačnega tipa neki spremenljivki, deklariranje spremenljivk z istim imenom pod istim obsegom, uporabiti neko spremenljivko katere nismo deklarirali ali pa na primer poimenovali spremenljivko po neki že zasedeni ključni besedi programskega jezika,… Tako lahko semantično analizo razdelimo na tri korake kot so: preverjanje tipa, preverjanje kontrolnega teka in validiranje identifikatorjev. Da izvedemo vse zgoraj navedene korake med semantično analizo, prevajalnik izvede popolni prehod abstraktnega sintaktičnega drevesa. Semantična analiza končno proizvede neko komentirano abstraktno sintaktično drevo, katero opiše vrednosti njegovih atributov.

4. Po sintaksni in semantični analizi izvorne kode, veliko prevajalnikov generira nizko nivojske oziroma strojne vmesne kode, katere si lahko predstavljamo kot nek program, ki je namenjen abstraktnemu stroju.

5. V fazi optimiziranja prevajalnik uporabi raznolike načine, da izboljša učinkovitost kode. Faze optimiziranja naj bi sledile trem pravilom: rezultat ne

sme spremeniti originalnega pomena programa oziroma njegove logike, tekom optimiziranja se mora fokusirati na manjšo porabo virov in po čim hitrejšem izvajanju operacij ter, da samo optimiziranje ne vpliva močno na celoten čas prevajanja.

6. Na koncu prevajalnik pretvori optimizirano vmesno kodo v strojno kodo, ki je namenjena našemu računalniku, ta faza se imenuje generiranje strojne kode. Končna koda mora imeti isti pomen, kot izvorna koda ter biti učinkovita z vidika uporabe spomina in centralno procesne enote.

Powershell kotiček

Aleš Lipušček
MCP, MCTS, MCITP
ales.lipuscek@kompas-xnet.si

Včasih se primeri, da v naših skriptah pride do okoliščin, ki zahtevajo povečane pravice, in če na to nismo računali, bodo seveda težave. Le teh se lahko lotimo s predpripravo (to je s predhodnim čekiranjem in načrtovanjem) ali pa s sprotnim lovljenjem napak in reagiranjem na le-te. Če se spomnimo iz enega od prejšnjih člankov, nekatere napake se da »preživeti«, druge pa bodo našo skripto dokončno ustavile.

Dobra skripta mora znati predvidevati tudi nepredvidljive napake :P, jih ujeti in javiti uporabniku. S prvo verzijo PowerShella smo dobili Trap, katerega značilnost je, da ga definiramo na koncu skripte, in ki »polovi« vse napake (t.j. vsako izvajanje, ki pripelje do napake, se konča v Trapu. Trap ima naslednjo obliko
Trap TipNapake { koda; akcija}

Tip napake določa, katere napake se ujamejo vanj, če pa ni določen, se vanj ujamejo vse. Koda je skupek ukazov, ki naj se izvedejo, ko do napake pride, akcija pa so lahko Break, Continue ali Return. Slednji je tudi privzeta akcija, ke le ta v opisu Trapa ni določena.
Z verzijo 2 smo dobili boljšo možnost s konstruktom Try / Catch / Finaly, pri čemer bo Try del skušal izvesti kodi, pri kateri lahko pride do napake. Če do te le pride, jo najbližji Catch blok ujame in obdela (vsak Try ima lahko samo en Catch blok, PowerShell pravzaprav ztraja, da ima lahko samo enega. Seveda pa lahko na koncu definiramo še blok Finaly, ki se bo izvedel neglede na to ali je prišlo do napake ali ne in je idealno mesto, na katerem lahko popravimo vse, kar se je neljubega zgodilo.

```
try
{ throw "opa,težave"}
catch
{ Write-Error $_
try
{ Test-System }
catch [Management.Automation.CommandNotFoundException]
{ "Where's the $($_.TargetObject) command?" } }
finally
{ "no, saj ni bilo hudega"}
```

Najbolje je seveda preveriti, če pravice res potrebujemo in v primeru manjka elegantno zaključiti skripto.
$identity = [Security.Principal.WindowsIdentity]::GetCurrent()
$principal = New-Object Security.Principal.WindowsPrincipal $identity $principal.IsInRole([Security.Principal.WindowsBuiltinRole]::Administrator) }

Uporabimo lahko pa tudi posebno vrsto komentarja, tj stavek #Requires s parametrom RunAsAdministrator. Ne moremo ga uporabljati v funkcijah, cmdletih in snapinih, in mora biti prvi element v vrstici.

Jože Markič
MCT, IT arhitekt, sistemski inženir, predavatelj
joze.markic@kompas-xnet.si

Azure podatkovni centri

Zaradi vedno večjega zanimanja in potreb po gostovanju storitev na Azure platformi, je Microsoft v zadnjih mesecih prejšnjega leta napovedal kar nekaj novih lokacij, tudi po Evropi. V januarju 2021 je slika Microsoft Azure podatkovnih centrov za Evropo približno taka:

Originalni lokaciji za Evropo sta Severna (Irska) in Zahodna (Nizozemska) Evropa, ki imata v naboru storitev za končne stranke tudi najbolj razširjeno ponudbo. Na teh dveh lokacijah lahko zaradi njune starosti izbiramo tudi med tremi ali celo štirimi generacijami virtualnih strežnikov iste serije.

Npr. ena izmed bolj pogosto uporabljenih strežniških serij je serija D, namenjena za vsestranske strežnike, ki nimajo posebno velikih obremenitev na posamezni komponenti (pomnilnik, disk, mreža, procesor, grafična kartica). Ta serija je trenutno na voljo v štirih različnih generacijah, kjer so na izbiro tako konfiguracije z AMD kot tudi z Intel procesorji (D, D v2, D v3 in D v4). Dodatno ima serija D na voljo še optimizirane konfiguracije za specifične scenarije (D*, D*a, D*as, D*d, D*ds, D*s), ki pa vseeno ne zahtevajo visoko zmogljivih komponent. Kombinacija različnih generacij z različnimi konfiguracijami virtualne strojne opreme nam da na voljo zelo veliko različnih možnosti za postavitev naših strežnikov. Ko dodatno ugotovimo, da se cene med temi konfiguracijami razlikujejo ne samo glede na količino posameznih komponent (pomnilnik, procesor, diski,…), ampak tudi glede na generacijo virtualne strojne opreme

in lokacijo le-te, postanejo ocene stroškov še toliko bolj kompliciranе.

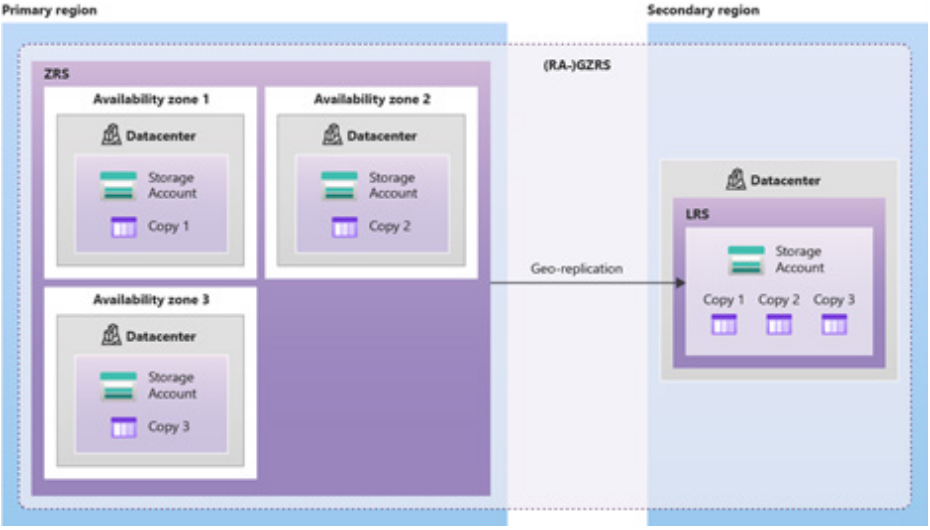
Kaj torej izbrati - katero lokacijo, katero velikost, katero generacijo? Odvisno… Če zaenkrat ostanemo samo pri lokaciji, bi bil dober začetek izbira tiste lokacije, ki nam je najbližja, ponuja vse storitve, ki jih potrebujemo (niso vse storitve na voljo na vseh lokacijah), hkrati pa je tudi finančno najbolj ugodna. Žal seveda dostikrat ni tako in moramo najti primeren kompromis. Če nam je odzivnost sistemov ključna, bomo izbrali lokacijo, do katere imamo najboljšo možnost hitre in zanesljive povezljivosti. Na hitro lahko to ocenimo z obiskom spletne strani http://azurespeed-test.azurewebsites.net/, ki naredi test latence med našim spletnim brskalnikom in diski na različnih lokacijah:

Data Center	Average Latency	History
Switzerland West	47ms	
Switzerland North	47ms	
France Central	54ms	
Germany North	56ms	
South UK	58ms	
West Europe	60ms	
West UK	61ms	
North Europe	61ms	

V mojem primeru lahko iz slike vidimo, da če bi nas zanimala samo odzivnost pri dostopu do naših gostovanih storitev (npr. spletna stran) iz moje trenutne lokacije, bi za Azure lokacijo izbrali podatkovne centre v Švici. Ko začnemo delati finančno analizo, pa hitro ugotovimo, da je ta lokacija tudi ena izmed najdražjih v Evropi. Ker je Švica ena izmed novejših lokacij, ne bomo mogli postavljati strežnikov s starejšimi generacijami, hkrati pa zaradi svoje »majhnosti« tudi ne podpira marsikatere storitve, ki jo imamo na voljo npr. v Zahodni Evropi.

Iz prve slike lahko vidimo, da je za podatkovni center v Severni Švici (Zürich) napovedana podpora za Availability Zone (AZ) – iz spletne dokumentacije ugotovimo, da bo AZ tu na voljo letos, v 2021. AZ je v Severni in Zahodni Evropa na voljo že kar nekaj časa, podobno tudi v Južni Angliji in na nekaterih drugih lokacijah po Evropi.

Zakaj bi nas zanimala podpora za AZ pri izbiri lokacije? Availability Zones so samostojne fizične lokacije z neodvisno elektriko, mrežo in hlajenjem, ki so med seboj dovolj oddaljene, da izpad ene ne pripelje do izpada celotne regije, hkrati pa so si tudi dovolj blizu, da imamo med njimi izredno hitre visoko propustne povezave, sinhrono replikacijo diskov,…. Administrativno imamo na voljo na posamezni lokaciji tri AZs, v praksi pa jih je lahko tudi več. Microsoft namreč ob polni zasedenosti obstoječega podatkovnega centra postavi novega nekje v bližini, a hkrati tudi dovolj daleč, da ustreza zahtevam za AZ. S časom (in pričakovano povečano porabo) bo torej lokacij, ki bodo podpirale AZs samo še več. Če se odločimo za postavitev servisa (npr. domenski strežniki) z dvema ali več strežniki, razporejenimi po AZs lokacijah, bomo imeli za ta servis zagotovljeno 99,99% razpoložljivost (dva strežnika na eni lokaciji brez AZ in z ustrezno konfiguracijo imata največ 99.95% SLA; en strežnik s premium SSD ali ultra diski ima 99,9% SLA; en strežnik s standard SSD diski ima 99,5% SLA; en strežnik s standard HDD diski ima zagotovljeno 95% razpoložljivost). Poleg strežnikov je izbira lokacije z AZs lahko pomembna tudi za virtualne diske. V osnovi je namreč vsak podatek na Azure diskih zapisan sinhrono 3x (tri kopije) na isti fizični lokaciji – to je Locally redundant storage (LRS), ki je tudi edina možnost pri uporabi premium SSD in ultra diskov. LRS nam zagotavlja razpoložljivost 99.999999999% (11 devetk). Če namesto LRS izberemo Zone-redundant storage (ZRS), se podatki sinhrono zapišejo na tri različne fizične lokacije (na tri AZs), s čimer pridobimo tudi višjo razpoložljivost 99.9999999999% (12 devetk). Če nam to še ni dovolj, pa se lahko odločimo tudi za Geo-redundant storage (GRS), kjer dobimo tri sinhrone kopije na primarni lokaciji (LRS) in dodatno asinhrono še tri kopije na sekundarni lokaciji (LRS), ki je nekaj sto kilometrov stran. S tako konfiguracijo nam Azure zagotavlja razpoložljivost 16 devetk. Če primarna lokacija podpira AZs, pa obstaja še ena možnost - Geo-zone-redundant storage (GZRS), kjer se podatki najprej trikrat sinhrono zapišejo na tri fizične lokacije v primarni regiji (AZs) in nato z zamikom (asinhrono) še trikrat sinhrono na sekundarni fizični lokaciji. Tudi GZRS nam omogoča razpoložljivost 16 devetk. Podatki, shranjeni na GZRS disku:



Seveda tudi to še ni vse, kar nam diski omogočajo pri replikaciji. Trenutno obstaja še RA-GRS in RA-GZRS. Glede na trenuten trend, v prihodnosti verjetno še kaj več. Kaj nam te možnosti ponujajo pa morda pogledamo naslednjič.

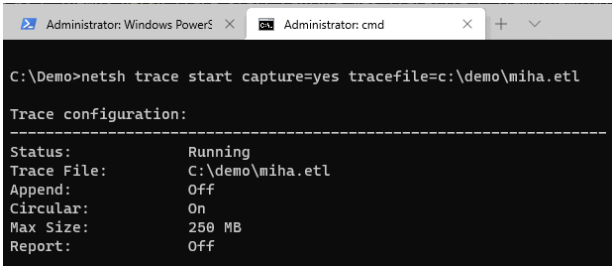
Če bi želeli izvedeti še kaj več o Azure infrastrukturi, podatkovnih centrih, virtualkah,… vas vabim, da se mi pridružite na kakšnem brezplačnem poslovnem zajtrku, SloWUG predavanju ali pa kar na Azure tečaju.

Mrežni zajem podatkov (network capture)

Miha Pihler
MCT, MCM, MVP
miha.pihler@telnet.si

Med nepogrešljiva orodja za sistemske skrbnike zagotovo sodijo orodja, ki omogočajo mrežni zajem podatkov (network capture). Med temi orodji se visoko uvrščajo Wireshark pa tudi Microsoft Message Analyzer, ki pa ga je Microsoft (žal) ukinil proti koncu leta 2019.

Že dolgo poznani način za zajem podatkov deluje preko ukaza netsh, kar omogoča zajem podatkov brez namestitve programov kot je Wireshark.
Primer zagona zajema mrežnega prometa z orodjem netsh



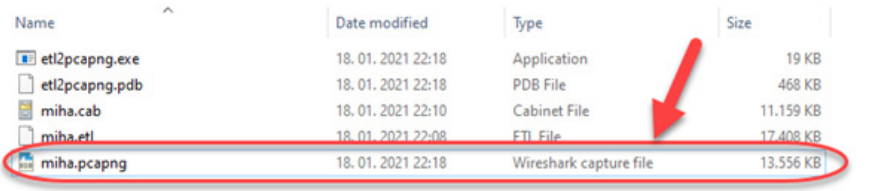
Ko smo zajeli želeni promet, ustavimo zajem z ukazom netsh trace stop

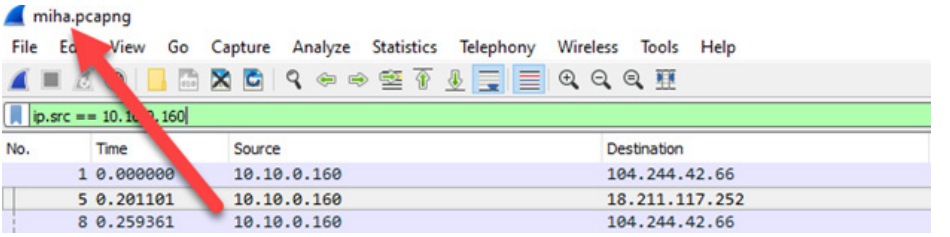
Dobra lastnost orodja Microsoft Message Analyzer-ja je bila, da je lahko neposredno odprl tako pridobljene podatke. Žal temu ni tako, če želimo uporabiti orodje Wireshark. Preden lahko podatke uporabimo v Wireshark-u, jih moramo konvertirati.
Za pretvorbo pridobljene ETL datoteke v obliko pcapng lahko uporabimo orodje Etl2pcapng.

exe, ki ga najdemo na GitHub-u (etl2pcapng) oz. v spletni iskalnik vpišemo etl2pcapng :-). Orodje lahko uporabimo tako, da vpišemo ime vhodne datoteke (npr. miha.etl), ki smo jo pridobili s pomočjo orodja netsh, ter ime izhodne datoteke (npr. miha.pcapng).

```
C:\Demo>etl2pcapng.exe miha.etl miha.pcapng
IF: medium=eth  ID=0    IfIndex=3
Converted 26601 frames

C:\Demo>
```

Zdaj lahko novo datoteko odpremo v Wireshark-u z dvoklikom.





Želim vam veselo lovljenje mrežnih paketkov :-)

GLOBAL CLOUD SKILLS v-TOUR

Ljubljana | 16 Marec 2021

Na zemljevid C3 Global Cloud Skill v-tour smo z velikim ponosom in odgovornostjo dodali tudi Slovenijo.

Vabimo vas **16. marca 2021** na virtualni dogodek pod sloganom **"DIGITAL IS ABOUT PEOPLE"**, ki je del globalne kampanje v sodelovanju z mednarodnim združenjem izobraževalnih centrov - LLPA, katerega predstavnik za Slovenijo je Kompas Xnet.

Na izobraževalnem dogodku, namenjen vodjem, kadrovskim specialistom in odločevalcem na področju izobraževanj, vodjem tehnoloških služb, IT strokovnjakom , se bomo s strokovnjaki pogovarjali kako graditi visoko zmogljive ekipe in bolje izkoristiti naložbe v oblaku. Izvedeli boste več o najnovejših usposabljanjih in certificiranjih, pobudah za usposabljanje, potrebnih veščinah, učnih poteh, dobrih praksah.

Med drugimi bo Miha Pihler predaval tudi o varnosti v oblaku, Jože Markič pa o upravljanju Azure virtualk.

Prijava

CLOUD SKILLS CUP TOURNAMENT SLOVENIA

The Leading Learning Partners Association (LLPA) in Kompas Xnet iščeta najboljšega Cloud, Data & AI in Developer strokovnjaka! Si to morda prav vi?

LLPA, ki mednarodno združuje 33 članov iz 55 držav, znova išče najboljše Cloud, Data&AI in Developer strokovnjake na turnirjih C3 Cloud Skills Cup in svetovnem prvenstvu - World Championships.

C3 Cloud Skills Cup turnir je sestavljen iz 20 vprašanj s treh področij Cloud, Data&AI in Developer.

Tisti, ki bo v najhitrejšem času pravilno odgovoril na največ vprašanj bo razglašen za najboljšega Cloud, Data&AI ali Developer strokovnjaka v Sloveniji.

S tekmovanjem v Sloveniji pričnemo 16. marca 2021.

Zmagovalec v Sloveniji se bo udeležil svetovnega tekmovanja, kjer bo tekmoval proti najboljšim iz 15 drugih držav članic LLPA, junija 2021.

Prijava

Petra Militarev
Vodja izobraževanj
petra.militarev@kompas-xnet.si

In se veselimo leta, ki prihaja

Na preteklo leto gledamo s hvaležnostjo do vseh, ki nam pomagate pri gradnji naše vizije. Leto nas je precej prevetrilo in postavilo pred zahtevne preizkušnje. Kljub vsem stiskam in nepričakovanim obratom, s katerimi smo se soočili, se z veseljem oziramo nazaj na vsa nova partnerstva, skupne uspehe in nepozabne trenutke ter upamo na še boljše leto 2021.

Uspelo nam je, da smo se hitro prilagodili in tako obdržali fokus na prenosu Microsoftovih znanj. Iz svetovnega nabora vrhunskih Microsoft partnerjev smo z združenjem LLPA postali Microsoft Partner leta 2020 na področju izobraževanj. Priznanje smo prejeli za izjemno kakovost in izjemen uspeh z različnimi formati izobraževanj na področju Microsoft Azure in Microsoft 365. In na to smo resnično ponosni.

Pripravili in izvedli smo številne live virtual tečaje. Nekateri udeleženci so imeli na začetku različne pomisleke o tovrstni obliki a ob koncu izobraževanja, so bili nad izvedbo pozitivno presenečeni.

Nove tehnologije in nenehno spreminjajoče se potrebe, spodbujajo številna podjetja k gradnji nenehne učne kulture. Digitalna doba zahteva nove veščine, ki jih morajo zaposleni pridobiti za preživetje in blaginjo organizacije. Prepričani smo, da je položaj Covid-19 deloval kot katalizator za to potrebo po nadgradnji in ponovnem izpopolnjevanju, saj so bila podjetja po vsem svetu prisiljena zapustiti pisarne in začeti delati od doma. Posledično so bila predstavljena nova orodja in platforme, zaposleni pa so se morali hitro naučiti, kako jih uporabljati in se prilagoditi novi resničnosti.

Dokler še ni omogočeno klasično izobraževanje, nadaljujemo z razpisanimi tečaji v live virtual obliki. V primeru, da se ukrepi sprostijo, vam bomo omogočili izbiro.

Live virtual tečaji

Tečaji za IT strokovnjake:

- Microsoft infrastruktura
- Microsoft razvoj
- Microsoft SQL
- Microsoft SharePoint
- Azure
- Data&AI
- Business applicatoin
- Masterclass izobraževanja

Tečaji za uporabnike:

- Microsoft Office
- Napredne delavnice

Cloud Fundamental tečaji

Spoznajte osnove oblaka in kako lahko koristi vašemu poslovanju. Kako oz kje začeti, je lahko potencialno zahtevno. Naj vas ne ustavi strah. Spoznajte osnove na enodnevnih tečajih, ki jih poleg javno razpisanih terminov izvajamo tudi za zaključene skupine.

DP 900: Microsoft Azure Data Fundamentals
AI-900: Microsoft Azure AI Fundamentals
PL-900: Microsoft Power Platform Fundamentals
MS-900: Microsoft 365 Fundamentals
AZ-900: Microsoft Azure Fundamentals

IZOBRAŽEVANJE PO MERI

Zaupajte nam vaše potrebe in poiskali bomo rešitve. Morda so to uporabniška Office znanja, uporaba Teams-ov, projektnega vodenja ali pa naprednejše, specialistične vsebine; portfelj.

Brezplačni poslovni zajtrki

Mesečno vam pripravljamo aktualne teme, ki jih v obliki spletnega zajtrka, prenašamo brezplačno. V kolikor ste zamudili predavanja, ki jih vodijo izvrstni strokovnjaki in predavatelji, vabljeni k vpisu v naš Youtube kanal, kjer si lahko zamujeno ogledate. Notri se trenutno nahajajo predavanja kot so: Funkcije za delo z dinamičnimi matrikami, Uvod v Azure za sistemce 1 del, Kaj zmore Power Platforma in Windows Server 2019.

PLANIRAJMO SKUPAJ

Veseli bomo, če nam zaupate vaše načrtovane izobraževalne plane. Okvirne termine za vaše zaključene skupine lahko že sedaj določimo in si s tem zagotovite tudi dodatne ugodnosti. Tudi zgodnje prijave in plačila prejeta najmanj 1 mesec pred pričetkom na MOC tečaje vam prinašajo 10% popust!

SA VOUCHERJI

Porabite svoje voucherje preden bo prepozno. Microsoft umika ugodnost SATV (Software Assurance Training Voucher) s katero ste se lahko brezplačno udeležili številnih MOC izobraževanj. S 1. februarjem 2021 tako ne bo več možnosti pridobivanja novih vavčerjev. **Po 30. juniju 2021 ne bo več možno kreirati** vavčerjev na izbrane tečaje. V kolikor imate še neuporabljene SATV, obrnite se na našo ekipo in pomagali vam bomo, da jih boste optimalno uporabili.

Ste morda to prav vi?

LLPA in in Kompas Xnet iščeta **najboljšega Cloud, Data & AI in Developer strokovnjaka!** C3 Cloud Skills Cup turnir je sestavljen iz 20 vprašanj s treh področij Cloud, Data&AI in Developer. Tisti, ki bo v najhitrejšem času pravilno odgovoril na največ vprašanj bo razglašen za najboljšega Cloud, Data&AI ali Developer strokovnjaka v Sloveniji. Zmagovalec v Sloveniji se bo udeležil svetovnega tekmovanja, kjer bo tekmoval proti najboljšim iz 15 drugih držav članic LLPA, junija 2021.Prijava

Vabimo vas, da se nam pridružite 16. marca 2021 na izobraževalnem dogodku C3 Global Cloud Skills v-Tour, ki bo letos pod sloganom »DIGITAL IS ABOUT PEOPLE" in ga organiziramo skupaj z LLPA. S strokovnjaki se bomo pogovarjali kako graditi visoko zmogljive ekipe in bolje izkoristiti naložbe v oblaku. Izvedeli boste več o najnovejšem usposabljanju in certificiranju, pobudah za usposabljanje, potrebnih veščinah, učnih poteh, dobrih praksah in še več.

Se vidimo!