Microsoft Azure – Message Queue

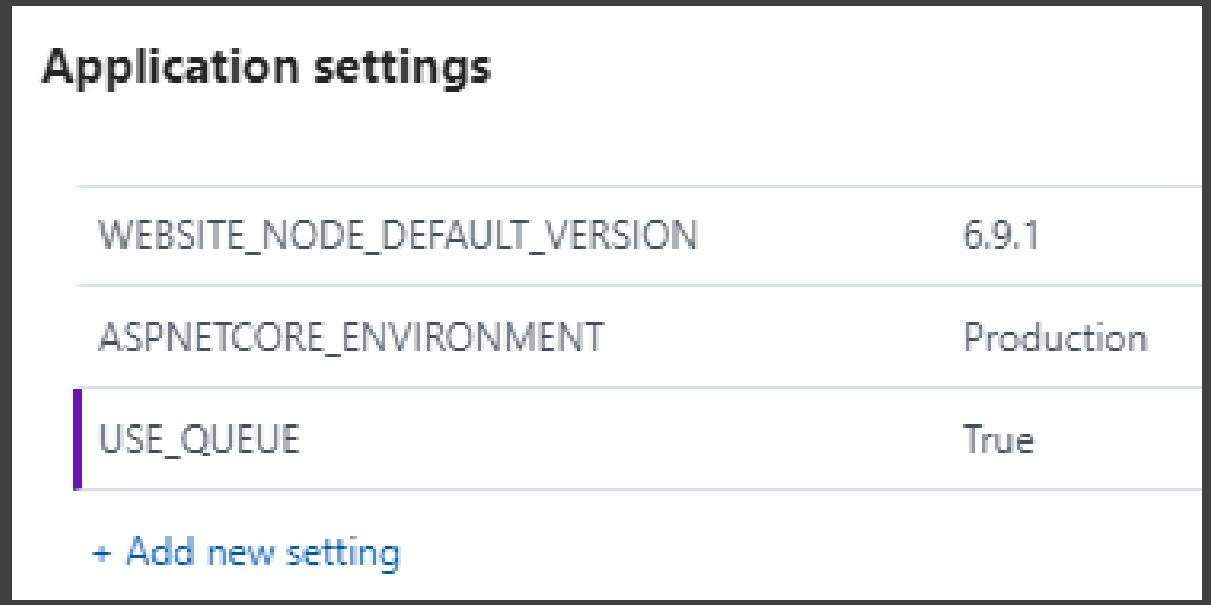
Milan Pekardy

pekardy@dcs.uni-pannon.hu

Exercise 1 – Modify the web app

- Modify the web application to process Todo creation requests with a message queue

1. Add an environment varible in the Azure portal:
   - App Services > AzureTodoWebApplication > Application settings > Add new setting…



Exercise 1 – Modify the web app

2. Add a queue service to the web application

```
public interface IQueueService
{
    2 references | 0 exceptions
    Task SendMessageAsync(Todo todo);
}
```

```
public class QueueService : IQueueService
{
    private CloudQueueClient _queueClient;

    private static readonly string todoQueueName = "todos";

    0 references | 0 exceptions
    public QueueService(CloudStorageAccount storageAccount)
    {
        _queueClient = storageAccount.CreateCloudQueueClient();
    }

    2 references | 0 exceptions
    public async Task SendMessageAsync(Todo todo)
    {
        CloudQueue queue = _queueClient.GetQueueReference(todoQueueName);
        await queue.CreateIfNotExistsAsync();

        var todoJson = JsonConvert.SerializeObject(todo);
        CloudQueueMessage message = new CloudQueueMessage(todoJson);

        await queue.AddMessageAsync(message);
    }
}
```

Exercise 1 – Modify the web app

3. Modify the TodoController's Create method to work based on the „USE_QUEUE" environment variable.

```
public async Task<IActionResult> Create([Bind("Name,Description,CreatedDate")] Todo todo, IFormFile photo)
{
    if (ModelState.IsValid)
    {
        todo.Id = Guid.NewGuid();
        todo.PhotoUrl = await _storageService.UploadPhotoAsync(photo);
        if (Environment.GetEnvironmentVariable("USE_QUEUE") == "True")
        {
            await _queueService.SendMessageAsync(todo);
        }
        else
        {
            _context.Add(todo);
            await _context.SaveChangesAsync();
        }
        return RedirectToAction(nameof(Index));
    }
    return View(todo);
}
```

Exercise 2 – Create a WebJob project

- Create a WebJob project to process the queue messages
  - Additional NuGet packages:
    - Microsoft.EntityFrameworkCore
    - Microsoft.EntityFrameworkCore.SqlServer
  - Right-click on the solution > Add > New Project… > Visual C# > Cloud > Azure WebJob
  - In the App.config file add your storage's connection string (AzureWebJobsDashboard, AzureWebJobsStorage),
  - Int the App.config file add your Azure SQL server's connection string (AzureSqlServer)
  - Add a Models folder with Todo.cs in it (same as in the web app)
  - Add a Data folder with TodoDbContext.cs in it (same as in the web app)

```
<configuration>
  <connectionStrings>
    <!-- The format of the connection string is "DefaultEndpointsProtocol=https;AccountName=NAME;AccountKe
    <!-- For local execution, the value can be set either in this config file or through environment varia
    <add name="AzureWebJobsDashboard" connectionString="DefaultEndpointsProtocol=https;AccountName=pekmilt
    <add name="AzureWebJobsStorage" connectionString="DefaultEndpointsProtocol=https;AccountName=pekmiltod
    <add name="AzureSqlServer" connectionString="Server=tcp:azure-test-db-server.database.windows.net,1433
  </connectionStrings>
```

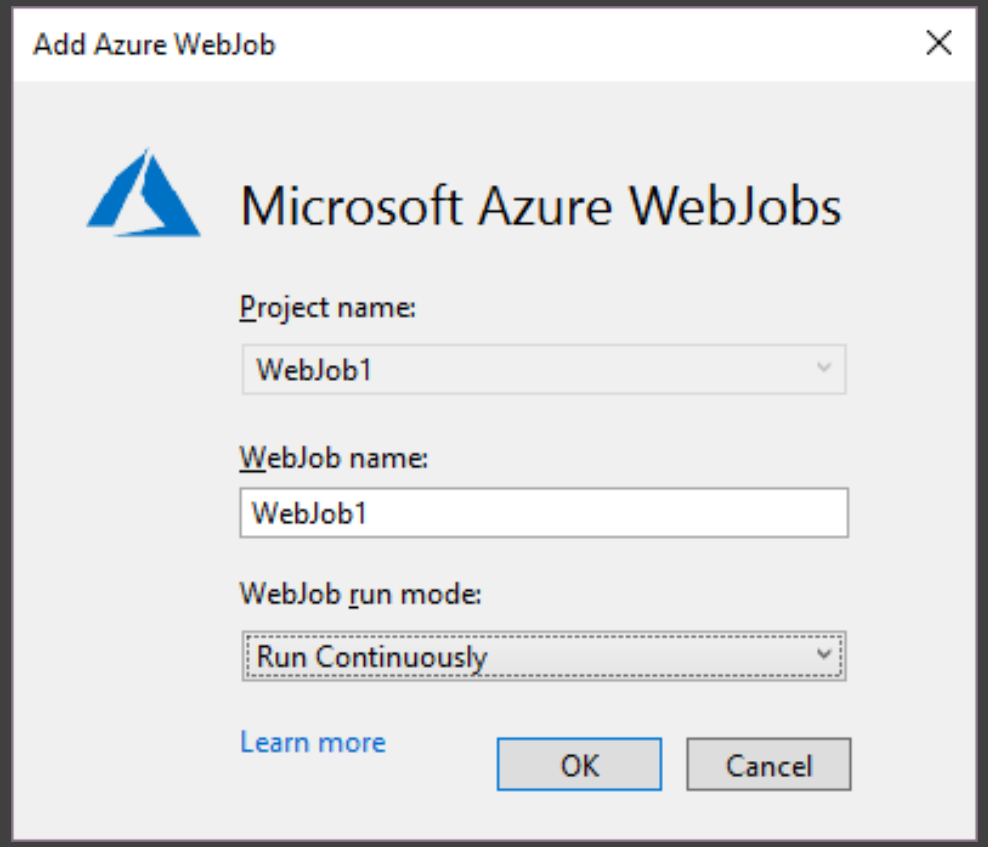
Exercise 2 – Create a WebJob project

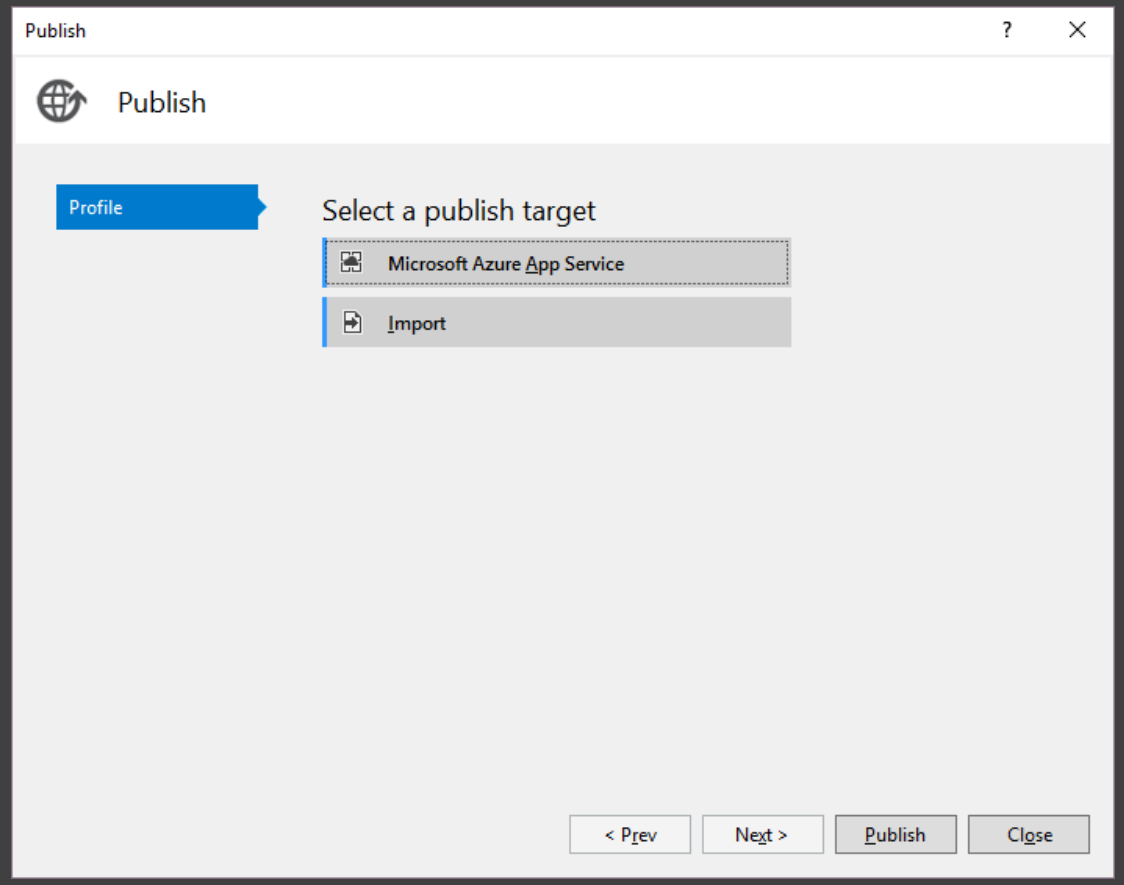
- In Functions.cs modify the ProcessQueueMessage method to process our Todo queue message:
  - Extract Todo entity from message object,
  - Insert Todo entity into the database.

```
public static void ProcessQueueMessage([QueueTrigger("todos")] CloudQueueMessage message, TextWriter log)
{
    log.WriteLine("TODOMSG: " + message.AsString);
    DbContextOptionsBuilder<TodoDbContext> builder = new DbContextOptionsBuilder<TodoDbContext>();
    builder.UseSqlServer(ConfigurationManager.ConnectionStrings["AzureSqlServer"].ConnectionString);
    using(TodoDbContext context = new TodoDbContext(builder.Options)){
        Todo todo = JsonConvert.DeserializeObject<Todo>(message.AsString);
        context.Add(todo);
        context.SaveChanges();
    }
}
```

Exercise 2 – Create a WebJob project

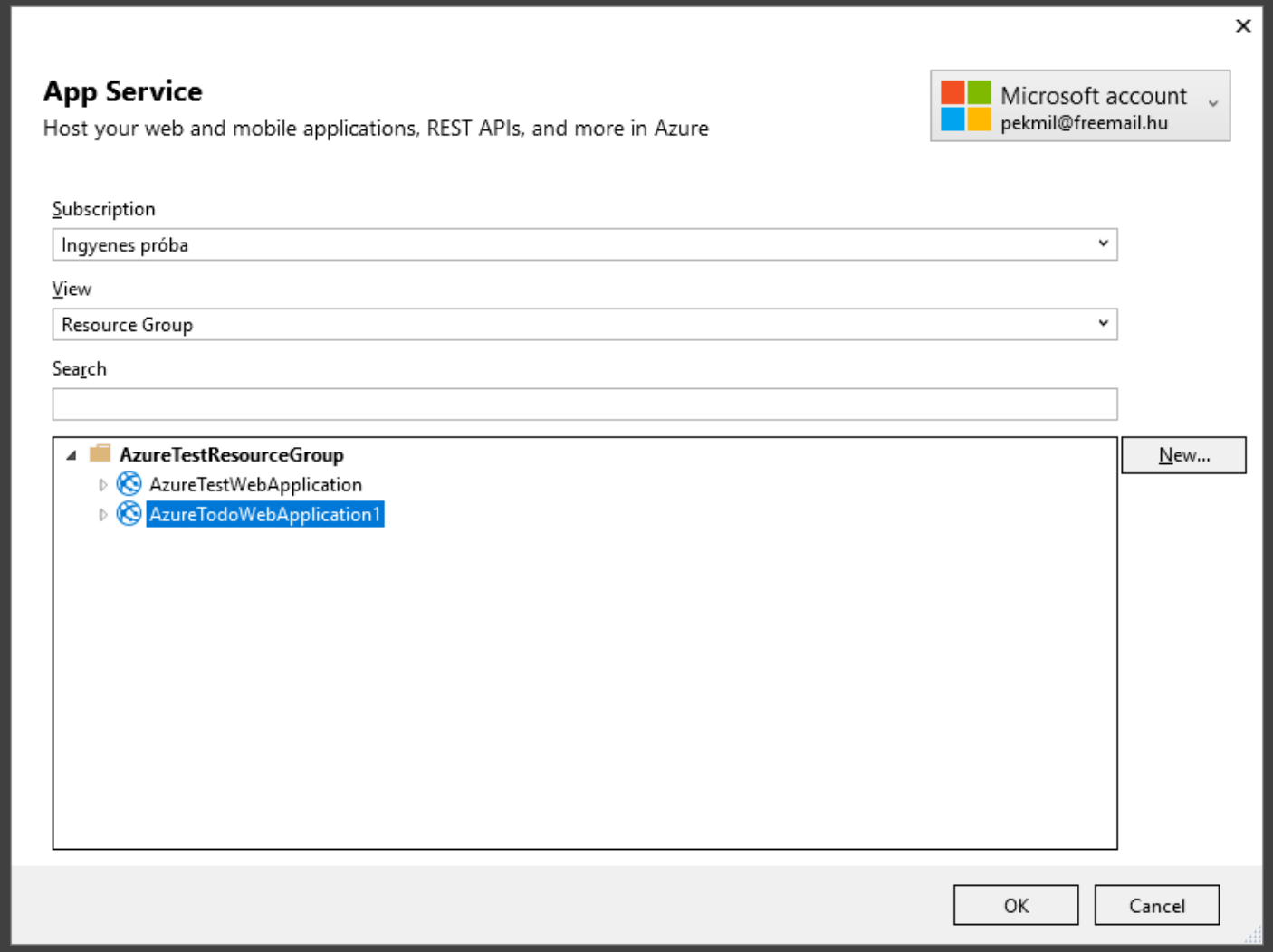
- Deploy the WebJob project:
  - Right-click on zhe project: Publish as Azure WebJob…
  - In the Publish dialog select a Microsoft Azure App Service publish target





Exercise 2 – Create a WebJob project

- In the App Service dialog select your existing resource group and web application



Exercise 2 – Create a WebJob project

- Check your message queue in your storage account:
  - Stop the WebJob
  - Set the „USE_QUEUE” environment variable to „True”
  - Ctreate a new todo in the web application
  - Check the message in your storage’s queue

Messages
todos

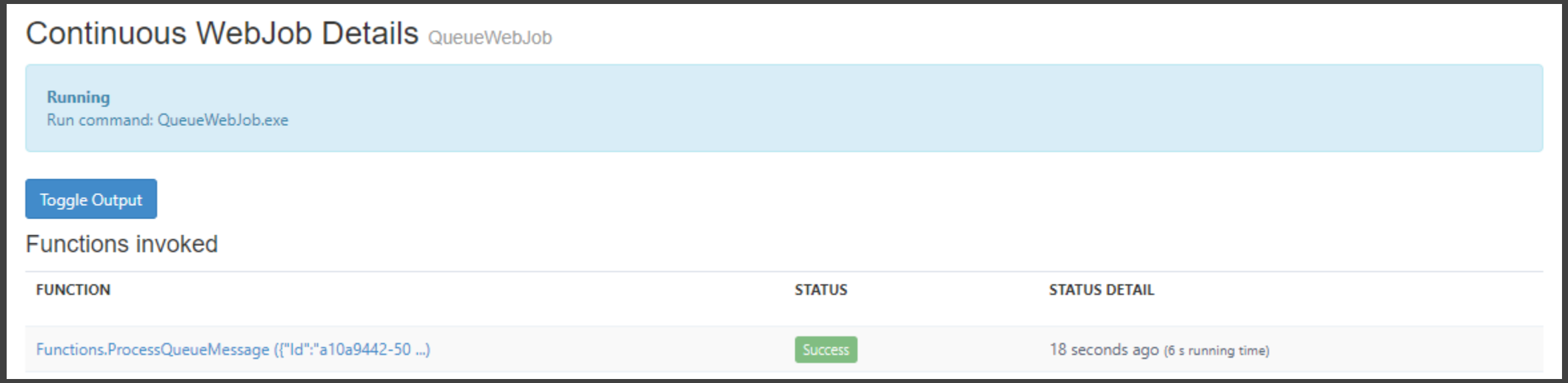
Refresh | Add message | Dequeue message | Clear queue

Search to filter items...

ID	MESSAGE TEXT	INSERTION TIME	EXPIRATION TIME	DEQUEUE COUNT
04e3f5e1-d0aa-4f9d-be...	{"Id":"a10a9442-5095-4746-a992-43edbfc08b55","Name":"Todo queue","Description":"Todo queue desc","Cr...	Sun, 04 Mar 2018 16:55:57 GMT	Sun, 11 Mar 2018 16:55:57 GMT	0

Exercise 2 – Create a WebJob project

- Restart the WebJob and check that the message got processed
  - App Services > AzureTodoWebApplication > WebJobs > Start
  - Check the WebJob's logs



Exercise 2 – Create a WebJob project

- The WebJob's message processor method got the message and inserted the new Todo entity into the SQL database
- Refresh the web application's Todo list page and check that you have the new Todo entity listed
- After successful message processing the message should be deleted from the queue