

# CLOUD COMPUTING SYSTEMS

## Lab 5

João Resende, Nuno Preguiça

(jresende\_at\_fct.un.pt, nuno.preguica\_at\_fct.unl.pt)

# GOAL

In the end of this lab you should be able to:

- Create Azure Functions in Java and deploy them in Azure Cloud Platform

# DOCUMENTATION

Azure documentation on writing function in Java:

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-reference-java>

Triggers and bindings:

Timer

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-timer?tabs=java>

HTTP

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-http-webhook>

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-http-webhook-trigger?tabs=java>

Blob storage

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-storage-blob>

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-storage-blob-trigger?tabs=java>

CosmosDB

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-cosmosdb-v2>

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-cosmosdb-v2-trigger?tabs=java>

# CREATE A PROJECT FOR THE FUNCTIONS

- **Alternative 1**

Use the example project available in CLIP.

- **Alternative 2**

Run the following command in an empty directory:

```
mvn archetype:generate \  
-DarchetypeGroupId=com.microsoft.azure \  
-DarchetypeArtifactId=azure-functions-archetype
```

# CONFIGURATION OF POM.XML

Some configurations need to be adjusted – functionAppName, functionAppRegion, functionPricingTier, functionResourceGroup.

```
<properties>
```

```
...
```

```
<functionAppName>sccfunwesteurope4204</functionAppName>
```

```
<functionAppRegion>westeurope</functionAppRegion>
```

```
<functionPricingTier>B1</functionPricingTier>
```

```
<functionResourceGroup>sc2223-rg-westeurope-4204</functionResourceGroup>
```

```
</properties>
```

# FUNCTIONS EXAMPLES

HttpTrigger

TimerTrigger

CosmosDBTrigger

BlobTrigger

# FUNCTIONS EXAMPLES

**HttpTrigger**

TimerTrigger

CosmosDBTrigger

BlobTrigger

# HTTP TRIGGER

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-http-webhook-trigger?tabs=java>

Defines a function that is called as the result of executing an HTTP request.

Trigger properties:

- **methods** : defines the HTTP methods  
E.g.: `methods = { HttpMethod.GET }`
- **authLevel** : defines the level of authentication necessary to run the function
- **route** : defines the path – URL will be: `{url}/api/{route}`  
E.g.: `route = "serverless/redis/{key}"`  
URL will be: `https://servername/api/serverless/redis/bla`
- **@BindingName** : allow to get part of the route path  
E.g.: `@BindingName("key") String key`



# HTTPTRIGGER (EXAMPLE)

```
@FunctionName("get-redis")
public HttpResponseMessage getRedis( @HttpTrigger(name = "req",
                                         methods = {HttpMethod.GET},
                                         authLevel = AuthorizationLevel.ANONYMOUS,
                                         route = "serverless/redis/{key}")
    HttpRequestMessage<Optional<String>> request,
    @BindingName("key") String key,
    final ExecutionContext context) {
    try (Jedis jedis = RedisCache.getCache().getJedisPool().getResource()) {
        String val = jedis.get(key);
        return request.createResponseBuilder(HttpStatus.OK)
            .body( "GET key = " + key + "; val = " + val)
            .build();
    }
}
```

ExecutionContext allows to access information about the invocation, including function name, invocation id, etc.  
<https://docs.microsoft.com/en-us/java/api/com.microsoft.azure.functions.executioncontext?view=azure-java-stable>

# FUNCTIONS EXAMPLES

HttpTrigger

**TimerTrigger**

CosmosDBTrigger

BlobTrigger

# TIMER TRIGGER

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-timer?tabs=java>

Defines a function that executes periodically.

Schedule format:

second minutes hours day-of-month month year

E.g.: 30 \*/5 \* \* \* \* means:

Every 5 minutes (\*/5)

at second 30

# TIMERTRIGGER (EXAMPLE)

```
@FunctionName("periodic-compute")
public void cosmosFunction( @TimerTrigger(name ="periodicSetTime",
                                         schedule = "*/20 * * * * *")
                           String timerInfo,
                           ExecutionContext context) {
    try (Jedis jedis = RedisCache.getCache().getJedisPool().getResource()) {
        jedis.set("serverlesstime", timerInfo);
    }
}
```

Check the written value using the HTTP redis get function on:  
[https://YOUR\\_SERVER.azurewebsites.net/api/serverless/redis/serverless-time](https://YOUR_SERVER.azurewebsites.net/api/serverless/redis/serverless-time)

# FUNCTIONS EXAMPLES

HttpTrigger

TimerTrigger

**CosmosDBTrigger**

BlobTrigger

# COSMOSDBTRIGGER

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-cosmosdb-v2-trigger?tabs=java>

Allows to execute a function when there is an insert or update in a collection (but not for deletes).

Trigger properties:

- **databaseName, collectionName** – name of database and collection
- **connectionStringSetting** – application property where connection string is defined
  - NOTE: the script created by AzureManagement sets a property with name [AzureCosmosDBConnection](#)

# COSMOSDBTRIGGER (EXAMPLE)

```
@FunctionName("cosmosDBtest")
public void updateMostRecentUsers(@CosmosDBTrigger(name = "cosmosTest",
    databaseName = "scc2122dbnmp",
    collectionName = "users",
    createLeaseCollectionIfNotExists = true,
    connectionStringSetting = "AzureCosmosDBConnection")

    String[] users,
    final ExecutionContext context ) {
    try (Jedis jedis = RedisCache.getCachePool().getResource()) {
        for( String u : users) {
            jedis.lpush("serverless::cosmos::users", u);
        }
        jedis.ltrim("serverless::cosmos::users", 0, 9);
    }
}
```

# COSMOSDBTRIGGER (EXAMPLE)

```
@FunctionName("cosmosDBtest")  
  
public void updateMostRecentUsers(@CosmosDBTrigger(name = "cosmosTest",  
    databaseName = "scc2122dbnmp",  
    collectionName = "users",  
    createLeaseCollectionIfNotExists = true,  
    connectionStringSetting = "AzureCosmosDBConnection")
```

```
String[] users,
```

```
final ExecutionContext context ) {
```

```
try (Jedis jedis = RedisCache.getCachePool().getResource()) {
```

```
    for( String u : users) {
```

```
        jedis.lpush("serverless::cosmos::users", u);
```

```
    }
```

```
    jedis.ltrim("serverless::cosmos::users", 0, 9);
```

```
}
```

```
}
```

Variable with the  
values  
inserted/updated.

Check the list of users on:

[https://YOUR\\_SERVER.azurewebsites.net/api/serverless/redis/lrange/serverless::cosmos::users](https://YOUR_SERVER.azurewebsites.net/api/serverless/redis/lrange/serverless::cosmos::users)



# COSMOSDBTRIGGER CONFIG

Run the following command to set the AzureCosmosDBConnection property

**[note: this command is in the .sh file created by AzureManagement]**

```
az functionapp config appsettings set \  
  --name functions name \  
  --resource-group functions group \  
  --settings  
  "AzureCosmosDBConnection=AccountEndpoint=<url>;AccountKey=  
<key>;"
```

# FUNCTIONS EXAMPLES

TimerTrigger

HttpTrigger

CosmosDBTrigger

**BlobTrigger**

# BLOBTRIGGER

<https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-storage-blob-trigger?tabs=java>

Allows to execute a function when there is an insert or update in collection (but not for deletes).

Trigger properties:

- **path** – includes container name
- **connection** – application property where connection string is defined
  - NOTE: the script created by AzureManagement sets a property with name **BlobStoreConnection**

# BLOBTRIGGER (EXAMPLE)

```
@FunctionName("blobtest")  
  
public void setLastBlobInfo(@BlobTrigger(name = "blobtest",  
                                     dataType = "binary",  
                                     path = "images/{name}",  
                                     connection = "BlobStoreConnection")  
                             byte[] content,  
                             @BindingName("name") String blobname,  
                             final ExecutionContext context) {  
    try (Jedis jedis = RedisCache.getCachePool().getResource()) {  
        jedis.set("serverless::blob::name", "Blob name : " +  
                blobname + " ; size = " +  
                (content == null ? "0" : content.length));  
    }  
}
```

# BLOBTRIGGER (EXAMPLE)

```
@FunctionName("blobtest")  
  
public void setLastBlobInfo(@BlobTrigger(name = "blobtest",  
                                dataType = "binary",  
                                path = "images/{name}",  
                                connection = "BlobStoreConnection")  
    byte[] content,  
    @BindingName("name") String blobname,  
    final ExecutionContext context) {  
    try (Jedis jedis = RedisCache.getCachePool().getResource()) {  
        jedis.set("serverless::blob::name", "Blob name : " +  
            blobname + " ; size = " +  
            (content == null ? "0" : content.length));  
    }  
}
```

Contents of the  
inserted/updated  
BLOB

Check the info of the last BLOB on:  
[https://YOUR\\_SERVER.azurewebsites.net/api/serverless/redis/serverless::blob::name](https://YOUR_SERVER.azurewebsites.net/api/serverless/redis/serverless::blob::name)

# BLOBTRIGGER CONFIG

Run the following command to set the BlobStoreConnection property.

**[note: this command is in the .sh file created by AzureManagement]**

```
az functionapp config appsettings set \  
  --name functions name \  
  --resource-group functions group \  
  --settings "BlobStoreConnection=<connection string>"
```

# DEBUG

In the Azure portal, it is possible to access the exceptions launched by an Azure function.

Microsoft Azure Search resources, services, and docs (G+)

Home > [scc2122funwesteuopenmp](#)

**scc2122funwesteuopenmp** | Functions ...

Function App

Search (Cmd+/) << + Create Refresh Delete

**Warning:** Your app is currently in read only mode because you are running from a package file. To make any changes update the content in your zip file and WEBSITE\_RUN\_FROM\_PACKAGE app setting.

Filter by name...

<input type="checkbox"/> Name ↑↓	Trigger ↑↓	Status ↑↓	
<input type="checkbox"/> <a href="#">echo</a>	HTTP	Enabled	...
<input type="checkbox"/> <a href="#">echo-simple</a>	HTTP	Enabled	...
<input type="checkbox"/> <a href="#">get-redis</a>	HTTP	Enabled	...
<input type="checkbox"/> <a href="#">http-stats</a>	HTTP	Enabled	...
<input type="checkbox"/> <a href="#">periodic-compute</a>	Timer	Enabled	...
<input type="checkbox"/> <a href="#">set-redis</a>	HTTP	Enabled	...

**Functions**

- Functions
- App keys
- App files
- Proxies

**Deployment**

- Deployment slots
- Deployment Center

# DEBUG (2)

In the Azure portal, it is possible to access the exceptions launched by an Azure function.

The screenshot shows the Azure portal interface for monitoring an Azure function named 'http-stats'. The left sidebar contains navigation options: Overview, Developer (Code + Test, Integration, Monitor, Function Keys). The main content area shows the 'Invocations' tab with a summary of 0 success and 3 error counts over the last 30 days. Below this, the 'Invocation Traces' section displays a table of the most recent function invocation traces.

**Invocations Summary:**

Success Count	Error Count
0	3
Last 30 Days	Last 30 Days

**Invocation Traces:**

The twenty most recent function invocation traces. For more advanced analysis, run the query in Application Insights.

Run query in Application Insights Refresh

Filter invocations

Date (UTC)	Success	Result Code	Duration (ms)	Operation Id
2021-11-07 00:00:10.715	✗ Error	500	97	a3d7b205b0cf5140aa6c216a843c4472
2021-11-07 00:00:08.159	✗ Error	500	57	0ac458bd6bc41c41abfaba2a493a8cb0
2021-11-06 23:59:47.753	✗ Error	500	314	e8be768d89137042938bc7d81f9d1513



# DEBUG (3)

In the Azure portal, it is possible to access the exceptions launched by an Azure function.

Microsoft Azure Search resources, services, and docs (G+ /)

Home > scc2122funwesteuopenmp > http-stats

**http-stats | Monitor** Function

Search (Cmd+/) «

Overview

Developer

Code + Test

Integration

**Monitor**

Function Keys

### Invocation Details

Run query in Application Insights

Timestamp	Message	Type
2021-11-07 00:00:10.716	Executing 'Functions.http-stats' (Reason='This function was programmatically called via the host APIs.', Id=241e756c-7fdd-4079-ba26-4ef4d8d1b829)	Information
2021-11-07 00:00:10.785	Result: Failure Exception: NullPointerException: Stack: java.lang.reflect.InvocationTargetException at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(Unknown Source) at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source) at java.base/java.lang.reflect.Method.invoke(Unknown Source) at com.microsoft.azure.functions.worker.broker.JavaMethodInvokeInfo.invoke(JavaMethodInvokeInfo.java:22) at com.microsoft.azure.functions.worker.broker.EnhancedJavaMethodExecutorImpl.execute(EnhancedJavaMethodExecutorImpl.java:55) at com.microsoft.azure.functions.worker.broker.JavaFunctionBroker.invokeMethod(JavaFunctionBroker.java:57) at com.microsoft.azure.functions.worker.handler.InvocationRequestHandler.execute(InvocationRequestHandler.java:33) at com.microsoft.azure.functions.worker.handler.InvocationRequestHandler.execute(InvocationRequestHandler.java:10) at com.microsoft.azure.functions.worker.handler.MessageHandler.handle(MessageHandler.java:45) at com.microsoft.azure.functions.worker.JavaWorkerClient\$StreamingMessagePeer.lambda\$onNext\$0(JavaWorkerClient.java:92) at java.base/java.util.concurrent.Executors\$RunnableAdapter.call(Unknown Source) at java.base/java.util.concurrent.FutureTask.run(Unknown Source) at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source) at java.base/java.util.concurrent.ThreadPoolExecutor\$Worker.run(Unknown Source) at java.base/java.lang.Thread.run(Unknown Source) Caused by: java.lang.NullPointerException at scc.serverless.HttpFunction.run(HttpFunction.java:34) ... 16 more	Error
2021-11-07 00:00:10.785	Executed 'Functions.http-stats' (Failed, Id=241e756c-7fdd-4079-ba26-4ef4d8d1b829, Duration=68ms)	Error
2021-11-07 00:00:10.795	Result: Failure Exception: NullPointerException: Stack: java.lang.reflect.InvocationTargetException at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) at	Error

# CODE PROVIDED

The code provided (lab5.zip) is a Maven project with example Azure functions that can be deployed to Azure.

**You will need to change the properties in pom.xml – check slide 5.**

For compiling and deploying, just run:

```
mvn compile package azure-functions:deploy
```

After deploying, **do not forget to set the properties used by the Azure functions.**

**NOTE:** Blob and CosmosDB functions assume certain container/db names – update accordingly for your code.

Functions that access Redis assume that the appropriate keys are set.

# DEPLOY FUNCTIONS ON AZURE (CONT'D)

```
(base) LazyMBP:lab5 nmp$ mvn azure-functions:deploy
[INFO] Scanning for projects...
[INFO]
[INFO] -----< pt.unl.fct.di.scc:scc2223-lab5 >-----
[INFO] Building Azure Java Functions 1.0
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- azure-functions-maven-plugin:1.21.0:deploy
[INFO] Auth type: AZURE_CLI
Default subscription: Azure para Estudantes(83abecdf-6b46-43d6-b0dc-b37ba4011955)
Username: nmp@FCT.UNL.PT
[INFO] Subscription: Azure para Estudantes(83abecdf-6b46-43d6-b0dc-b37ba4011955)
[INFO] Reflections took 125 ms to scan 5 urls, producing 25 keys and 751 values
[INFO] Set function worker runtime to java.
[INFO] Starting deployment...
[INFO] Trying to deploy artifact to scc2223funwesteuropenmp...
[INFO] Successfully deployed the artifact to https://scc2223funwesteuropenmp.azurewebsites.net
[INFO] Deployment done, you may access your resource through scc2223funwesteuropenmp.azurewebsites.net
[INFO] Syncing triggers and fetching function information
[INFO] Querying triggers...
[INFO] HTTP Trigger Urls:
[INFO]   echo : https://scc2223funwesteuropenmp.azurewebsites.net/api/serverless/echo/{text}
[INFO]   echo-simple : https://scc2223funwesteuropenmp.azurewebsites.net/api/serverless/echosimple/{text}
[INFO]   get-redis : https://scc2223funwesteuropenmp.azurewebsites.net/api/serverless/redis/{key}
[INFO]   http-info : https://scc2223funwesteuropenmp.azurewebsites.net/api/serverless/info
[INFO]   http-stats : https://scc2223funwesteuropenmp.azurewebsites.net/api/serverless/stats
[INFO]   lrange-redis : https://scc2223funwesteuropenmp.azurewebsites.net/api/serverless/redis/lrange/{key}
[INFO]   set-redis : https://scc2223funwesteuropenmp.azurewebsites.net/api/serverless/redis/{key}
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:05 min
[INFO] Finished at: 2022-10-17T16:29:57+01:00
[INFO] -----
```

Sometimes deployment fails... when it succeeds you should have something like this output, with the URLs to access functions triggered by HTTP.

# TODO

Use Azure function for implementing features in your project.

Suggestions:

- Use timer functions for closing auctions and clean-up tasks, periodically.
- Use CosmosDB function for maintain list of most recent auctions.