

Hands-on lab

# *Implementing Caching*

Contents

Contents

[*Implementing Caching* 1](#_Toc400974266)

[Overview 2](#_Toc400974267)

[System requirements 2](#_Toc400974268)

[Exercise 1: Create the Redis Cache 3](#_Toc400974269)

[Task 1 – Create the Redis Cache 3](#_Toc400974270)

[Exercise 2: Configure and Instrument the Application 4](#_Toc400974271)

[Task 1 – Install the Assemblies and Configure the Connection 4](#_Toc400974272)

[Task 2 – Initialize the Connection 5](#_Toc400974273)

[Task 3 – Instrument the Application using the Redis Cache 6](#_Toc400974274)

[Task 4 – Check in the changes and deploy to production. 7](#_Toc400974275)

# Overview

In this lab, you will create a Redis cache instance and instrument your application to use it.

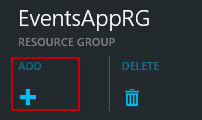
## System requirements

* 1. You must have the following to complete this lab:
  + A reliable Internet connection
  + An active Microsoft Azure subscription
  1. Estimated time to complete this demo:  **20 Minutes**

# Exercise 1: Create the Redis Cache

## Task 1 – Create the Redis Cache

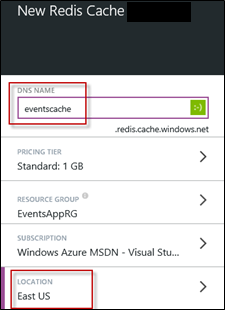
* 1. Within the Azure portal (<https://portal.azure.com>) open the EventsAppRG resource group. Then click the **ADD** button.



* 1. Select Redis Cache. Then click **CREATE** on the next screen.



* 1. Specify a unique name for the cache. Change the location to match the location of your website and database.



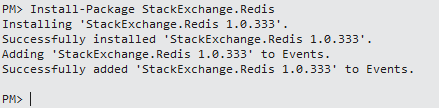
* 1. Click **CREATE**.

# Exercise 2: Configure and Instrument the Application

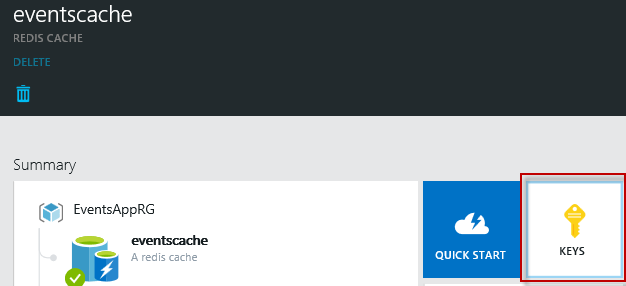
## Task 1 – Install the Assemblies and Configure the Connection

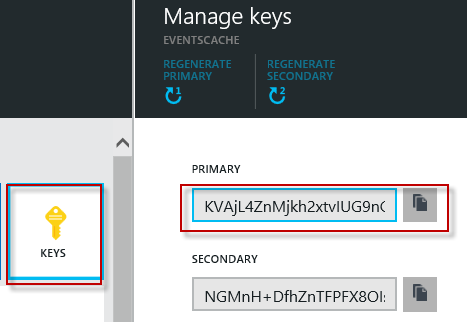
* 1. Within the Events project in Visual Studio open the Package Manager Console and execute the following command. Ensure the default project is set to **Events**.

Install-Package StackExchange.Redis



* 1. Open the Redis cache in the Azure portal to retrieve the authentication key.





* 1. Construct a connection string using the cache name and the primary key in the following format. Replace the placeholder values with the values for your cache.

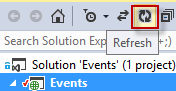
[cache name].redis.cache.windows.net,ssl=true,password=[primary key]

* 1. Create a new connection string in the web.config of the Events application.

Key: RedisConnection and specify the connection string for the value.



* 1. Add the Redis .NET extensions to your project by copying the folder from C:\WebsitesTraining\Samples\RedisExtentions to C:\WebsitesTraining\Events\Events.
  2. Within Visual Studio, click the refresh button to show the new folder.



* 1. Right click on the RedisExtensions folder and click Include in Project.

## Task 2 – Initialize the Connection

* 1. Using Visual Studio add the following namespaces to the global.asax.

using System.Configuration;  
using StackExchange.Redis;

* 1. Add the following static property to the MVCApplication class in the global.asax.cs file.

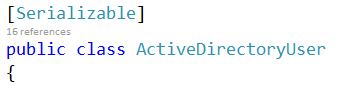
private static ConnectionMultiplexer redisCache;  
 public static ConnectionMultiplexer RedisCache  
 {  
 get  
 {  
 if (redisCache == null || !redisCache.IsConnected)  
 {  
 redisCache = ConnectionMultiplexer.Connect(ConfigurationManager.ConnectionStrings["RedisConnection"].ConnectionString);  
 }  
 return redisCache;  
 }  
 }

## Task 3 – Instrument the Application using the Redis Cache

* 1. Open the file Models\ActiveDirectoryUser.cs
  2. Add the system namespace to the file.

using System;

* 1. Add the [Serializable] attribute to the class.



* 1. Save the file.
  2. Open the \Repository\EventsRepository.cs file.
  3. Add the namespaces for Redis to the top of the page under the existing namespaces.

using StackExchange.Redis;  
using Events.RedisExtensions;

* 1. Update the code to use the Redis cache.

**Find the line:**

if (HttpContext.Current.Cache["ADUSERS"] == null)

**Replace it with the following:**

IDatabase cache = MvcApplication.RedisCache.GetDatabase();  
 var adUsers = cache.Get<List<ActiveDirectoryUser>>("ADUSERS");  
 if (adUsers == null)

**Find the line:**

HttpContext.Current.Cache.Insert("ADUSERS", activeDirectoryUsers, null, DateTime.Now.AddMinutes(5), TimeSpan.Zero, System.Web.Caching.CacheItemPriority.Normal, null);

**Replace it with the following:**

cache.Set("ADUSERS", activeDirectoryUsers, TimeSpan.FromMinutes(5));

**Find the Line:**

activeDirectoryUsers = (List<ActiveDirectoryUser>)HttpContext.Current.Cache["ADUSERS"];

**Replace it with the following:**

activeDirectoryUsers = (List<ActiveDirectoryUser>)cache.Get("ADUSERS");

* 1. Using Visual Studio, open the Controllers\AdImagesController.cs file.

Add the following namespaces:

using StackExchange.Redis;

using Events.RedisExtensions;

* 1. Update the code to use the Redis cache.

**Find the line:**

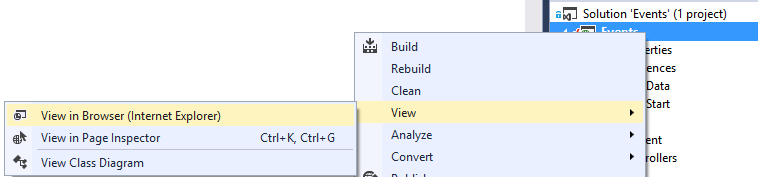
if (HttpContext.Cache["ADUSERS"] != null)

**Replace it with the following:**

IDatabase cache = MvcApplication.RedisCache.GetDatabase();  
 var users = cache.Get<List<ActiveDirectoryUser>>("ADUSERS");  
 if (users != null)

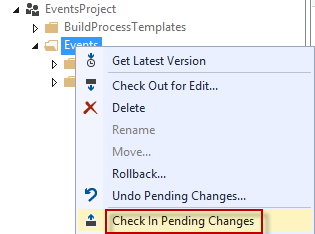
**Remove the line:**List<ActiveDirectoryUser> users = (List<ActiveDirectoryUser>)HttpContext.Cache["ADUSERS"];

* 1. Rebuild the application and test locally by viewing in the browser.

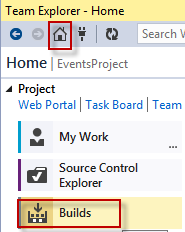


## Task 4 – Check in the changes and deploy to production.

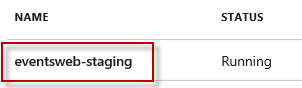
* 1. Within Visual Studio, Open Team Explorer and click on Source Control Explorer.
  2. Right click on Events and click on **Check In Pending Changes.**



* 1. Add a check in comment "Added Redis Cache Support" and click **Check In**.
  2. Monitor the build status by clicking the home button in **Team Explorer**, and click on **Builds**.



* 1. Open the staging slot for the site in the Azure management portal.



* 1. Click **BROWSE** and validate the site functions correctly.
  2. Click the Swap button on the top menu. When prompted click on the staging deployment slot to update the production slot with the changes automatically deployed to the staging slot.



* 1. Open the production slot and browse to ensure the site is functional.



Summary

In this lab, you created a Redis cache instance and instrumented your application to use it.