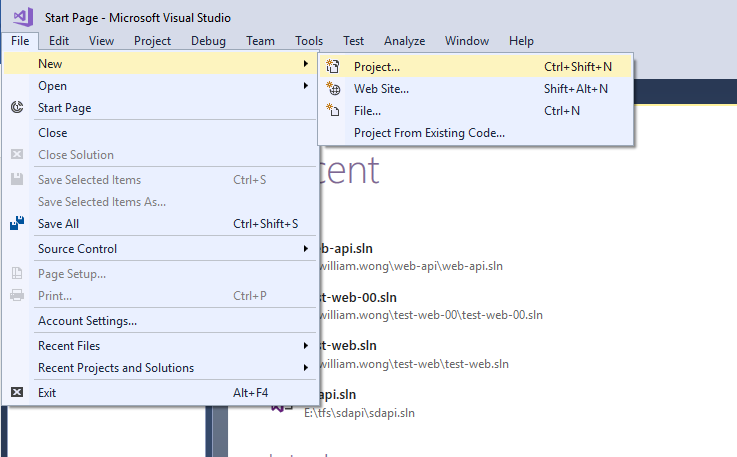
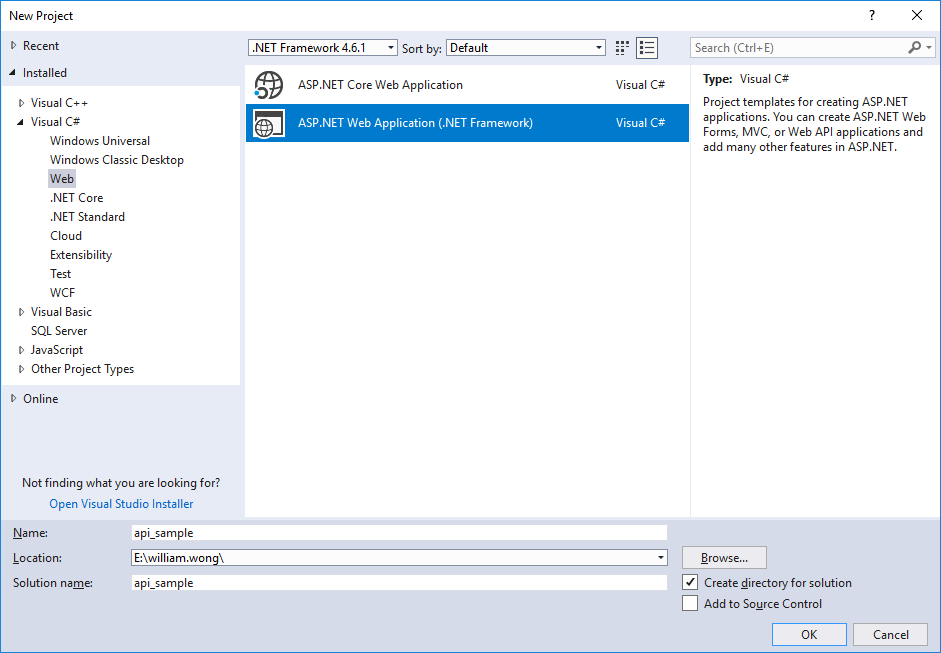
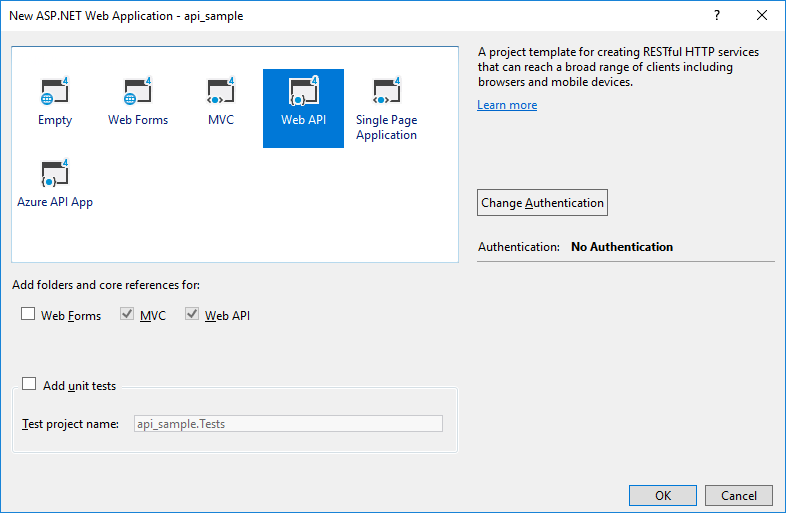
打開vs，新增1個專案



選擇c#內的web，再選擇asp .net web application，專案取名為api\_sample

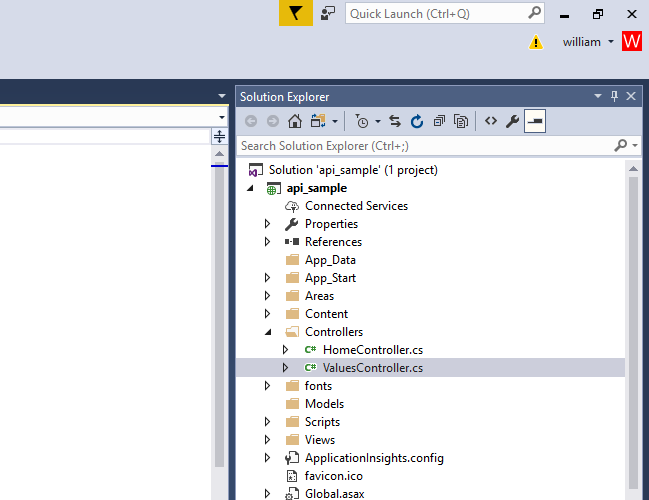


選擇web api，然後在reference部份同時勾選mvc和web api

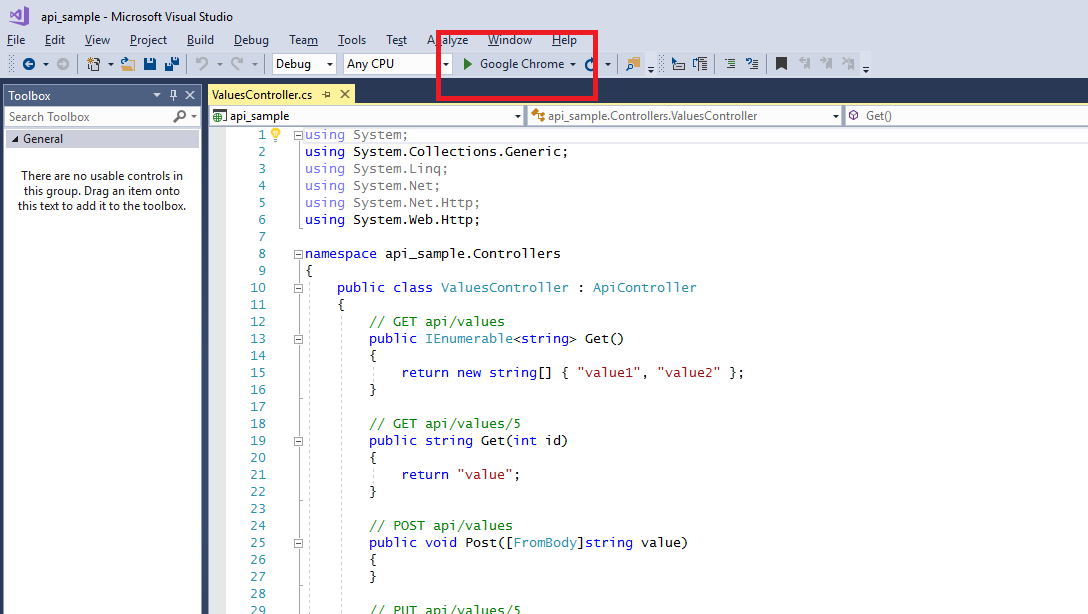


\*這個階段我們先修改一下原有的範例作教學

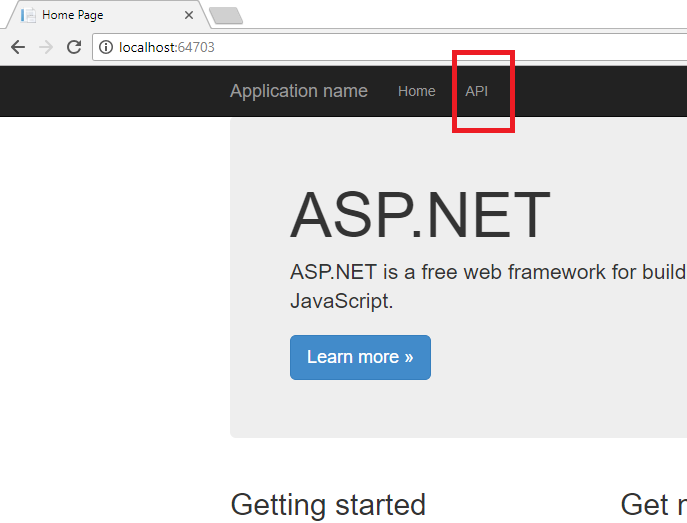
在../solution/controllers/打開ValuesController.cs



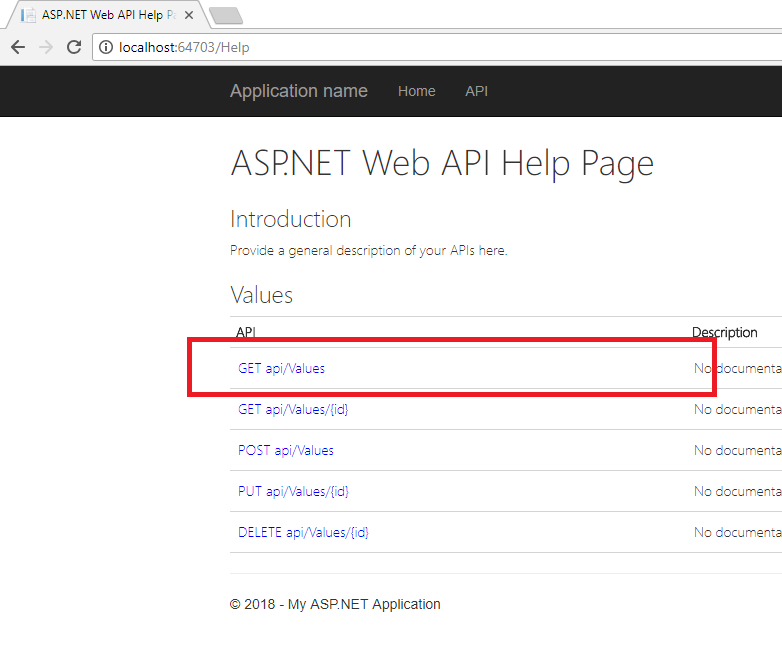
可以看到範本的api內容，執行Google chrome debug測試

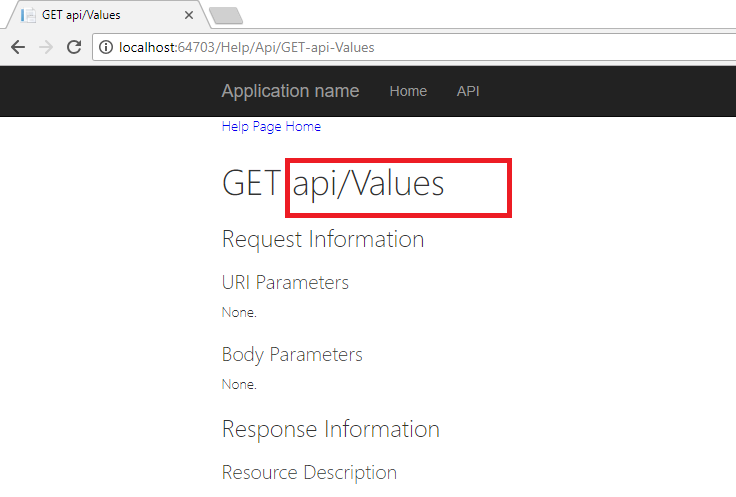


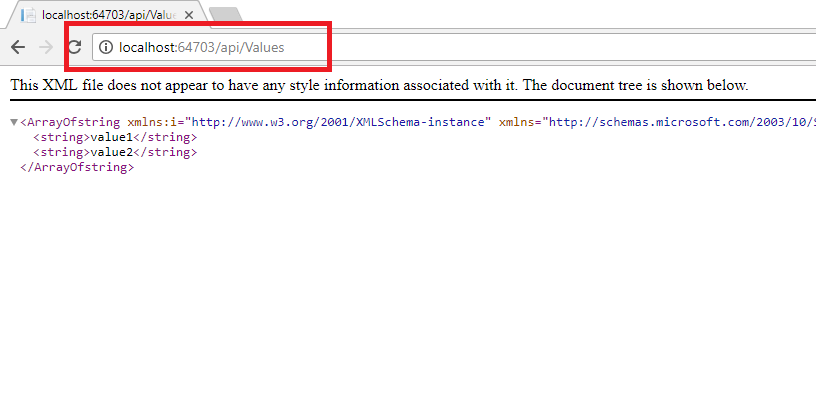
這是範例的網站，點擊api可以看到web server所提供的api list



這是某一個controller所支援/提供的api，點選連結看api的達細內容

這是一個get method，而他的網址是 …/api/Value

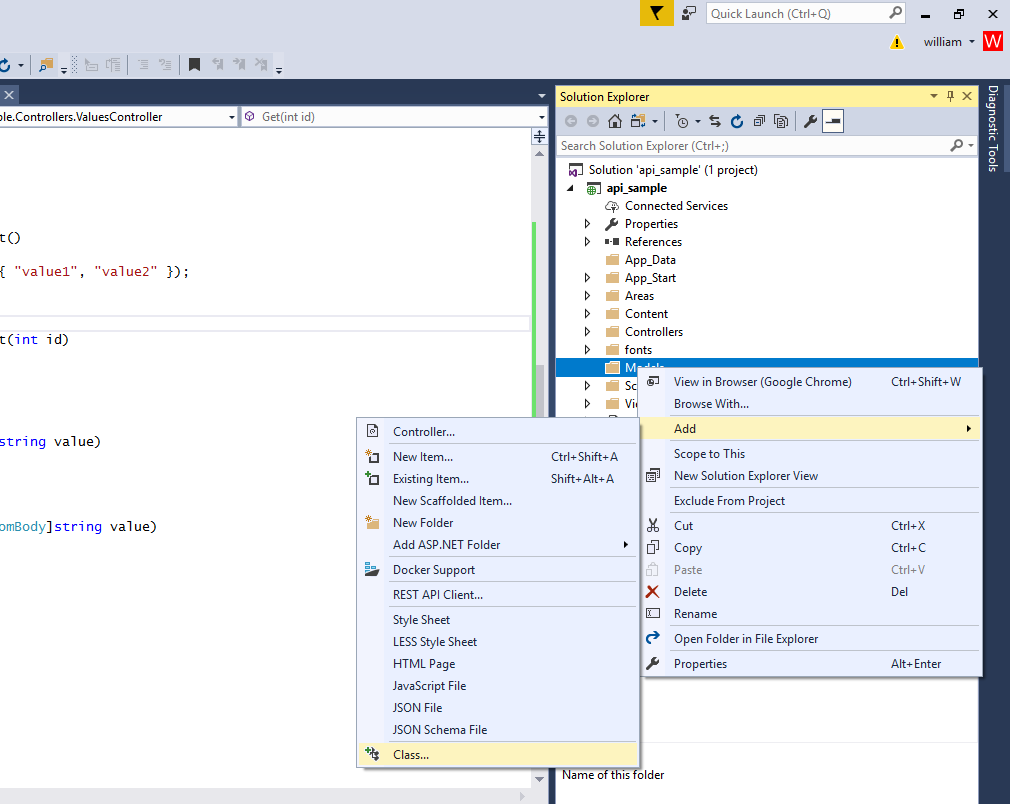


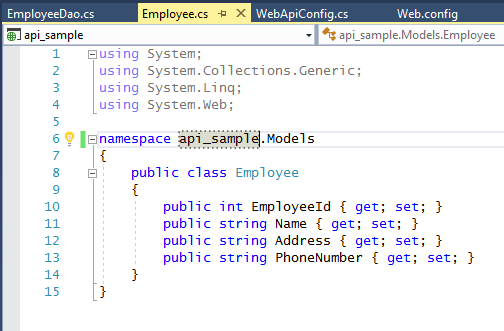
把網址換成<http://localhost:64703/api/Values>，可以看到成功呼叫api的例子

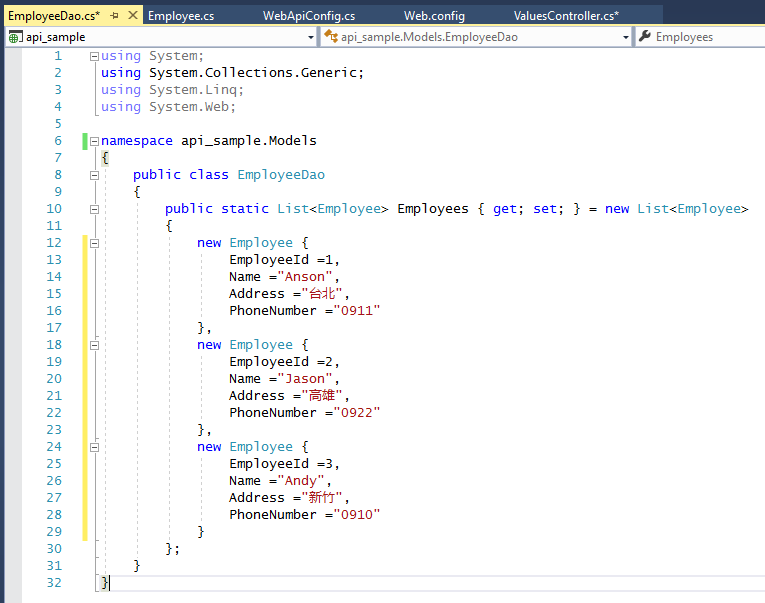
把範本的api內容換成 (這部份暫不作解釋)

|  |
| --- |
| // GET api/values  public IHttpActionResult Get()  {  return Ok(new string[] { "value1", "value2" });  }  // GET api/values/5  public IHttpActionResult Get(int id)  {  return Ok("value");  }  // POST api/values  public void Post([FromBody]string value)  {  }  // PUT api/values/5  public void Put(int id, [FromBody]string value)  {  }  // DELETE api/values/5  public void Delete(int id)  {  } |

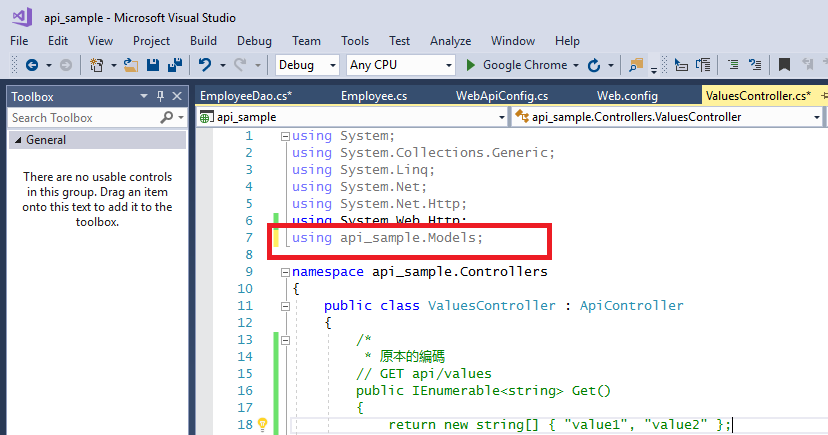
先作1個Employee例子，先在Models內創建Employee.cs及EmployeeDao.cs，後者為資料庫的功用







修改ValuesController.cs，把Models創建的內容都引入到Controller里



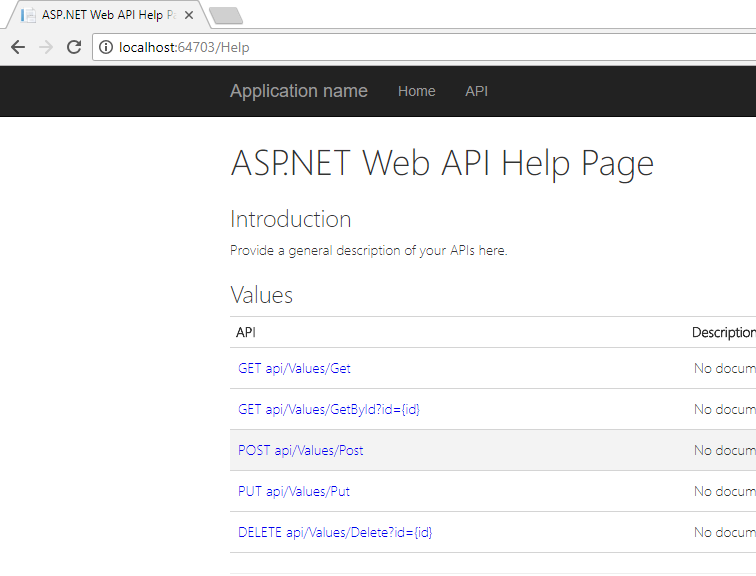
再次把把範本的api內容換成，現在呼叫api即可獲得employee的內容

|  |
| --- |
| …  namespace api\_sample.Controllers  {  public class ValuesController : ApiController  {    public IHttpActionResult Get()  {  return Ok(EmployeeDao.Employees);  }  // GET api/values/5  public IHttpActionResult Get(int id)  {  return Ok(EmployeeDao.Employees.FirstOrDefault(x => x.EmployeeId == id));  }  // POST api/values  public IHttpActionResult Post(Employee employee )  {  EmployeeDao.Employees.Add(employee);  return Ok(EmployeeDao.Employees);  }  // PUT api/values/5  public IHttpActionResult Put(Employee employee)  {  var result=EmployeeDao.Employees.FirstOrDefault(x => x.EmployeeId == employee.EmployeeId);  if (result != null)  {  result.Name = employee.Name;  result.PhoneNumber = employee.PhoneNumber;  result.Address = employee.Address;  }  return Ok(EmployeeDao.Employees);  }  // DELETE api/values/5  public IHttpActionResult Delete(int id)  {  EmployeeDao.Employees.RemoveAll(x=>x.EmployeeId==id);  return Ok(EmployeeDao.Employees);  }  }  } |

然後使用router attribute來快速定義路由，加上以下內容

|  |
| --- |
| [RoutePrefix("api/Values")]  public class ValuesController : ApiController  {  [Route("Get")]  [HttpGet]  public IHttpActionResult Get()  {  return Ok(EmployeeDao.Employees);  }  [Route("GetById")]  [HttpGet]  public IHttpActionResult GetById(int id)  {  return Ok(EmployeeDao.Employees.FirstOrDefault(x => x.EmployeeId == id));  }  [Route("Post")]  [HttpPost]  public IHttpActionResult Post(Employee employee)  {  EmployeeDao.Employees.Add(employee);  return Ok(EmployeeDao.Employees);  }  [Route("Put")]  [HttpPut]  public IHttpActionResult Put(Employee employee)  {  var result = EmployeeDao.Employees.FirstOrDefault(x => x.EmployeeId == employee.EmployeeId);  if (result != null)  {  result.Name = employee.Name;  result.PhoneNumber = employee.PhoneNumber;  result.Address = employee.Address;  }  return Ok(EmployeeDao.Employees);  }  [Route("Delete")]  [HttpDelete]  public IHttpActionResult Delete(int id)  {  EmployeeDao.Employees.RemoveAll(x => x.EmployeeId == id);  return Ok(EmployeeDao.Employees);  }  } |

在這可以看到部份需要參數的function呼叫方式是 ?id={id}，參數會以string方式傳遞且能夠在url看到

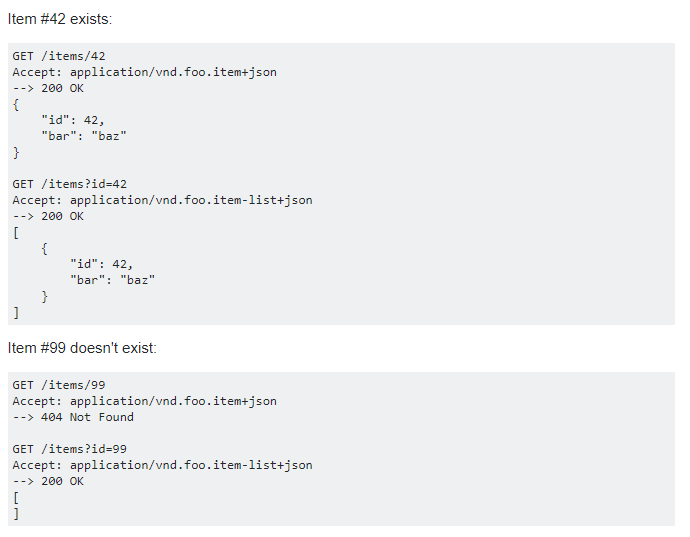


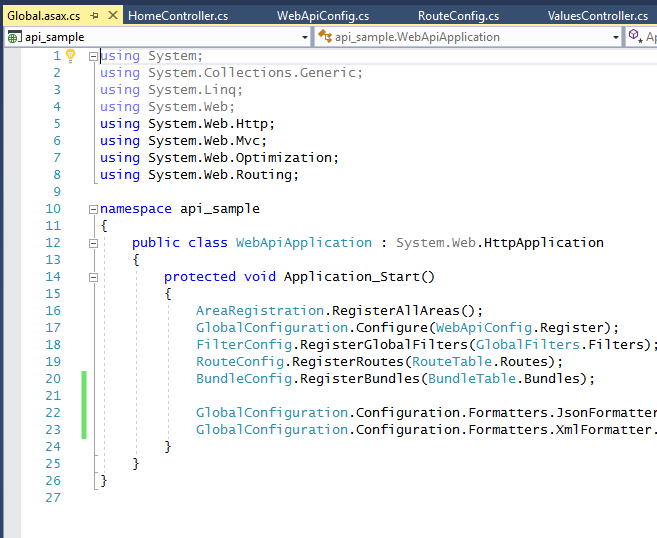
\*在這裡說一下path parameters和request parameters分別：

如需要用path parameters，我們需要修改一下router attribute，以GetById為例，然後在url的呼叫方式將變為…/{id}

|  |
| --- |
| [Route("GetById/{id}")]  [HttpGet]  public IHttpActionResult GetById(int id)  {  return Ok(EmployeeDao.Employees.FirstOrDefault(x => x.EmployeeId == id));  } |

使用path參數時，目的是用上key value找1個不會重覆的內容，他是以單一object形式回傳，而沒有相符結果的情況下直接返回404錯誤；使用request參數時，目的大多為尋找相關記錄，他會將所有符合條件的內容以list-object形式回傳，而沒有相符結果的情況下只會回傳一個空白的list

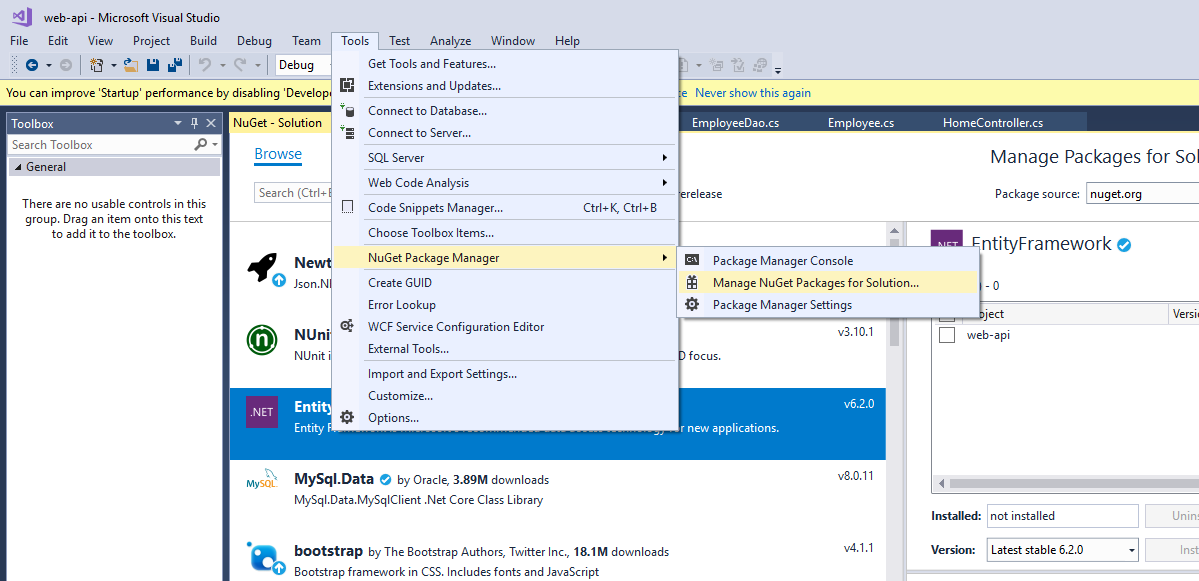


修改solution底下的Global.asax，加上以下指令，使回傳內容由xml變為json格式

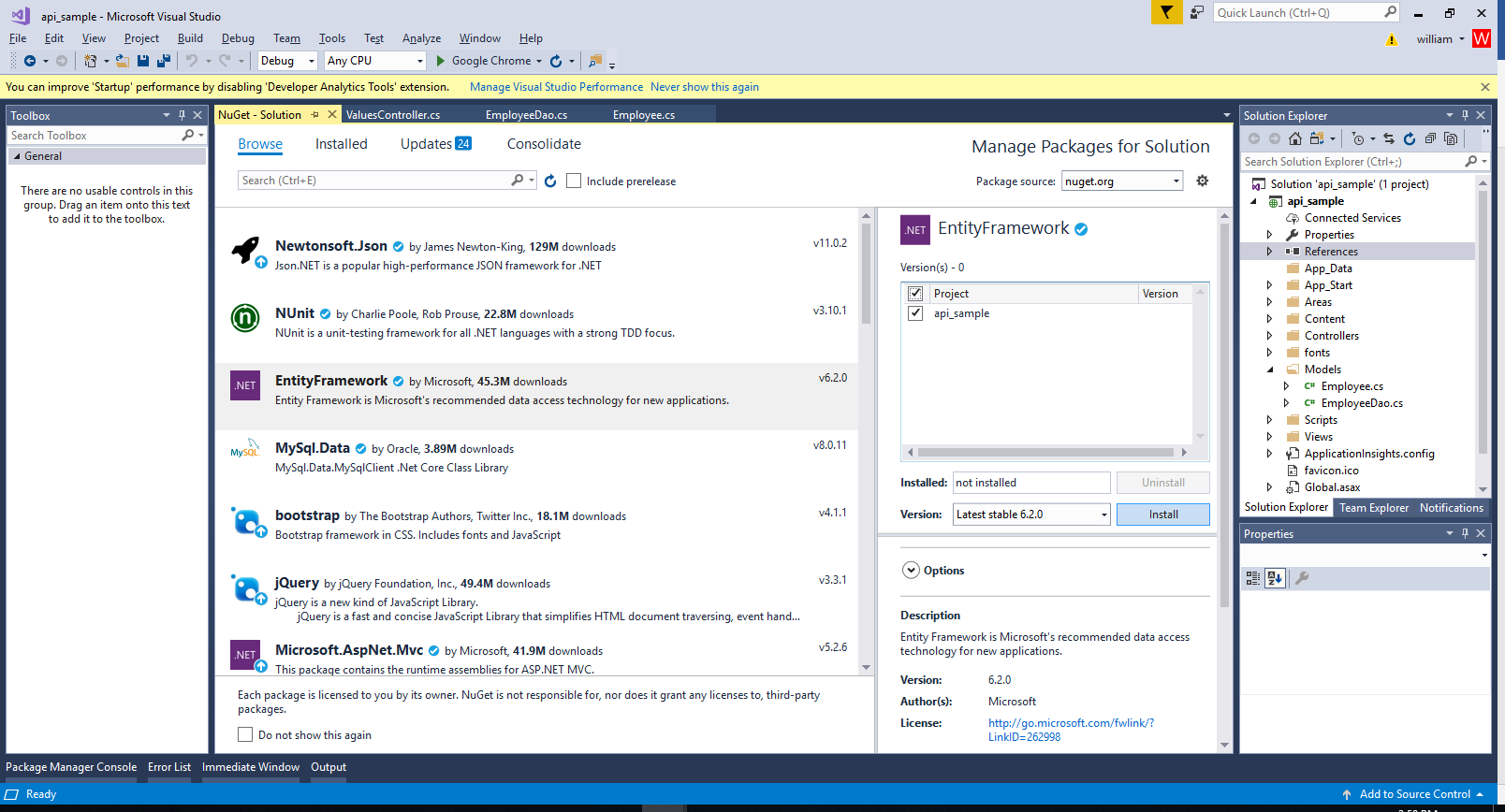
|  |
| --- |
| GlobalConfiguration.Configuration.Formatters.JsonFormatter.SerializerSettings.ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.Ignore;  GlobalConfiguration.Configuration.Formatters.XmlFormatter.SupportedMediaTypes.Clear(); |

**連結資料庫**

加入Entity framework，先到Tools>NuGet Package Manager>Manage package for solution



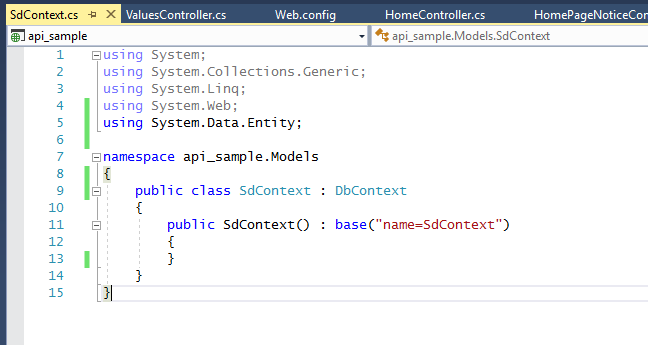
搜尋Entity framework，安裝最新版本



為資料庫加入一個Pre-Set的connectionStrings，打開Web.config，在<configuration>...</configuration>內加入以下內容，attr name及connectionString的值因應需要修改

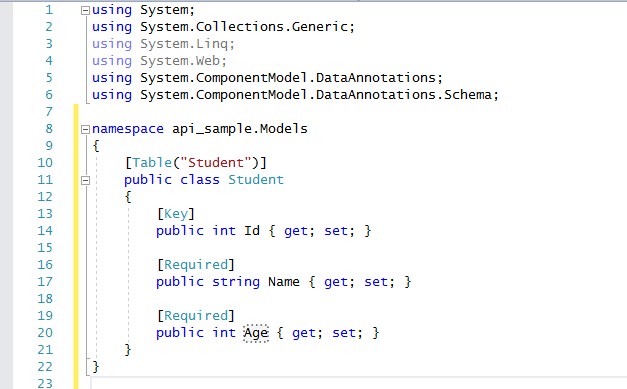
|  |
| --- |
| <connectionStrings>  <add name="SdContext"  connectionString="data source=192.168.1.221,1433;initial catalog=SdSystem\_dev;user id=sa;password=sd123456;multipleactiveresultsets=True;application name=EntityFramework"  providerName="System.Data.SqlClient" />  </connectionStrings> |

在Model下新增1個class(名稱沒有規範，暫稱SdContext)，用於連接db，加入using System.Data.Entity



假設在資料庫內有1個student表格，分別儲存id/name/age

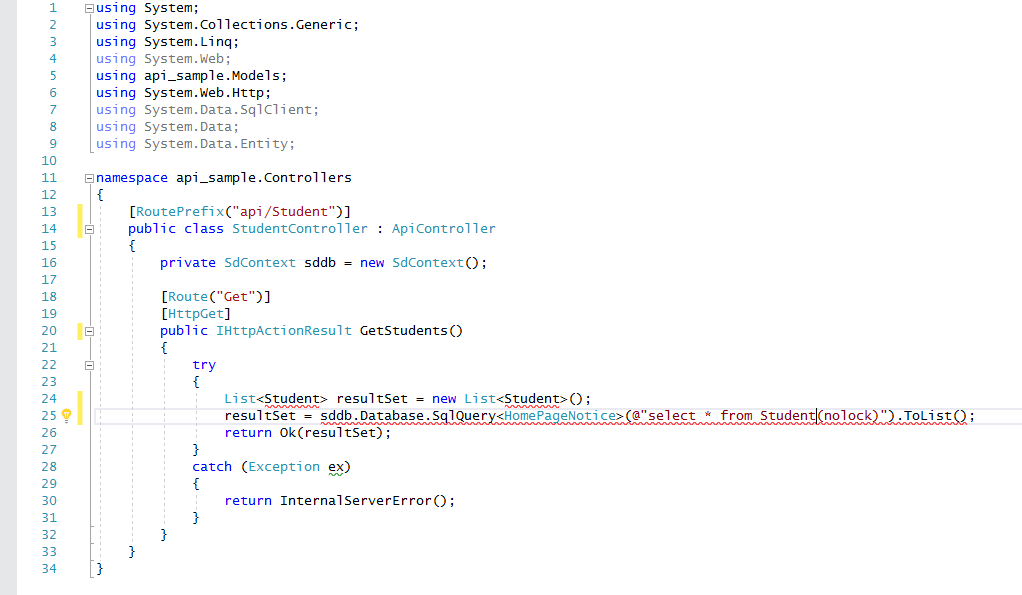
在Model下新增1個class “Student”，定義相對應的屬性id/name/age



此時需要回到SdContext，加入以下內容在class SdContext之內

|  |
| --- |
| public virtual DbSet<Student> Student { get; set; } |

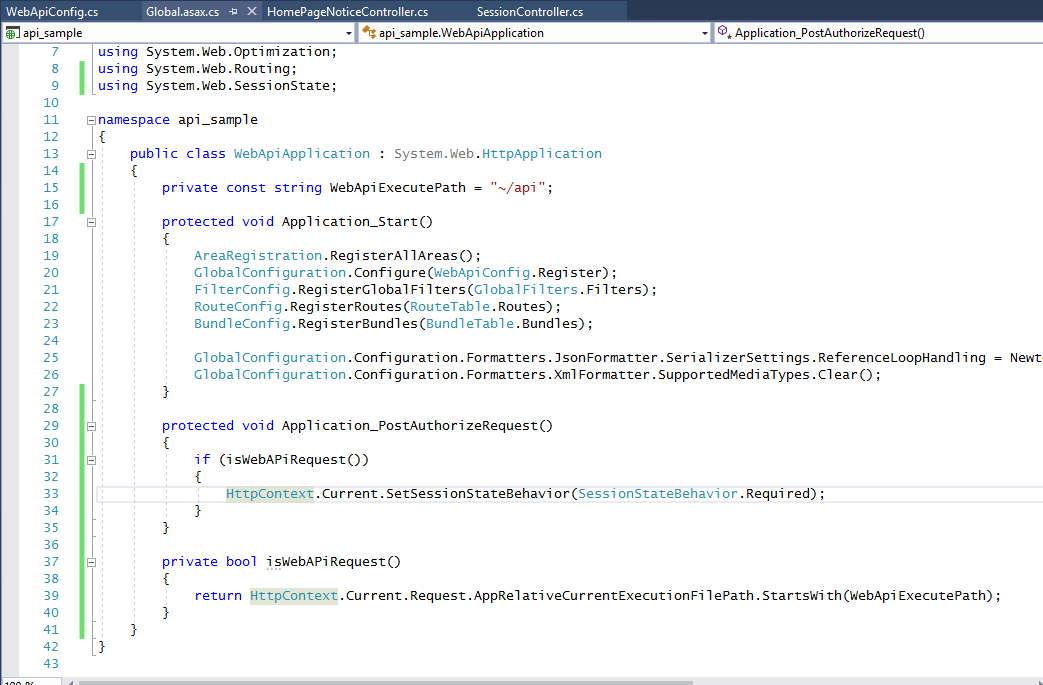
在Controllers下新增StudentController用來存取資料，只在1個簡單的GET



**啟用http session**

web api默認沒有啟動session部份，session內容是null

第一步是需要修改Global.asax



完成後內容如下，藍色部份為新增的指令，功能是在滿足指定路俓的情況下才執行session

|  |
| --- |
| …  using System.Web.Routing;  using System.Web.SessionState;  namespace api\_sample  {  public class WebApiApplication : System.Web.HttpApplication  {  private const string WebApiExecutePath = "~/api";  protected void Application\_Start()  {  AreaRegistration.RegisterAllAreas();  GlobalConfiguration.Configure(WebApiConfig.Register);  FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);  RouteConfig.RegisterRoutes(RouteTable.Routes);  BundleConfig.RegisterBundles(BundleTable.Bundles);  GlobalConfiguration.Configuration.Formatters.JsonFormatter.SerializerSettings.ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.Ignore;  GlobalConfiguration.Configuration.Formatters.XmlFormatter.SupportedMediaTypes.Clear();  }  protected void Application\_PostAuthorizeRequest()  {  if (isWebAPiRequest())  {  HttpContext.Current.SetSessionStateBehavior(SessionStateBehavior.Required);  }  }  private bool isWebAPiRequest()  {  return HttpContext.Current.Request.AppRelativeCurrentExecutionFilePath.StartsWith(WebApiExecutePath);  }  }  } |

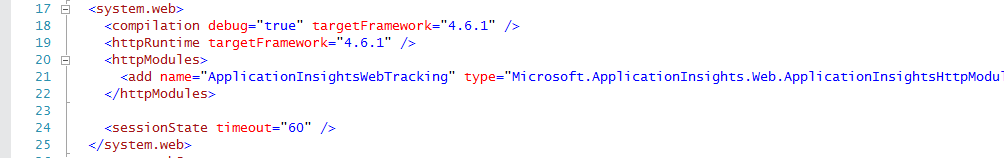
protected void Application\_PostAuthorizeRequest(): 每次執行request時觸發

HttpContext.Current.SetSessionStateBehavior(): 設定session狀態

SessionStateBehavior.Required: 此為Full read-write session

HttpContext.Current.Request.AppRelativeCurrentExecutionFilePath.StartsWith(): 檢查url path是否滿足指定路俓

修改session timeout時間，在Web.config內修改<system.web>…..</system.web>部份，加入指令<sessionState timeout="60" />



『實現session讀取及寫入』

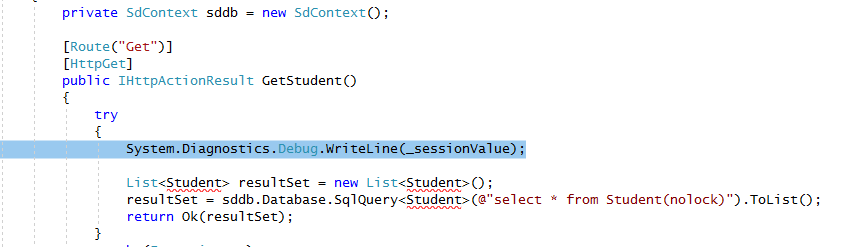
先在Student class下新建1個api(由於是舉例說明之用，邏輯與標準的api有別)，執行此api後session[“index01”]內容變為hello session

|  |
| --- |
| [Route("GetSession")]  [HttpGet]  public IHttpActionResult GetHomePageNotices\_Session()  {  if (HttpContext.Current.Session != null)  {  HttpContext.Current.Session["index01"] = "hello session";  }  else  {  System.Diagnostics.Debug.WriteLine("Session null");  }  return Ok();  } |

然後新建1個controller “SessionController.cs”，然後他作為其他api controller的父類別，使所有他的子類別都可以存取到同一個session內容

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Web;  using System.Web.Http;  namespace api\_sample.Controllers  {  public class SessionController : ApiController  {  protected string \_sessionValue = "";  public SessionController()  {  if (HttpContext.Current.Session != null)  {  if (HttpContext.Current.Session["index01"] != null)  {  \_sessionValue = HttpContext.Current.Session["index01"].ToString();  }  else  {  System.Diagnostics.Debug.WriteLine("Session index01 null");  }  }  else  {  System.Diagnostics.Debug.WriteLine("Session null");  }  }  }  } |

最後修改一下Student GET的部份，把session內容在執行時print出來



可以分別根據以下動作呼叫api，即可session存取情況

1. api/Student/Get > api/Student/GetSession
2. api/Student/GetSession > api/Student/Get