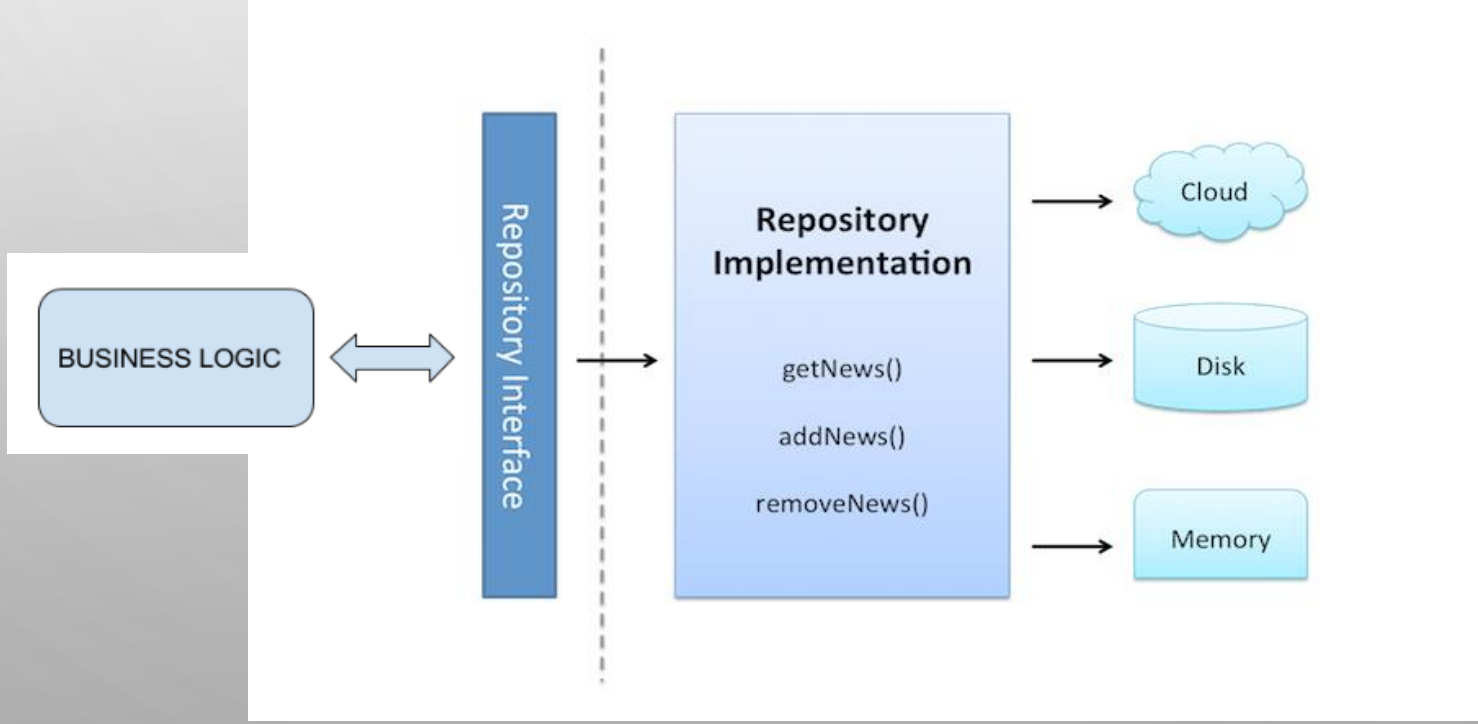
REACT

11 FETCH AND JQUERY AJAX CALLS

REPOSITORY PATTERN

- נרצה לממש את פונקציות ה CRUD בנפרד מהמחלקות של הMODEL.
- נייצר INTERFACE עם פונקציות ה CRUD
- נממש את ה INTERFACE ע"י מחלקות שמחוברות לבסיסי נתונים שונים למשל SQL או IN –MEMORY



INTERFACE

```
3 references
interface IStudentsRepository
{
    3 references | 0 exceptions
    IEnumerable<Student> GetALLStudents();
    5 references | 0 exceptions
    Student GetStudentById(int id);
    3 references | 0 exceptions
    Student AddStudent(Student s);
    3 references | 0 exceptions
    Student UpdateStudent(int id, Student s);
    3 references | 0 exceptions
    bool DeleteStudent(Student s);
}
```

IMPLEMENT INTERFACE

```
1 reference
public class MockStudentsRepository : IStudentsRepository
{
    static public List<Student> students = new List<Student>()
    {
        new Student(){ID=1, Name="avi", Grade=100 },
        new Student(){ID=2, Name="charlie", Grade=90 },
        new Student(){ID=3, Name="benny", Grade=98 },
        new Student(){ID=4, Name="dora", Grade=97 },
    };

    3 references | 0 exceptions
    public Student AddStudent(Student s)
    {
        students.Add(s);
        return s;//in sql will be returned with the newly created id!
```

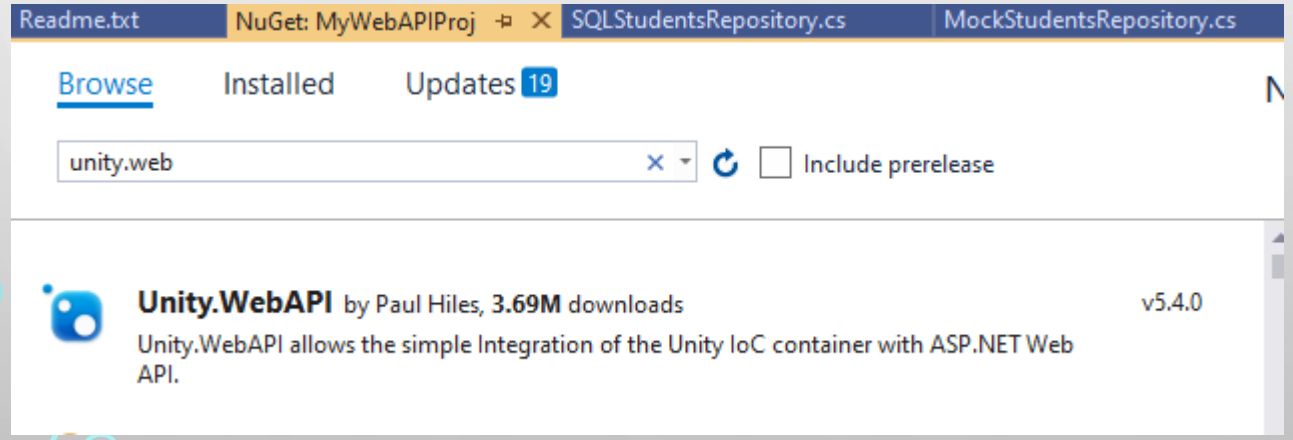
ANOTHER IMPLEMENTATION

```
0 references
public class SQLStudentsRepository : IStudentsRepository
{
    //SQL Connection...

    3 references | 0 exceptions
    public Student AddStudent(Student s)
    {
        //SQL insert...
        return new Student();//in sql will be returned with the newly created id!
    }

    3 references | 0 exceptions
    public bool DeleteStudent(Student s)
    {
        //sql DELETE...
```

CONNECT THE REPOSITORY TO THE CONTROLLER – THE WRONG WAY.

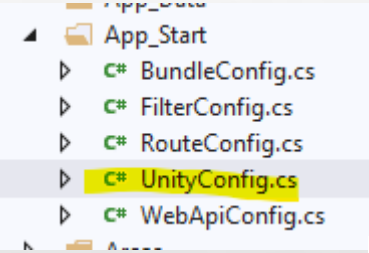
```
0 references
public class StudentsRWController : ApiController
{
    IStudentsRepository repo = new MockStudentsRepository();
    //IRepositoryStudents repo = new SQLRepositoryStudents();

    0 references | 0 requests | 0 exceptions
    public IHttpActionResult Get()
    {
        try
        {
            Student[] sa = repo.GetALLStudents().ToArray();
            //throw new InvalidOperationException("nirs message - invalid op!");
            return Ok(sa);
        }
        catch (Exception ex)
        {
            //return BadRequest(ex.Message); //opt 1
            return Content(HttpStatusCode.BadRequest, ex); //opt 2
        }
    }
```

CONNECT THE REPOSITORY TO THE CONTROLLER – THE WRONG WAY. DEPENDENCY INJECTION – INVERSION OF CONTROL - USING UNITY



- בכדי לעשות הפרדה אמיתית בין הCONTROLLER לבין ה REPOSITORY. דהיינו בין ה BAL לDAL. כך ששינוי נניח בCTOR של ה REPOSITORY לא יצריך שום שינוי ב CONTROLLER נעשה שימוש ב DI.

CONNECT THE REPOSITORY TO THE CONTROLLER



- הפעם נחבר את ה CONTROLLER ל REPOSITORY **לא** מתוך הCONTROLLER. אלא מבחוץ ע"י UnityConfig.

```
1 reference
public static class UnityConfig
{
    1 reference | 0 exceptions
    public static void RegisterComponents()
    {
        var container = new UnityContainer();

        // register all your components with the container here
        // it is NOT necessary to register your controllers

        // e.g. container.RegisterType<ITestService, TestService>();
        container.RegisterType<IStudentsRepository, MockStudentsRepository>();

        GlobalConfiguration.Configuration.DependencyResolver = new UnityDependenc
    }
}
```

DEPENDENCY INJECTION

```
5 references
public interface IStudentsRepository
{
    3 references | 0 exceptions
    IEnumerable<Student> GetALLStudents();
    5 references | 0 exceptions
    Student GetStudentById(int id);
```

```
1 reference
public class StudentsRWController : ApiController
{
    IStudentsRepository repo; //= new MockStudentsRepository();
    //IRepositoryStudents repo = new SQLRepositoryStudents();

    0 references | 0 exceptions
    public StudentsRWController(IStudentsRepository ir)
    {
        repo = ir;
    }
```

- שימו לב שאנחנו לא אילו שקוראים למאתחל של ה CONTROLLER. הסביבה עושה זאת באופן אוטומטי עבורינו, ומזריקה את הREPOSITORY לבד.