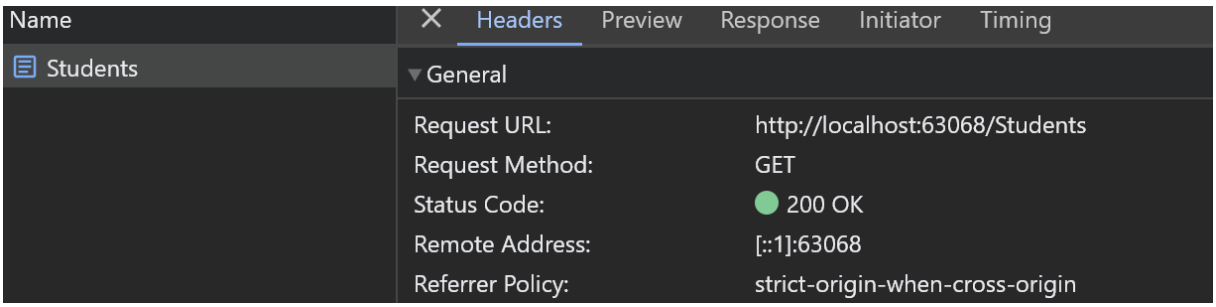
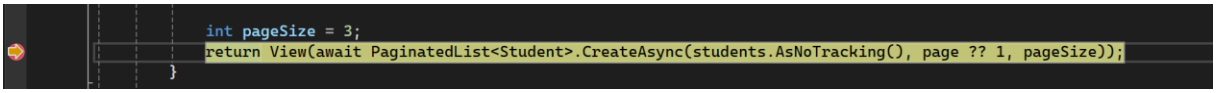
CST1 - Practicum Les1 - Webapplicatie en SQL Injection

Begrijpen van deze webapplicatie

1. *Ga naar [Students]. Welke action method wordt aangeroepen als je naar deze pagina gaat? Bewijs dit door een breakpoint in de C# code te plaatsen.*
   Als ik naar Students ga, wordt de action method "GET" aangeroepen. Dit is onder het netwerk tab in F12 te zien.



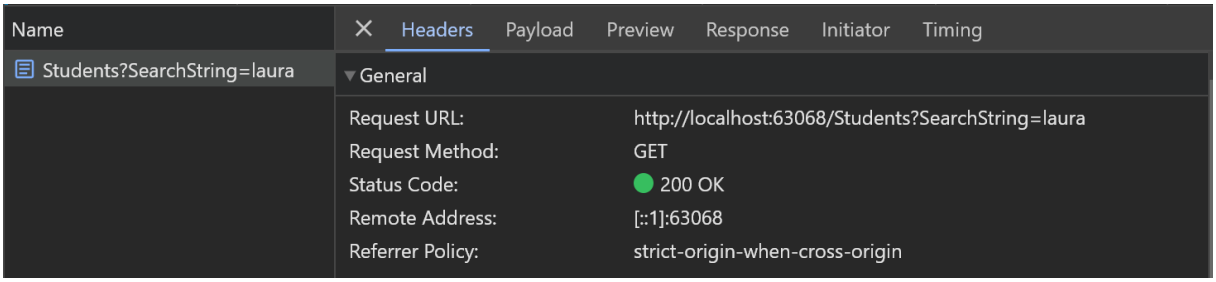
   Nu nog bewijzen via een breakpoint. Ik heb op de return view een breakpoint geplaatst. Zodra ik op de website op de students tab klik, blijft de website hangen op de breakpoint. De return view wordt nog niet uitgevoerd. Pas zodra ik verder stap, wordt de students tab getoond. Deze return view is verantwoordelijk om de students pagina te tonen. Een pagina wordt opgehaald met de GET functie.



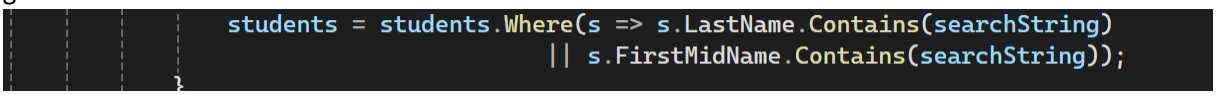
2. *Waar staat de HTML code van deze pagina?*

   Ik kan de html code vinden in: "C:\Users\20119070\Desktop\CST\ContosoLes1 - serverside\Views\Students\Index.cshtml"

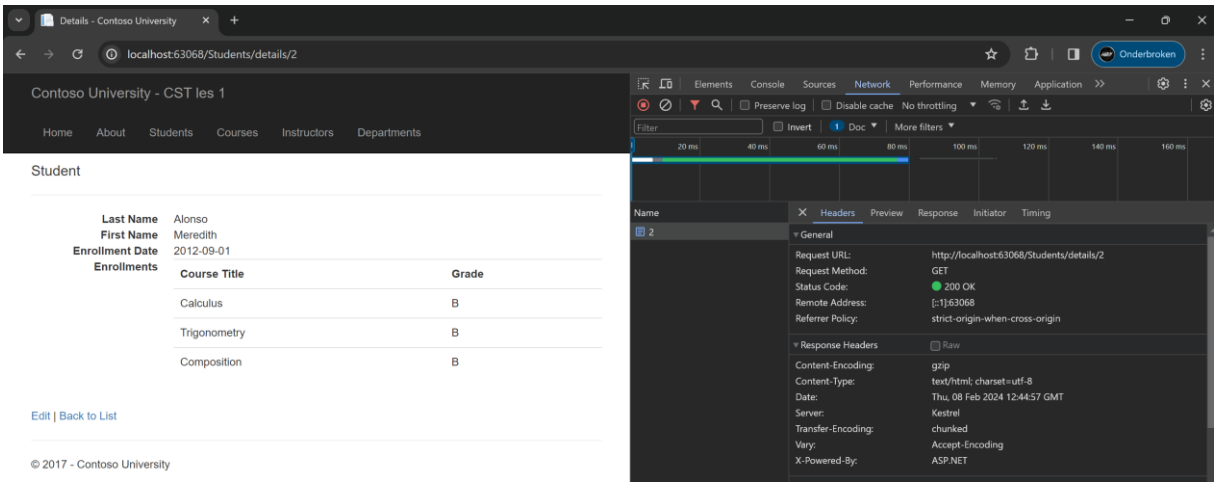
3. *Zoek in de browser in deze webpagina naar Laura. De applicatie toont nu niet meer alle studenten maar alleen Laura Norman. Bekijk via de developer console (F12) via Elements wat er gaat gebeuren als je op de "search" button klikt.*
   De pagina http://localhost:63068/Students?SearchString=laura wordt op het scherm getoond door middel van een GET functie zodra ik op "laura" zoek.



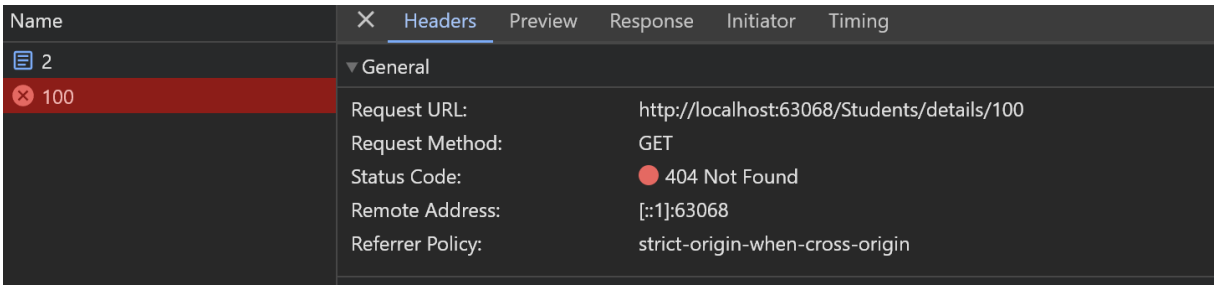
4. *Hoe ontvangt de action methode de naam die je hebt ingevuld in de webpagina? En wat doet de C# code met deze ontvangen data?*
   De GET methode roept de students aan. Deze methode students kijkt in de lijst lastname of firstmidname om te zoeken naar de studenten. Dan wordt deze informatie op de site getoond.



5. *Type het volgende in de adresbalk van de browser: http://localhost:63068/Students/details/2 Wat zie je en welke action methode wordt aangeroepen. Verder wat betekent het cijfer aan het eind? Wat gebeurt er als je dit cijfer verandert in 100? Hint: Denk aan de routing regel.*

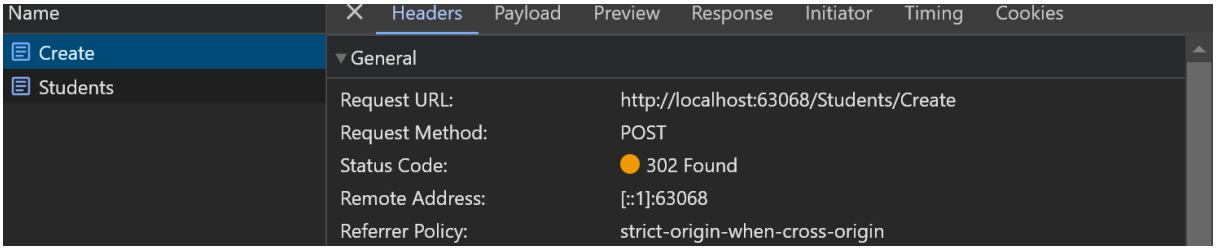


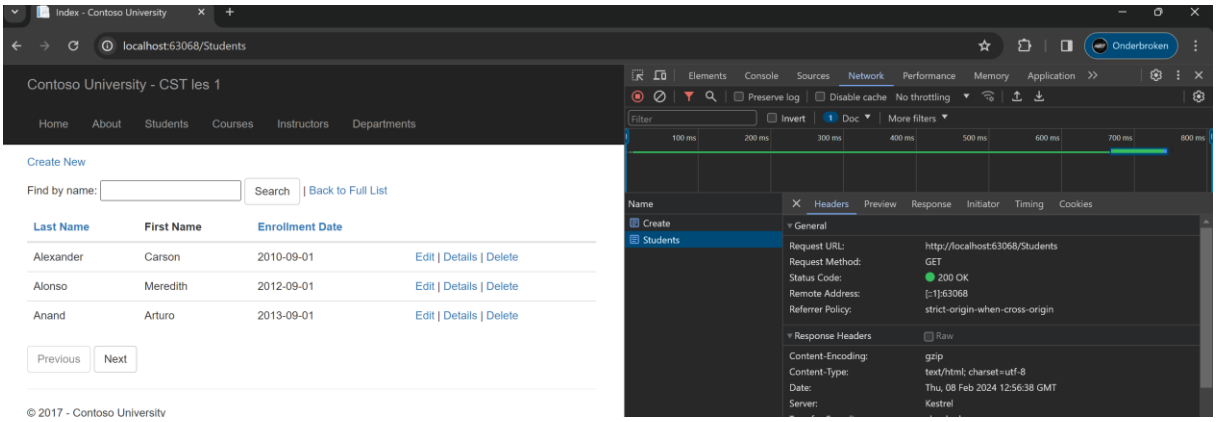
Als je naar deze pagina gaat kun je de cijfers van Meredith zien. Deze pagina wordt opgevraagd door een GET methode. Het cijfer aan het eind staat voor het nummer dat gepaard is aan de student. Elke student die wordt toegevoegd krijgt een bepaald student nummer. Meredith haar nummer is 2. Als je het nummer aanpast naar 100, wordt de pagina van student nummer 100 getoond. In dit geval is er in de database geen student 100 aanwezig. Daarom krijg je de status code 404 te zien. In de les heb ik geleerd dat een error code 400-499 een fout is vanuit de client. Ik vraag een pagina aan die niet bestaat. Vandaar deze status code.



6. *Wat betekent `return View(student);` in deze action methode? En wat is de waarde van de variabele student? Laat dit zien mbv een breakpoint.*
De view(students) geeft je de opgehaalde studentengegevens op basis van de vorige ingevoerde condities.

7. *Maak een nieuw student aan maar laat zien in de developer console (F12) wat het http request en response is als je submit.*





Als ik een student submit, vind er een POST request plaats. De data die ik heb ingevoerd wordt meegegeven aan de database. Hierna vind er een GET request plaats die de student pagina terug opbrengt.

8. *Als het goed is in het http verkeer een 302 (redirect). Welke C# code is hier voor verantwoordelijk en waar staat deze?*
   Om deze code te zien ga ik naar: “C:\Users\20119070\Desktop\CST\ContosoLes1 - serverside\Controllers\StudentsController.cs”. Onderaan is een “return View(student);” te zien. Dit brengt de gebruiker terug naar de pagina “student”.

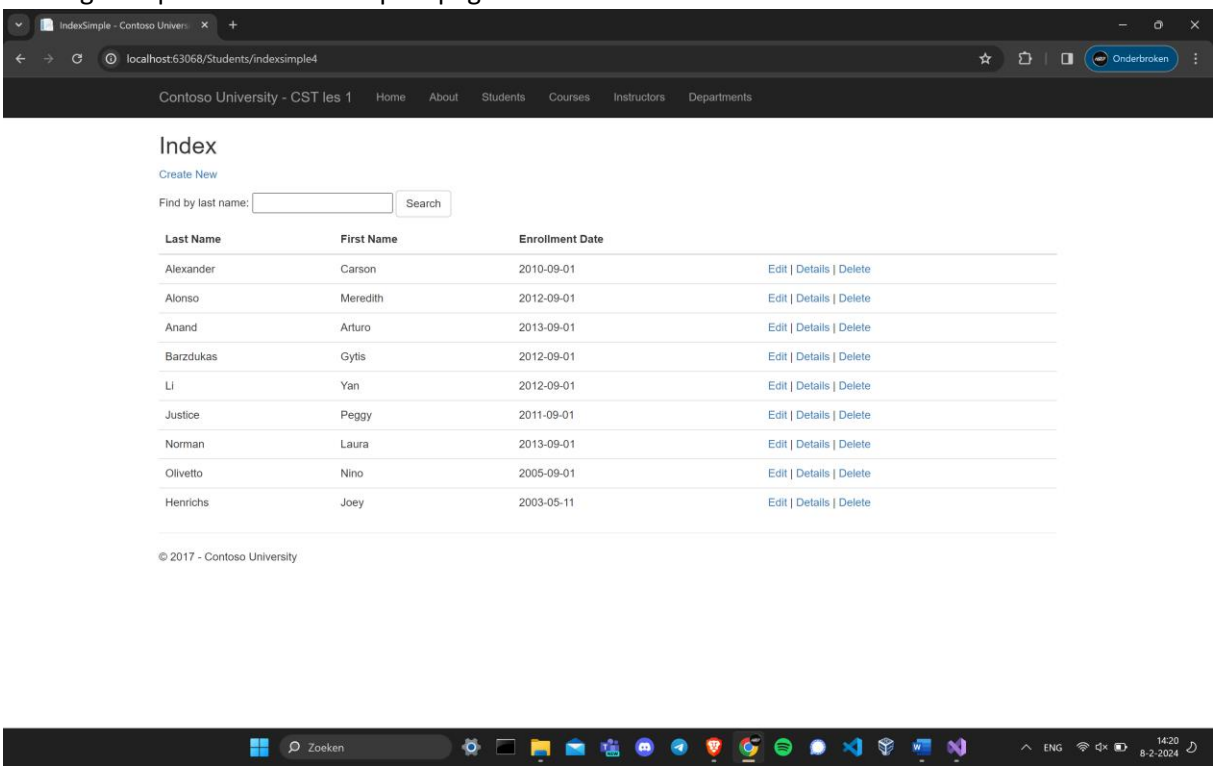
```
// POST: Students/Create
// To protect from overposting attacks, please enable the specific properties you want to bind to, for
// more details see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> Create(
    [Bind("EnrollmentDate,FirstMidName,LastName")] Student student)
{
    try
    {
        if (ModelState.IsValid)
        {
            _context.Add(student);
            await _context.SaveChangesAsync();
            return RedirectToAction("Index");
        }
    }
    catch (DbUpdateException /* ex */)
    {
        //Log the error (uncomment ex variable name and write a log.
        ModelState.AddModelError("", "Unable to save changes. " +
            "Try again, and if the problem persists " +
            "see your system administrator.");
    }
    return View(student);
}
```

Testen op Kwetsbaarheid SQL injection

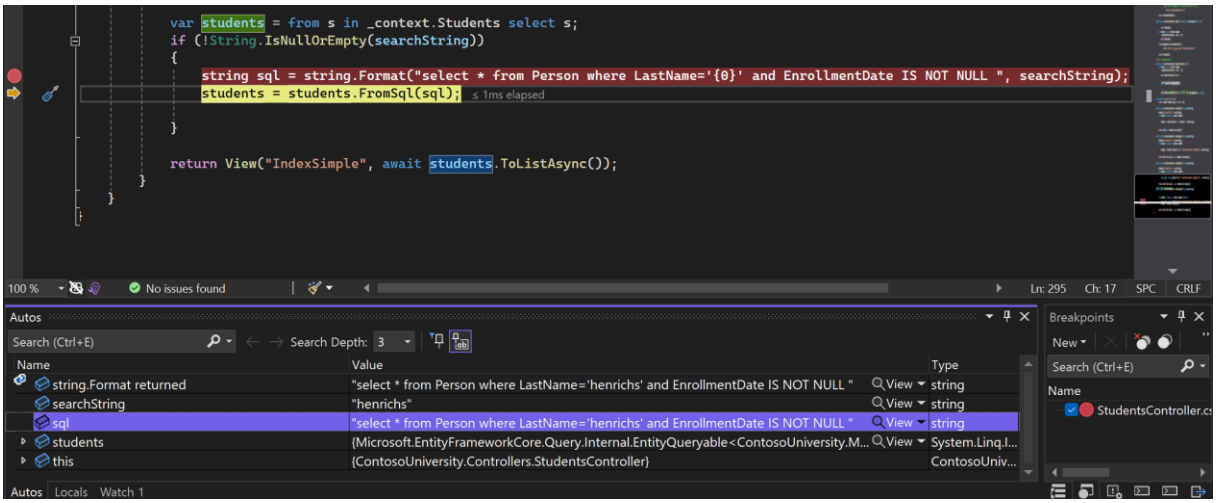
9. *Ga naar het volgende adres: http://localhost:63068/Students/indexsimple4 en zoek op een achternaam. Welk SQL statement wordt naar de database verstuurd? Dit kan je achterhalen door een breakpoint te plaatsen*

Ik begin met een breakpoint te zetten op de sql code.

Vervolgens open ik de indexsimple4 pagina.

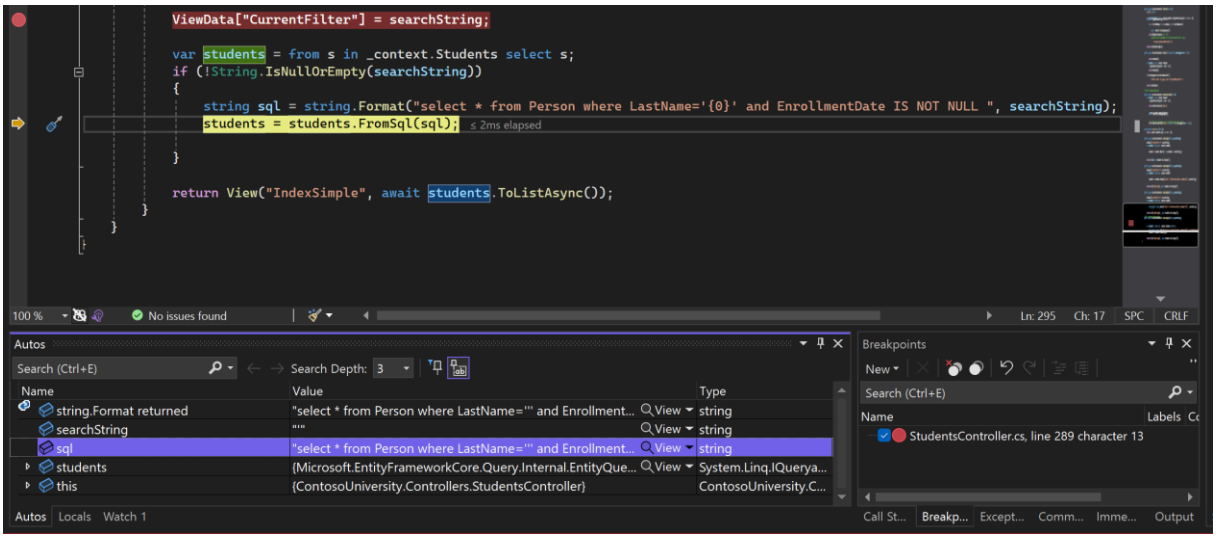


Hier zoek ik op een achternaam. De pagina blijft hangen. Ik druk op een stap verder en kan nu het sql statement zien. Dit statement is: select * from Person where LastName='henrichs' and EnrollmentDate IS NOT NULL.



Dit had ik ook al kunnen zien in de code aangezien het stukje {0} gelijk is aan de user input.

10. *Wat gebeurt er als je een single quote invult en dan zoekt? Welk SQL statement wordt nu naar de database verstuurd?*
    Als je een single quote invult is de sql statement: "select * from Person where LastName='’' and EnrollmentDate IS NOT NULL "



11. *Maak nu een sql injection waarbij je een OR gebruikt waardoor je zoekt op niet bestaande naam maar nu alle studenten als zoek resultaat terug krijgt. Let goed op je single quote en of het totale sql statement dat naar de database verstuurd wordt geen syntax fout bevat. Noteer zowel wat je hebt ingevuld en wat het totale sql statement is.*
    Ik heb “Jan' OR '1'='1” ingevoerd op de website. Jan is een naam die niet in de database stond. Hierbij zorged 1=1 voor het vertonen van alle studenten. De sql statement is dan: “select * from Person where LastName='Jan' OR '1'='1' and EnrollmentDate IS NOT NULL”

Index

Create New

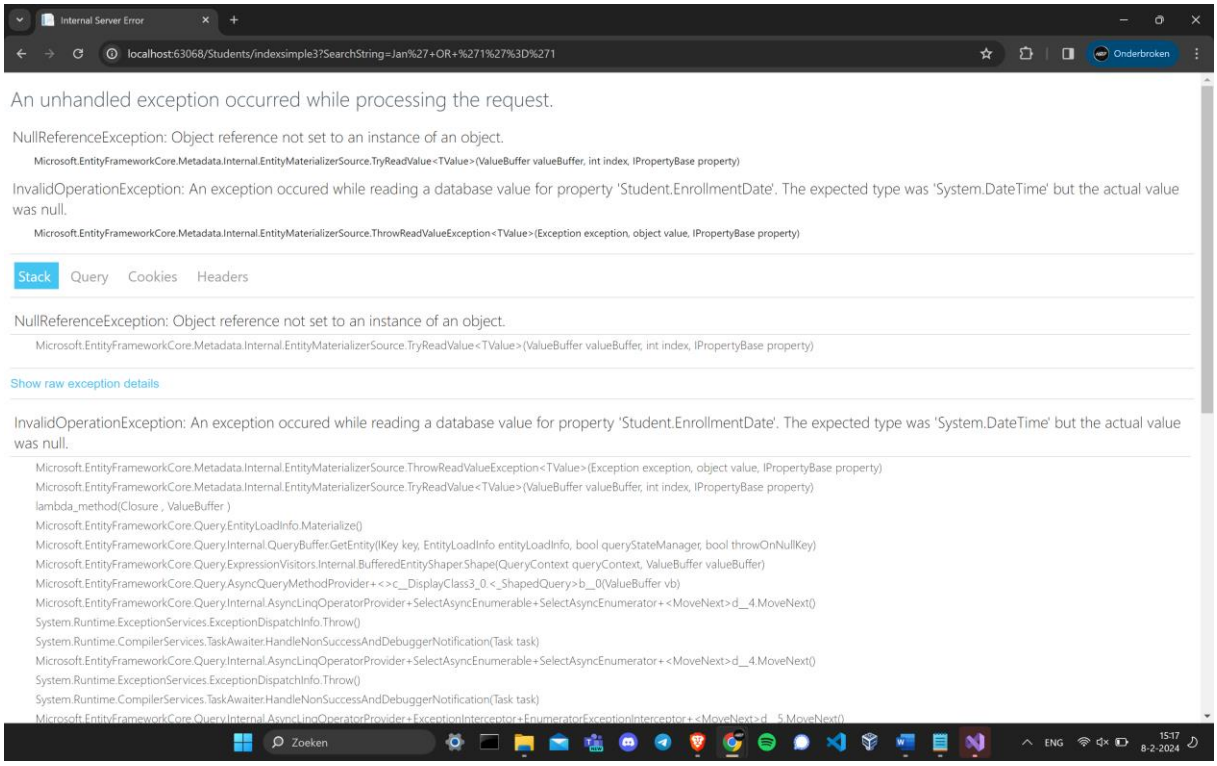
Find by last name: Jan' OR '1'= '1 Search

Last Name	First Name	Enrollment Date	
Alexander	Carson	2010-09-01	Edit | Details | Delete
Alonso	Meredith	2012-09-01	Edit | Details | Delete
Anand	Arturo	2013-09-01	Edit | Details | Delete
Barzdukas	Gytis	2012-09-01	Edit | Details | Delete
Li	Yan	2012-09-01	Edit | Details | Delete
Justice	Peggy	2011-09-01	Edit | Details | Delete
Norman	Laura	2013-09-01	Edit | Details | Delete
Olivetto	Nino	2005-09-01	Edit | Details | Delete
Henrichs	Joey	2003-05-11	Edit | Details | Delete

© 2017 - Contoso University

12. *Wat gebeurt er als je de injection van de vorige vraag invult bij http://localhost:63068/Students/indexsimple3*

    Als je de injection van de vorige pagina op deze pagina invoert krijg je een error. Hieruit kan ik concluderen dat de sql statement hier anders is.



Dit is te bevestigen als ik naar de code kijk. Hierbij is de statement: "select * from Person where LastName='{0}'".

```
0 references
public async Task<IActionResult> IndexSimple3(string searchString)
{

    ViewData["CurrentFilter"] = searchString;

    var students = from s in _context.Students
                   select s;
    if (!String.IsNullOrEmpty(searchString))
    {
        string sql = string.Format("select * from Person where LastName='{0}'", searchString);
        students = students.FromSql(sql);

    }
    return View("IndexSimple", await students.ToListAsync());
}
```

13. *Ga naar [View][SQL Server object explorer] en open de data van de tabel Person. Vul nu in alle rows een enrollment date in.*

Table voor aanpassingen:

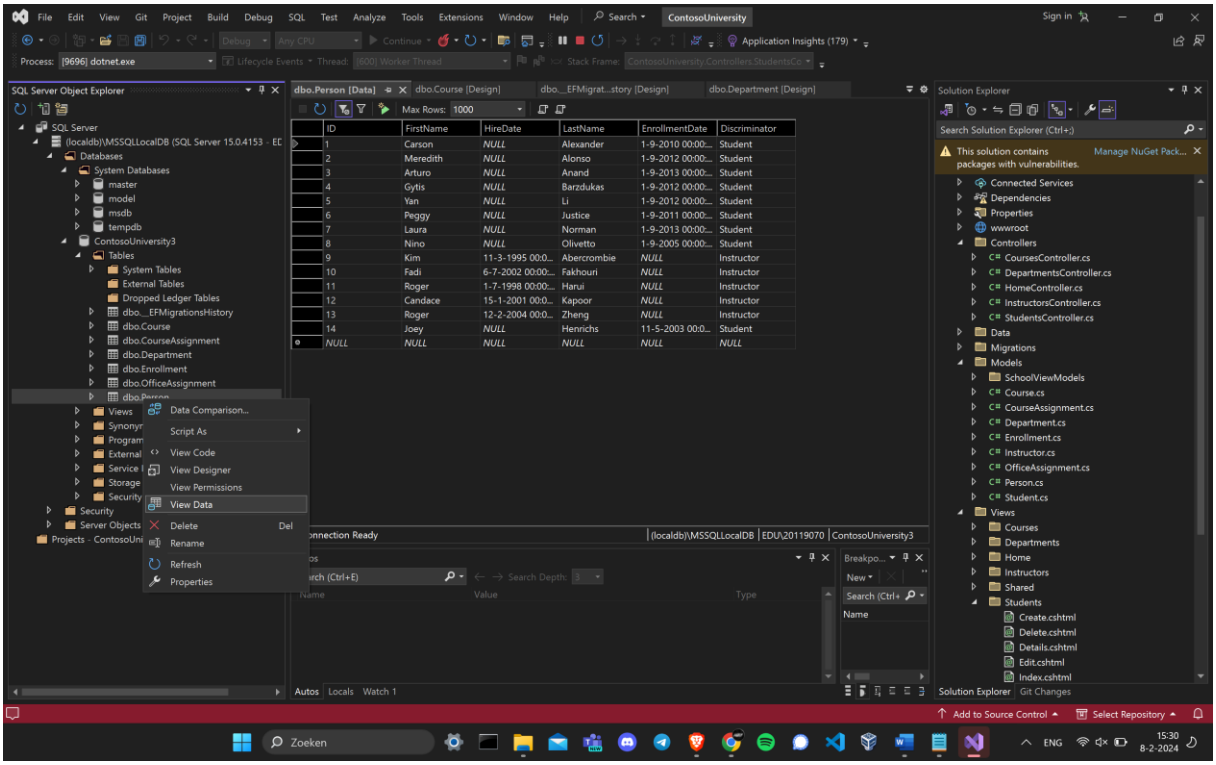


Table na aanpassingen:

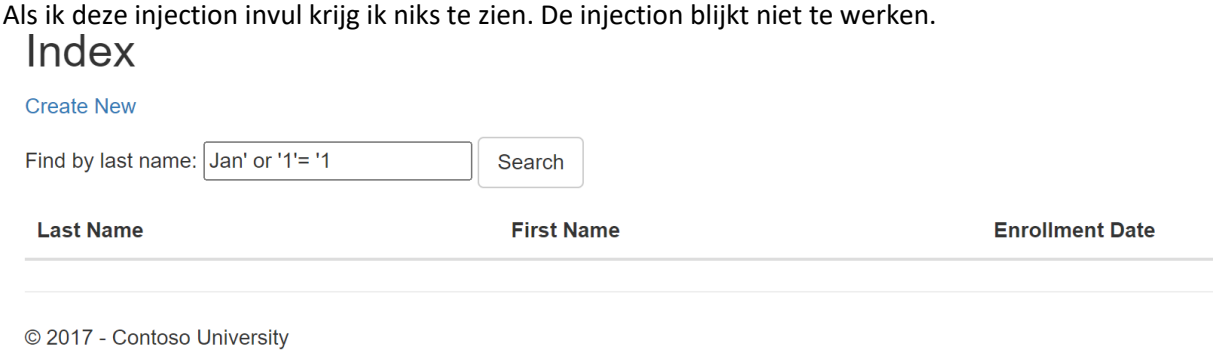
dbo.Person [Data] | dbo.Course [Design] | dbo.__EFMigrat...story [Design] | dbo.Department [Design]

Max Rows: 1000

ID	FirstName	HireDate	LastName	EnrollmentDate	Discriminator
1	Carson	NULL	Alexander	1-9-2010 00:00:...	Student
2	Meredith	NULL	Alonso	1-9-2012 00:00:...	Student
3	Arturo	NULL	Anand	1-9-2013 00:00:...	Student
4	Gytis	NULL	Barzdukas	1-9-2012 00:00:...	Student
5	Yan	NULL	Li	1-9-2012 00:00:...	Student
6	Peggy	NULL	Justice	1-9-2011 00:00:...	Student
7	Laura	NULL	Norman	1-9-2013 00:00:...	Student
8	Nino	NULL	Olivetto	1-9-2005 00:00:...	Student
9	Kim	11-3-1995 00:0...	Abercrombie	11-5-2003 00:0...	Instructor
10	Fadi	6-7-2002 00:00:...	Fakhouri	11-5-2003 00:0...	Instructor
11	Roger	1-7-1998 00:00:...	Harui	11-5-2003 00:0...	Instructor
12	Candace	15-1-2001 00:0...	Kapoor	11-5-2003 00:0...	Instructor
13	Roger	12-2-2004 00:0...	Zheng	11-5-2003 00:0...	Instructor
14	Joey	NULL	Henrichs	11-5-2003 00:0...	Student
NULL	NULL	NULL	NULL	NULL	NULL

14. *Wat gebeurt er nu als je de injection van vraag 12 herhaalt?*
De injection werkt nu wel. Dit komt aangezien er nu wel een enrollment date aanwezig is voor alle studenten. Bij indexsimple4 werkte de sql statement nog wel als niet iedereen hier een datum had. Bij 3 niet.

15. *Wat gebeurt er als je de injection bij http://localhost:63068/Students/indexsimple2 invult?*
Wat is er verschillend in de code?
Als ik deze injection invul krijg ik niks te zien. De injection blijkt niet te werken.



In de code van de vorige 2 pagina's werd de invoer rechtstreeks in de SQL query gebruikt. Hierdoor kan de gebruiker deze makkelijk manipuleren.

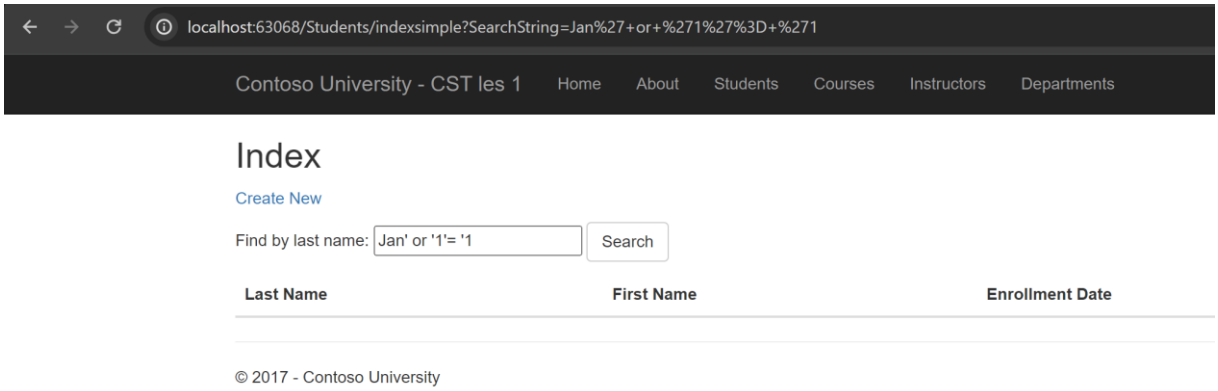
```
string sql = string.Format("select * from Person where LastName='{0}'", searchString);
students = students.FromSql(sql);
```

Bij indexsimple2 bevat de code niet meer het stukje "string sql =". De SQL code is gescheiden van de gebruikersinvoer. Hierdoor werkt de SQL injection niet meer.

```
if (!String.IsNullOrEmpty(searchString))
{
    students = students.FromSql("select * from Person where LastName={0}", searchString);
}
```

16. *Zelfde vraag voor http://localhost:63068/Students/indexsimple*

    Bij indexsimple werkt de SQL injection niet.

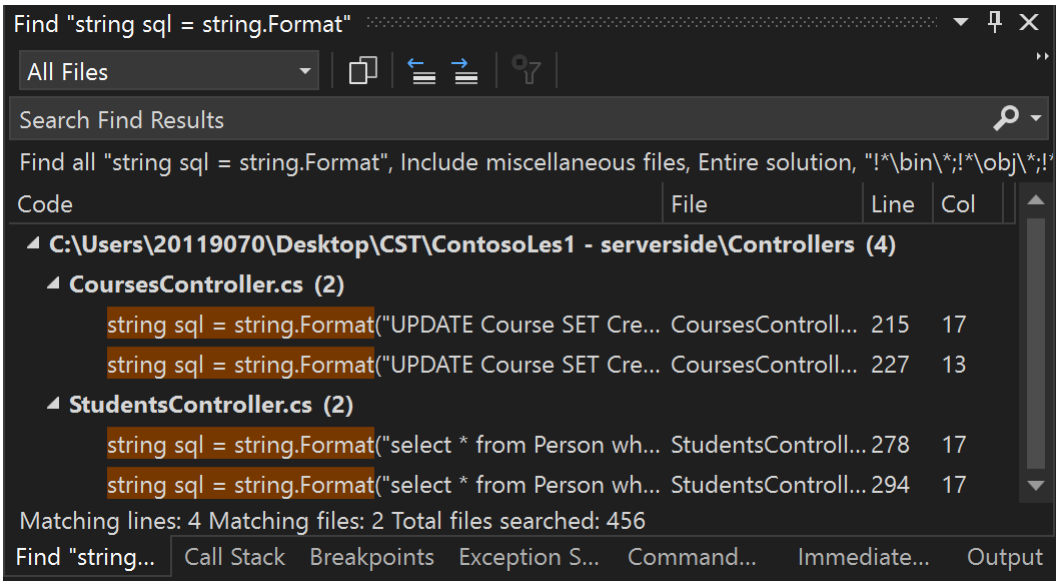


    Als we naar de code kijken zien wij dat er niet meer gebruik wordt gemaakt van gebruikersinvoer in de SQL query, maar dat er gebruik wordt gemaakt van een language integrated query. Hierdoor heb je niet meer dat jouw invoer in een query wordt gebruikt maar dat er in de C# een geïntegreerde code wordt gebruikt om de query op te bouwen.

```
public async Task<IActionResult> IndexSimple(string searchString)
{

    ViewData["CurrentFilter"] = searchString;

    var students = from s in _context.Students
                   select s;
    if (!String.IsNullOrEmpty(searchString))
    {
        students = students.Where(s => s.LastName == searchString);
    }



    return View(await students.ToListAsync());
}
```

17. *Welke van de 4 verschillende versie van indexsimpleX methodes zijn nu kwetsbaar voor sql injection?*
    Indexsimple3 en indexsimple4 zijn kwetsbaar voor sql injection.

18. *Waarom kan je op basis van de source code nu heel snel achterhalen of dezelfde voor sql injection kwetsbare code ook elders in deze applicatie wordt gebruikt? Hint [Edit}{Find in files]*
    Doordat wij nu de source code hebben kan je op stukken kwetsbare code zoeken. Hierdoor is het gelijk zichtbaar waar dezelfde kwetsbare code wordt gebruikt. Hierdoor heb je in 1 keer een overzicht van alle locaties van deze kwetsbare code. Hierdoor heb je snel in handen welke code kwetsbaar is.



19. *Waarom zou een blackbox pentest veel meer tijd kosten om deze zelfde kwetsbare code te achterhalen?*
    Aangezien als je begrijpt hoe SQL werkt, kun je al gelijk snel zien welke code kwetsbaar is voor een SQL injection. Hierdoor zie je snel de fouten binnen de code. Een blackbox pentest is van buiten af en heeft de code niet in handen. Hierdoor zijn de fouten niet gelijk zichtbaar. Deze test zou allemaal SQL injections pogen wat dus een tijd inneemt.