Лабораторная работа

Анализ и повышение качества кода инструментами Visual Studio 2013

Lab version: 12.0.30723.00 Update 3

Last updated: 12/17/2013

СОДЕРЖАНИЕ

Резюме

Функция анализа кода в Visual Studio производит статический анализ с целью упростить процесс выявления проблем с архитектурой, взаимодействием, производительностью, безопасностью и т.д. Анализ кода можно запустить как вручную, так и настроить его как часть процесса Team Build или чекина в Team Foundation Server. В этой лабораторной работе мы рассмотрим функцию анализа кода, настройку правил и подавление на уровне проекта и исходного кода.

Примечание: эта функция доступна в Visual Studio 2012 Professional, Premium и Ultimate. Основные возможности доступны в редакции Express бесплатно (C++, C#, Visual Basic).

Prerequisites

Для выполнения лабораторной работы вам понадобится виртуальная машина с Visual Studio 2013. Подробнее про то, где загрузить и как ее использовать, здесь.

О компании Fabrikam Fiber

Эти лабораторные работы в качестве основы для сценариев, о которых вы узнаете в процессе, оперируют несуществующей компанией Fabrikam Fiber. Fabrikam Fiber занимается кабельным телевидением и сопутствующими сервисами в США. Компания быстро растет и уже начала использовать Microsoft Azure для того, чтобы масштабировать свой веб-сайт для обслуживания их запросов и отслеживания деятельности инженеров. Компания использует локальное приложение ASP.NET MVC для управления заказами клиентов.

В этих лабораторных работах вы изучите сценарии, включенные в рабочий процесс команды разработки и тестирования Fabrikam Fiber. Команда, состоящая из 8-10 человек, решила использовать средства управления жизненным циклом проектов Visual Studio 2013 для того, чтобы контролировать программный код, выполнять сборки, тестировать веб-сайты, планировать и отслеживать происходящее с проектом

Упражнения

Эта лабораторная работа включает в себя следующие упражнения

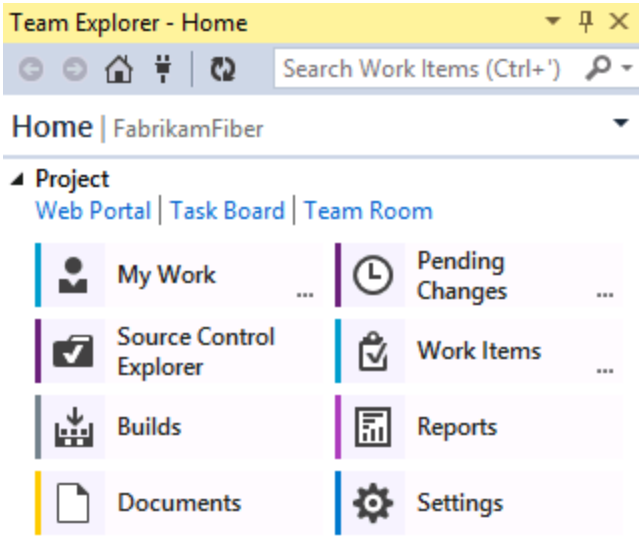
1. Введение в Code Analysis
2. Подавление предупреждений при анализе кода

Примерное время выполнения лабораторной работы: **30 минут**

Упражнение 1: Введение в Code Analysis

В этом упражнении вы научитесь использовать функцию Code Analysis, доступную в Visual Studio 2013, настраивать набор правил, выполнять анализ кода на тестовом проекте и разрешать некоторые из предупреждений.

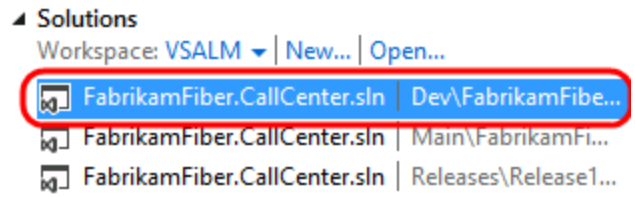
1. Войдите под аккаунтом **Julia Ilyiana** (VSALM\Julia). Пароль: **P2ssw0rd.**
2. Запустите **Visual Studio 2013** и откройте **Team Explorer**. Вы должны быть подключены к командному проекту FabrikamFiber, если этого не произошло, нажмите **Connect to Team Projects** ( ) и инициируйте подключение.



Изображение 1

Team Explorer - Home

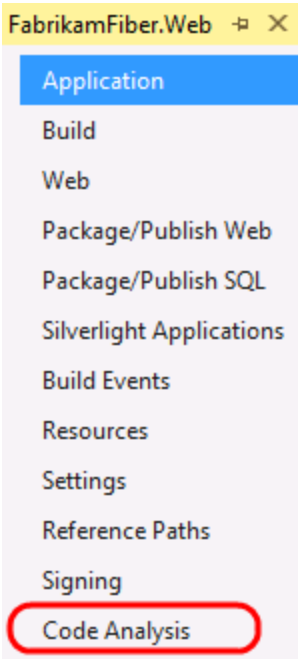
3. В **Team Explorer – Home** н**ажмите два раза** на первом решении **FabrikamFiber.CallCenter.sln**.



Изображение 2

Открытие решения Fabrikam Fiber

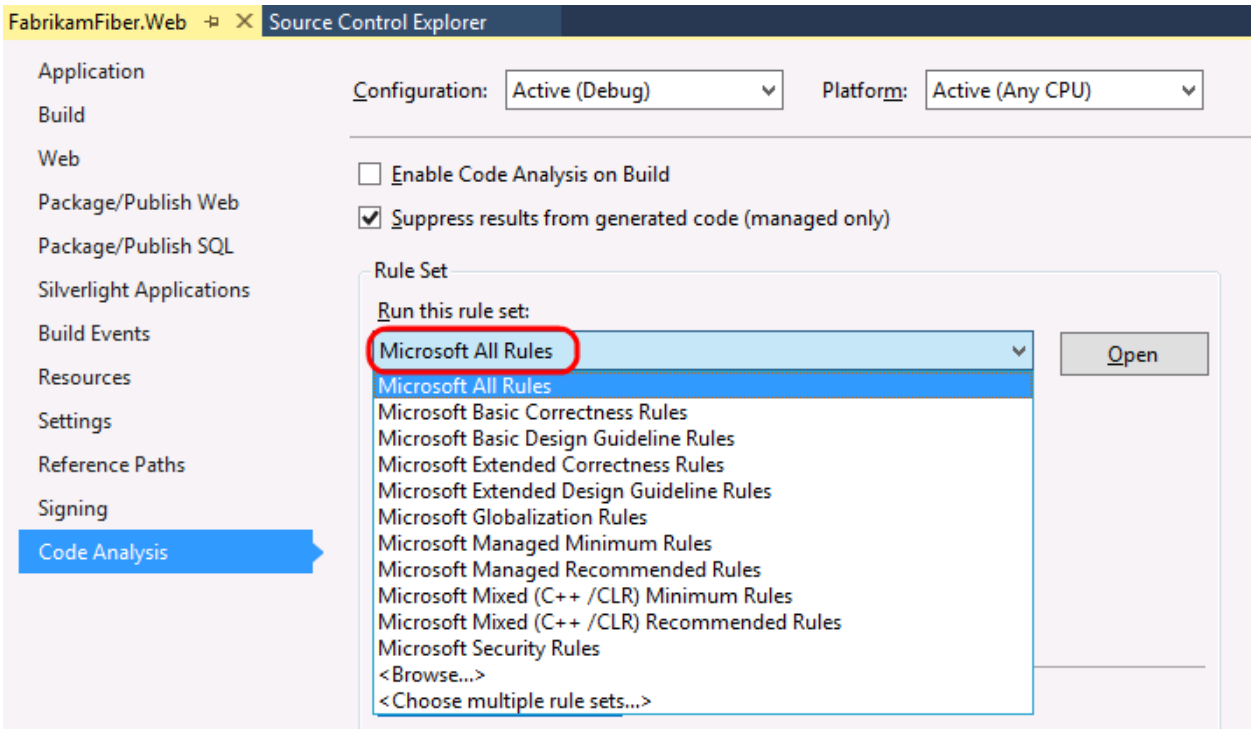
4. Пересоберите решение (**Build | Rebuild Solution** в меню).
5. В Solution Explorer нажмите правой кнопкой на **FabrikamFiber.Web** и вызовите **Properties**.
6. Перейдите на вкладку **Code Analysis** .



Изображение 3
Настройка Code Analysis

Примечание: на этой вкладке можно выбрать набор правил.

7. Выберите в Rule Set опцию “**Microsoft All Rules”**.

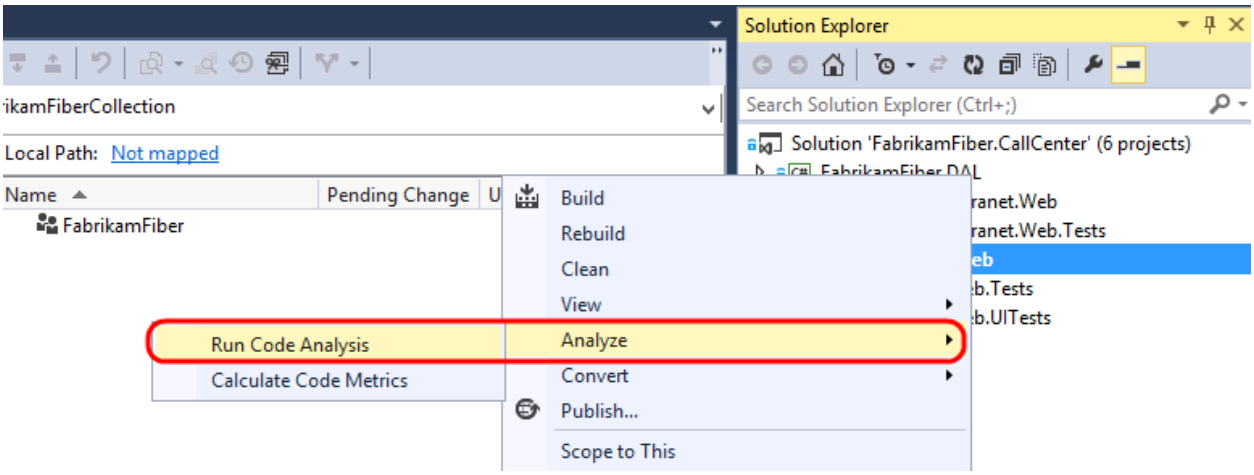


Изображение 4

Настройка набора правил для Code Analysis

> **Примечание:** в Visual Studio редакций Professional, Premium и Ultimate можно создавать собственные правила для C++.

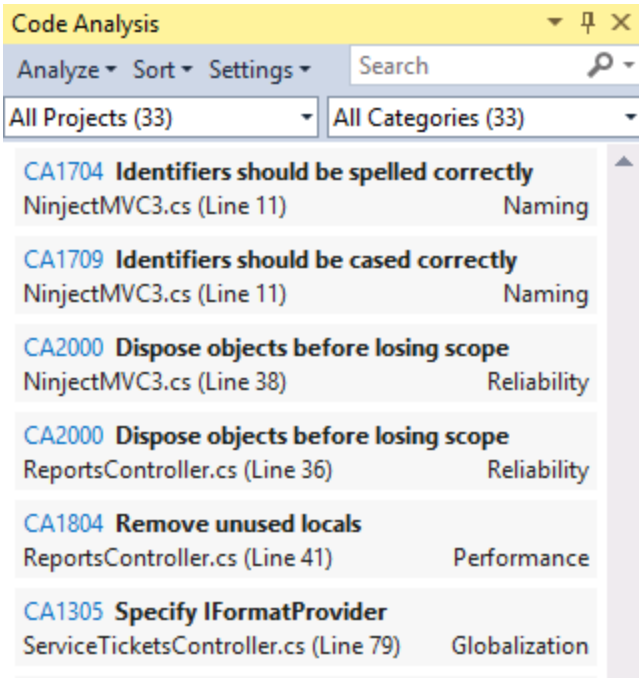
8. В Solution Explorer нажмите правой кнопкой на проекте **FabrikamFiber.Web**. Нажмите **Analyze** | **Run Code Analysis**.



Изображение 5

Запуск Code Analysis

9. Code Analysis использует правила статического анализа, определенные Microsoft, и показывает результаты в окне Code Analysis. Изучите результаты.

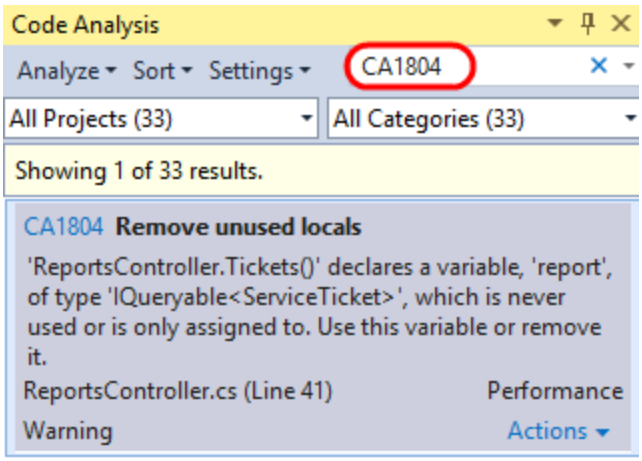


Изображение 6

Результаты Code Analysis

> **Примечание:** в зависимости от версии проекта FabrikamFiber у вас может отображаться разное количество результатов. Правила Code Analysis могут быть настроены таким образом, чтобы показывать еще и ошибки.

10. В результате анализа Code Analysis мы получаем большое количество информации в виде предупреждений, включая идентификатор категории (например, CA2000), название, описание проблемы, расположение файла и предлагаемое решение.
11. В окне Code Analysis можно использовать текстовое поле для фильтрации результатов по номеру и тексту в названии, сообщении, пути к файлу и названию функции.



Изображение 7

Фильтрация результатов Code Analysis

12. Найдите предупреждение, которое будет просто разрешить (например, предупреждение CA1804). **Нажмите два раза** на нем для перехода в место, где возникла проблема.

```
0 references | Julia Ilyiana | 1 change
public ActionResult Tickets()
{
    var report = this.serviceTicketRepository.AllForReport(
        serviceticket => serviceticket.Customer,
        serviceticket => serviceticket.CreatedBy,
        serviceticket => serviceticket.AssignedTo);

    return View();
```

Изображение 8

Ошибки Code Analysis привязываются к коду

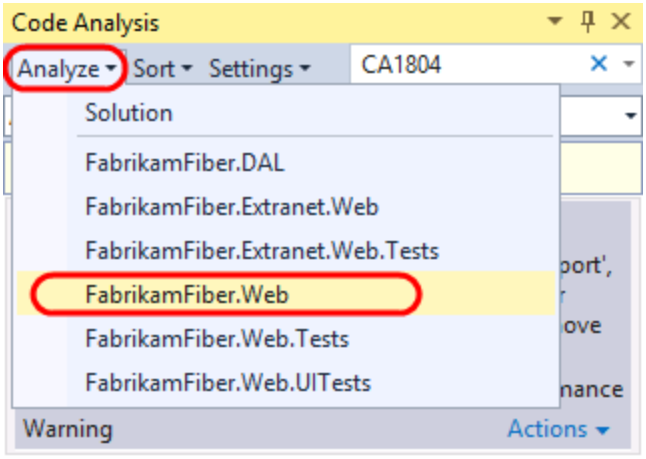
13. Исправьте ошибку. Для CA1804 нужно убрать неиспользуемые переменные – сделайте это, убрав определение report.

```
0 references | Julia Ilyiana | 1 change
public ActionResult Tickets()
{
    this.serviceTicketRepository.AllForReport(
        serviceticket => serviceticket.Customer,
        serviceticket => serviceticket.CreatedBy,
        serviceticket => serviceticket.AssignedTo);

    return View();
}
```

Изображение 9

Удаление неиспользуемых переменных

14. В окне **Code Analysis** выберите **Analyze | FabrikamFiber.Web** и убедитесь, что предупреждение исчезло.



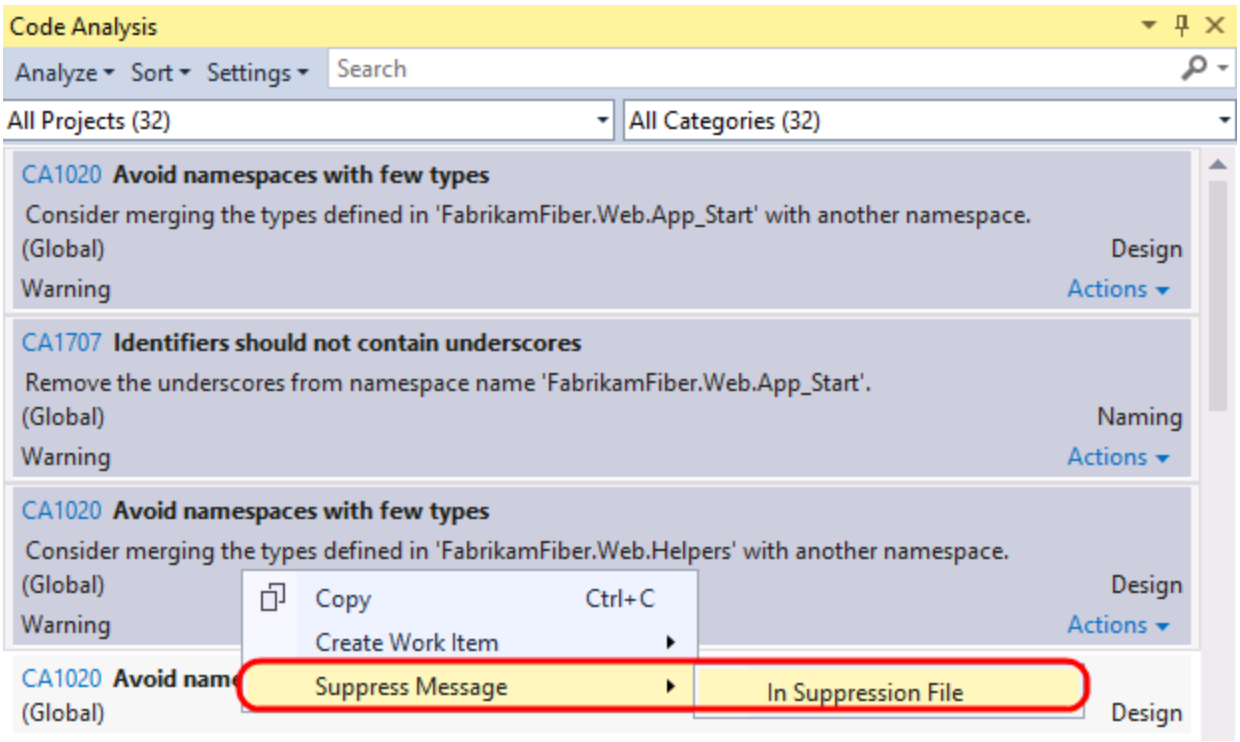
Изображение 10
Запуск анализа кода для проверки исправления

Упражнение 2: Подавление предупреждений при анализе кода

В этом упражнении вы научитесь подавлять предупреждения Code Analysis на уровне проекта и исходного кода.

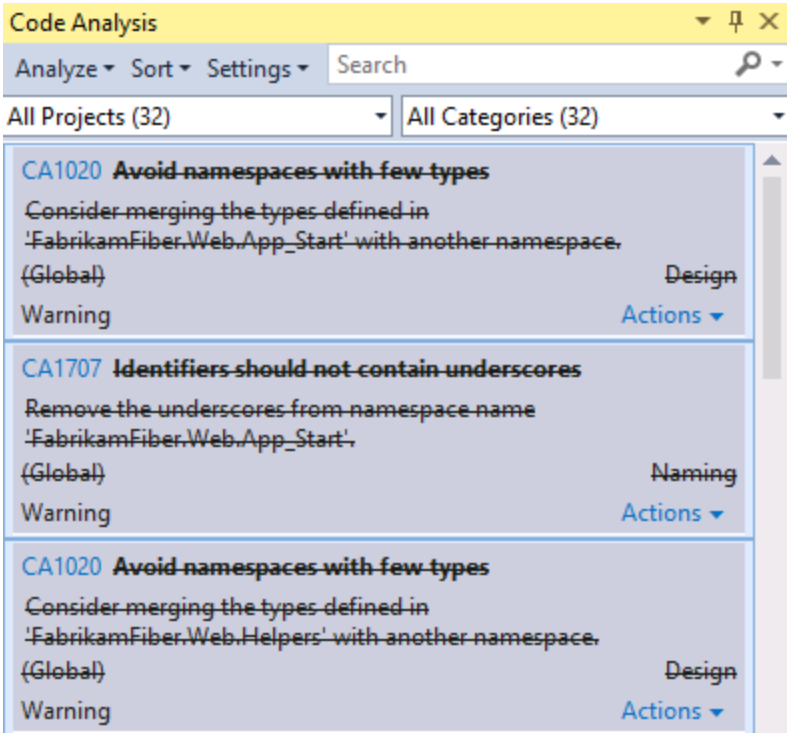
1. Выберите в **Code Analysis** первые три предупреждения. Предположим, мы не собираемся решать эти проблемы и не хотим, чтобы они появлялись при следующем анализе.
2. При выборе предупреждений они будут развернуты. Нажмите правой кнопкой на выбранных предупреждениях и нажмите **Suppress Message | In Suppression File**. После этого в файл GlobalSuppressions.cs будут добавлены метаданные уровня сборки.

> **Примечание:** то же самое можно сделать с помощью ссылки Actions.



Изображение 11

Подавление предупреждений



Изображение 12

Подавленные предупреждения зачеркиваются

3. Откройте файл **GlobalSuppressions.cs**.

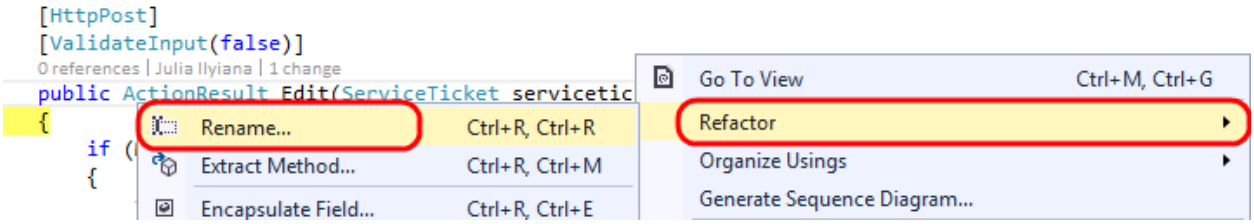
```
// This file is used by Code Analysis to maintain SuppressMessage
// attributes that are applied to this project.
// Project-level suppressions either have no target or are given
// a specific target and scoped to a namespace, type, member, etc.
//
// To add a suppression to this file, right-click the message in the
// Code Analysis results, point to "Suppress Message", and click
// "In Suppression File".
// You do not need to add suppressions to this file manually.

[assembly: System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Design", "CA1020:AvoidNamespacesWithFewTy
[assembly: System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Naming", "CA1707:IdentifiersShouldNotCont
[assembly: System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Design", "CA1020:AvoidNamespacesWithFewTy
```

Изображение 13

Содержимое GlobalSuppressions.cs

4. Вернитесь в окно **Code Analysis** и пролистайте его до конца. Выделите CA1704, предлагающий скорректировать имя параметра serviceticker. **Нажмите два раза**.

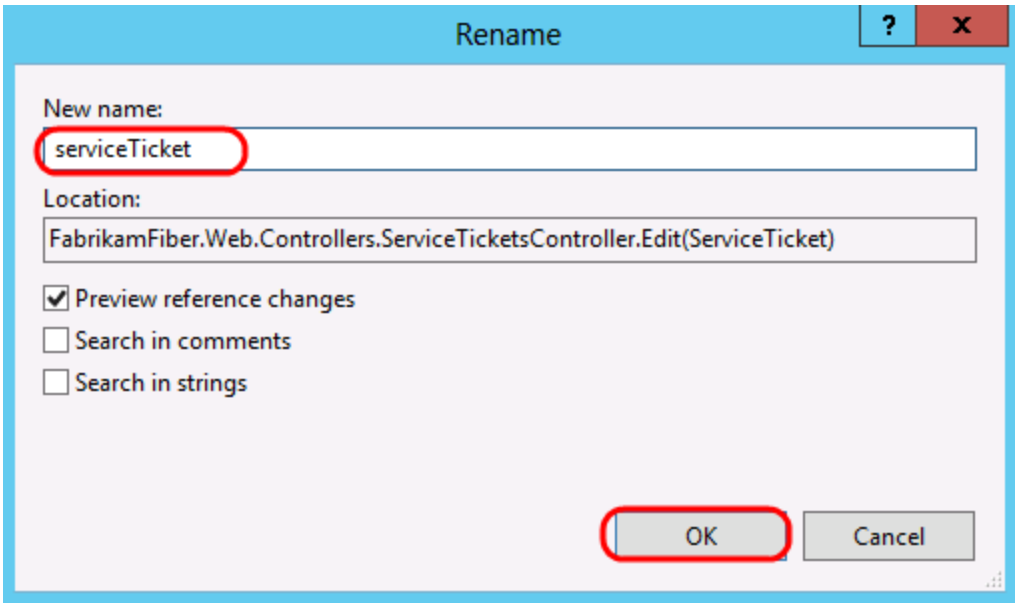
5. **Нажмите правой кнопкой** на параметре serviceticket и нажмите **Refactor | Rename…**.



Изображение 14

Переименование параметра

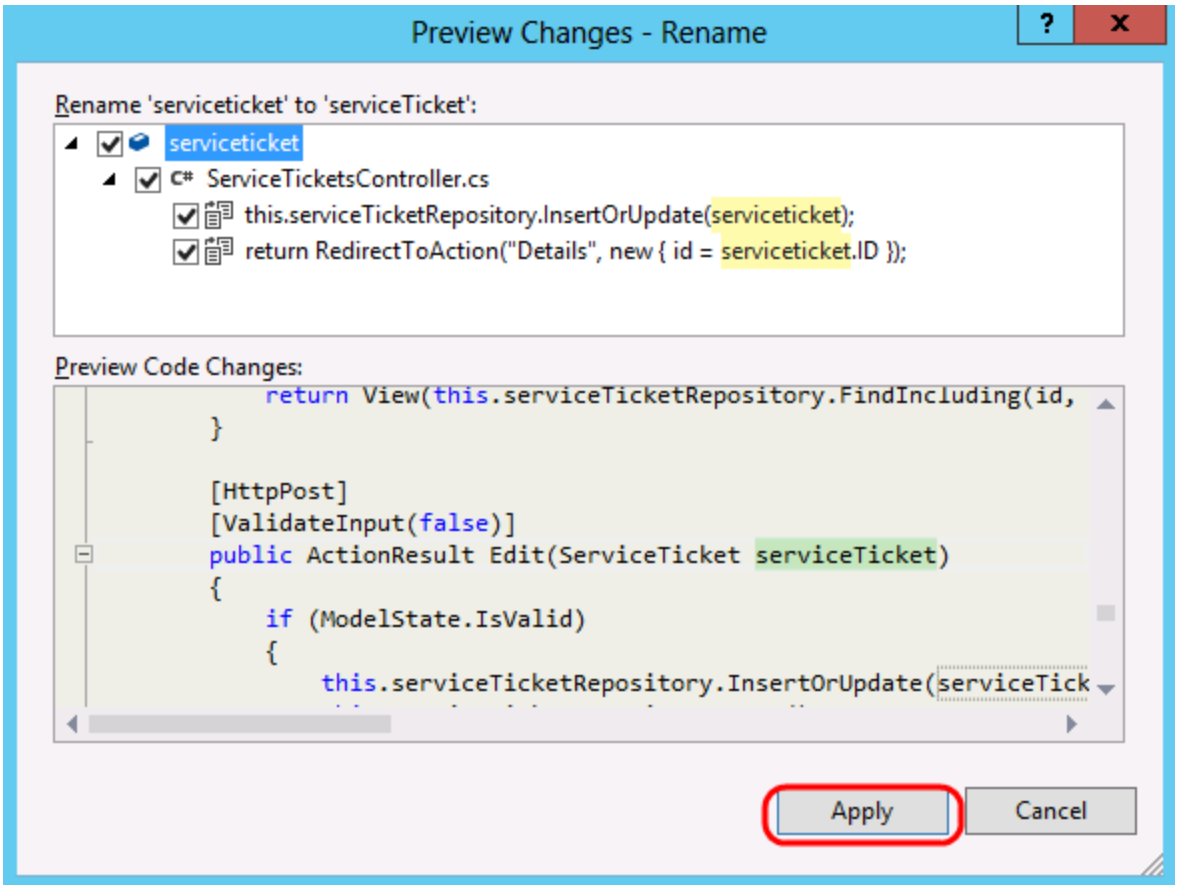
6. В окне Rename измените ‘serviceticket’ на ‘**serviceTicket’** и нажмите на **OK**.



Изображение 15

Переименование

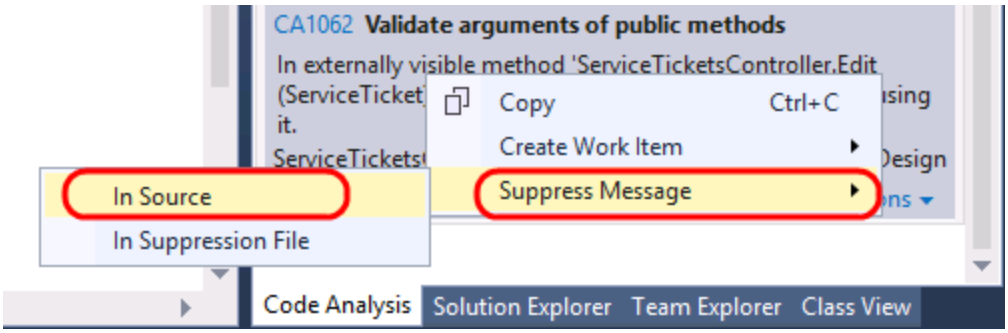
7. В **Preview Changes – Rename** нажмите на **Apply**.



Изображение 16

Preview Changes – Rename

8. Выделите следующее предупреждение. Мы хотим подавить его на уровне конкретного файла. **Нажмите правой кнопкой** на предупреждении и нажмите **Suppress Message(s) | In Source**.



Изображение 17

Кнопка In Source

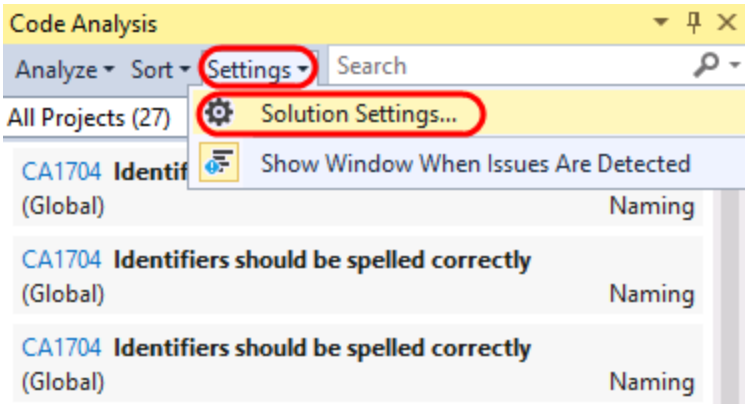
9. Правило применится к методу в виде атрибута **SuppressMessage**.

```
[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Design", "CA1062:Validate arguments of public methods", Messa
[ValidateInput(false)]
0 references | Julia Ilyiana | 1 change
public ActionResult Edit(ServiceTicket serviceTicket)
```

Изображение 18

Подавление предупреждения на уровне исходного кода

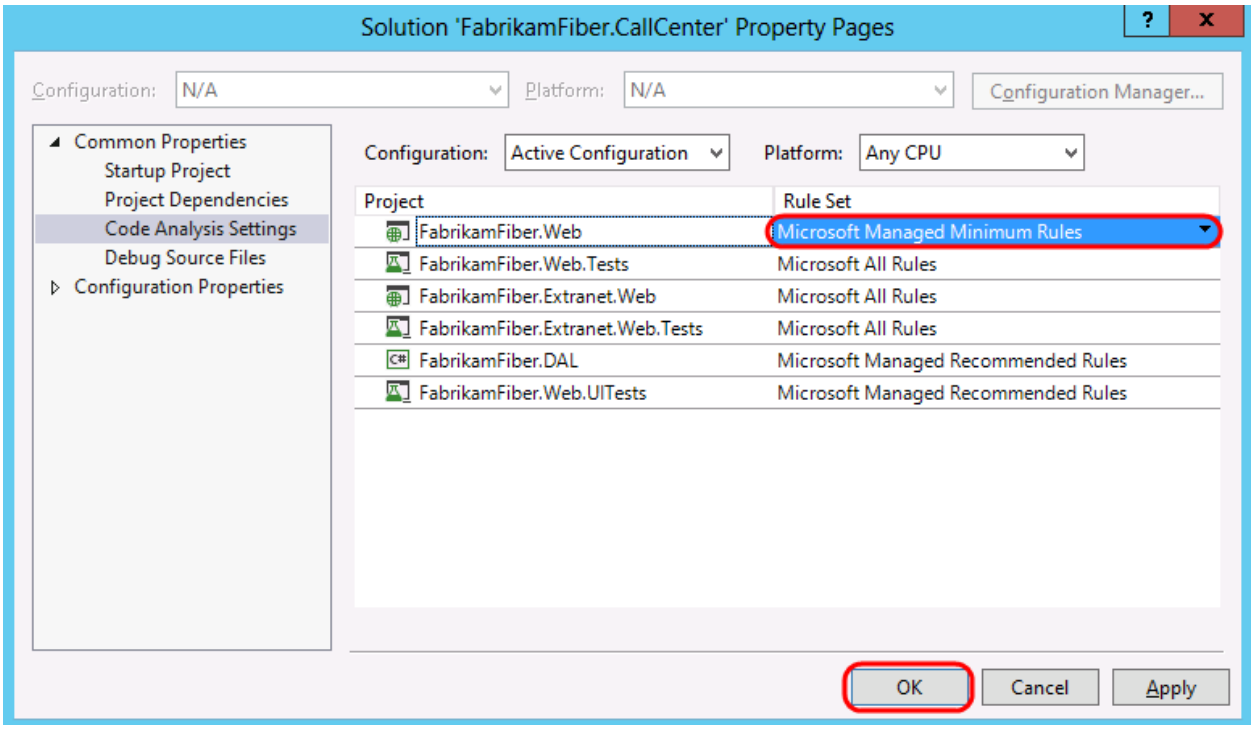
10. В окне **Code Analysis** нажмите на **Analyze | FabrikamFiber.Web** чтобы проверить, исчезло ли еще одно предупреждение.
11. Предположим, что мы хотим проигнорировать оставшиеся предупреждения. Нажмите в окне Code Analysis на **Settings | Solution Settings**.



Изображение 19

Настройки Code Analysis

12. Измените значение **Rule Set** для **FabrikamFiber.Web** с '**Microsoft All Rules**' на '**Microsoft Managed Minimum Rules**'. Нажмите на **OK**.



Изображение 20

Изменение набора правил

13. Перейдите в **Code Analysis Analyze | FabrikamFiber.Web** - количество предупреждений значительно сократилось. Предупреждения, появившиеся в результате использования нового набора правил, выглядят более критичными для работы программы.
