VARIAN ESAPI SCRIPTING EXERCISE 2

Developer Workshop 2.0 – Austin, Texas – July 18th, 2014

VARIAN
ONCOLOGY
SYSTEMS

Wayne Keranen

Product Manager, Varian APIs

July 18th, 2014

VARIAN
medical systems

Disclaimers

- Eclipse™, SmartAdapt™, ARIA® and ARIA LINK™, and TrueBeam™ are trademarked by Varian Medical Systems.
- Word™, Excel™, Office™ are trademarked by Microsoft.
- Visual Studio™ is trademarked by Microsoft.

Exercise 2 Learning Goals

We will:

1) Learn how to create a Standalone Executable ESAPI Script.
2) Learn how the context differs for a Standalone Executable.
3) Use ESAPI to create a simple data mining script to navigate deeply into the Patient / Course / Plan hierarchy.

Sign in to Virtual Eclipse Environment

- Before we start, sign in with your assigned userid/pwd to your assigned Eclipse Client.
- TBD

VARIAN medical systems

Two kinds of Eclipse scripts

- Eclipse calls you - **Plugin**

- You call Eclipse - **Standalone Executable**

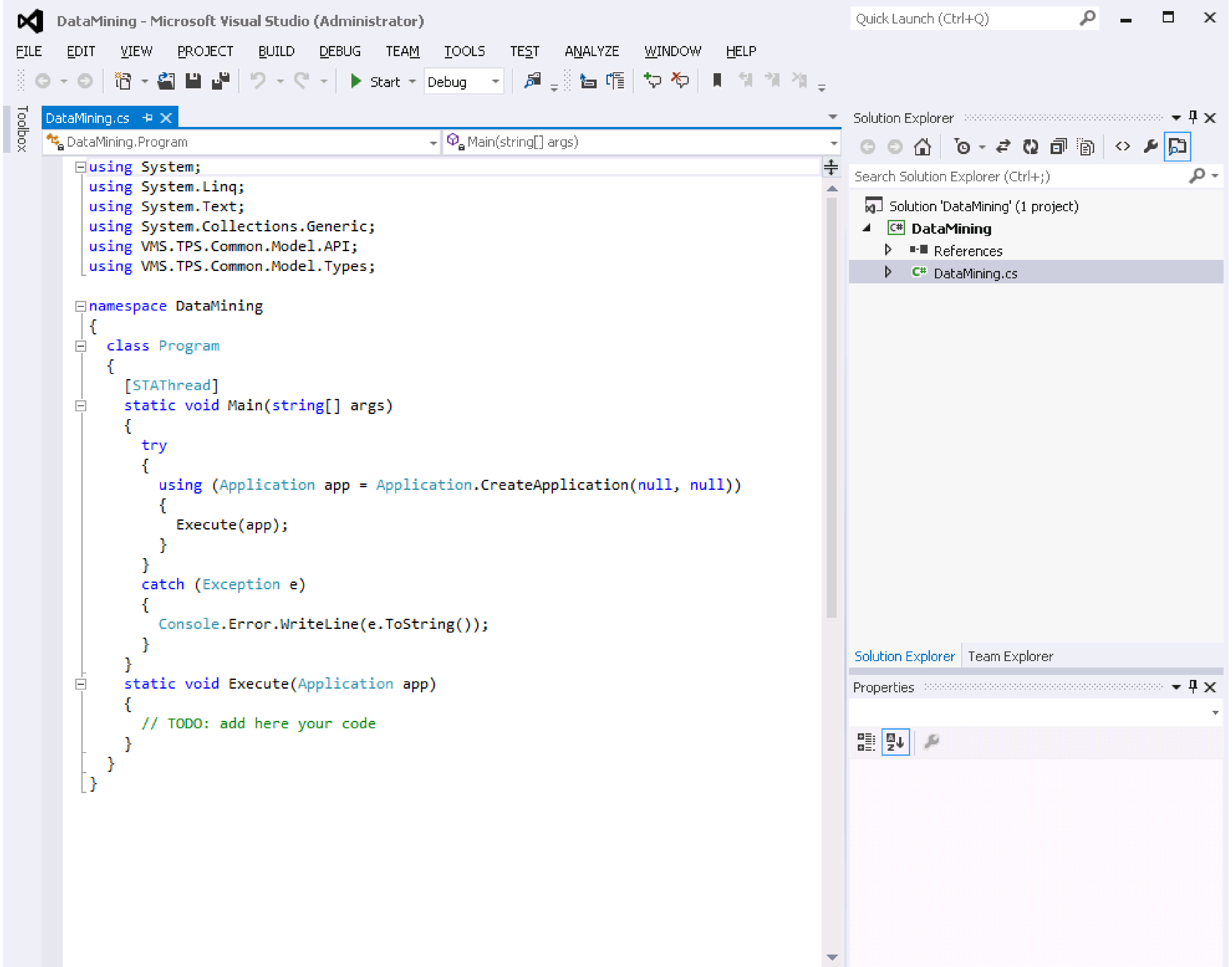
  (Standalone Executable - “An Application”.

  Examples: Microsoft Word, Excel)

Exercise 2 – Standalone Executable

- Step 1:

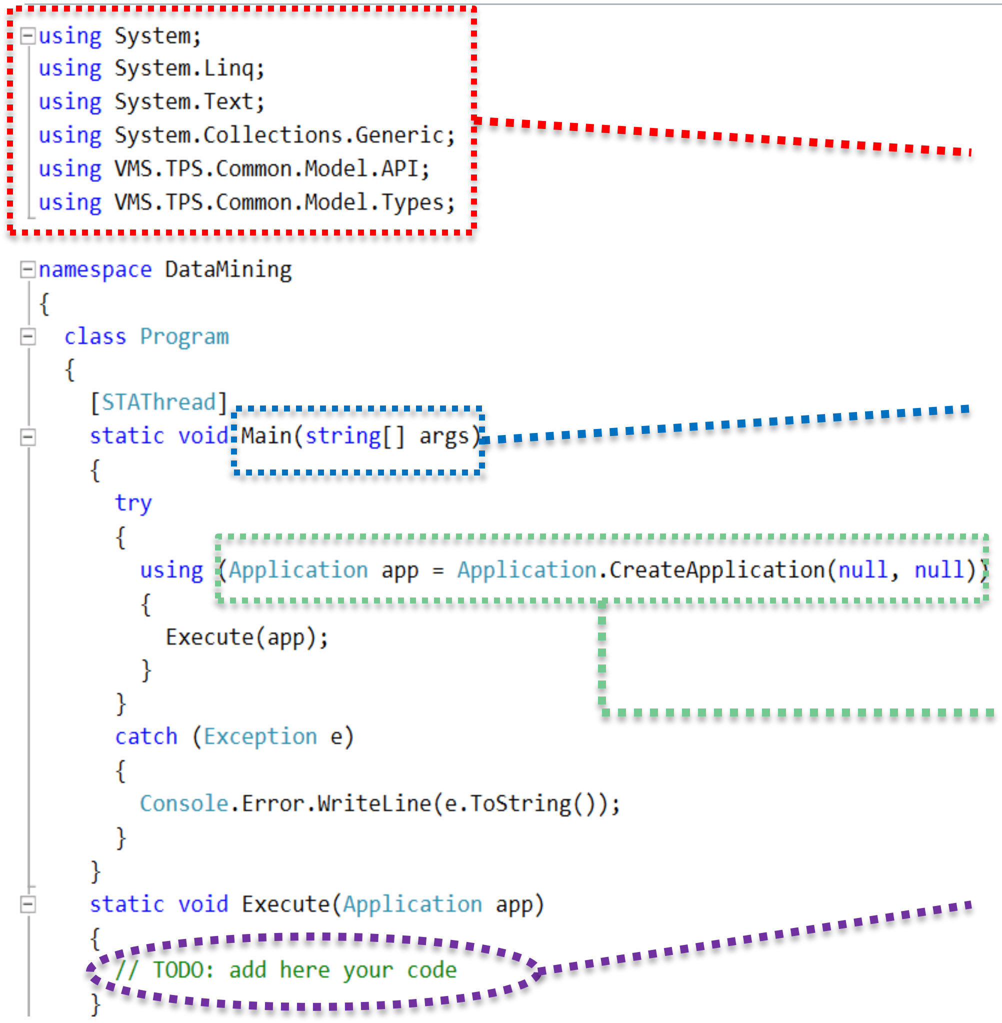
1) Run Eclipse Script Wizard.
2) Create a Standalone Executable Script and name it “DataMining”,
3) Open project in Visual Studio.
4) Open file “DataMining.cs”.


DataMining - Microsoft Visual Studio (Administrator)
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Start
Debug
DataMining.cs
DataMining.Program
Main(string[] args)
Toolbox
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'DataMining' (1 project)
DataMining
References
DataMining.cs
Solution Explorer
Team Explorer
Properties

```
using System;
using System.Linq;
using System.Text;
using System.Collections.Generic;
using VMS.TPS.Common.Model.API;
using VMS.TPS.Common.Model.Types;

namespace DataMining
{
  class Program
  {
    [STAThread]
    static void Main(string[] args)
    {
      try
      {
        using (Application app = Application.CreateApplication(null, null))
        {
          Execute(app);
        }
      }
      catch (Exception e)
      {
        Console.Error.WriteLine(e.ToString());
      }
    }
    static void Execute(Application app)
    {
      // TODO: add here your code
    }
  }
}
```

Standalone Exe - C# Syntax Notes



```
using System;
using System.Linq;
using System.Text;
using System.Collections.Generic;
using VMS.TPS.Common.Model.API;
using VMS.TPS.Common.Model.Types;

namespace DataMining
{
    class Program
    {
        [STAThread]
        static void Main(string[] args)
        {
            try
            {
                using (Application app = Application.CreateApplication(null, null))
                {
                    Execute(app);
                }
            }
            catch (Exception e)
            {
                Console.Error.WriteLine(e.ToString());
            }
        }
        static void Execute(Application app)
        {
            // TODO: add here your code
        }
```

C# imports - similar to C++ ‘#include’, java & python ‘import’.

Standalone Executable loads the Eclipse runtime. Starts with a Main (same as in C – main routine).

Create connection to Eclipse, login (null causes login prompt).

The real code starts here.

Standalone Executable Context

- For Plugin Script, Eclipse passes the context to script through variable ScriptContext.

- For Standalone Executable script, the script must establish its own context.

Standalone Executable Context

Loop over all patients in the database to find matching patient:

```
static void Execute(Application app)
    {
        // Loop over patients, load each one
        foreach (PatientSummary ps in app.PatientSummaries)
        {
            if (ps.Id == "exercise1")
            {
                Patient patient = app.OpenPatient(ps);
                app.ClosePatient(); // one open at a time.
            }
        }
```

Standalone Executable Context

Load a specific patient by ID:

```
static void Execute(Application app)
    {
        // load patient by ID
        Patient p = app.OpenPatientById("exercise1");
        if (p != null)
        {
            app.ClosePatient(); // one open at a time.
        }
```

VARIAN medical systems

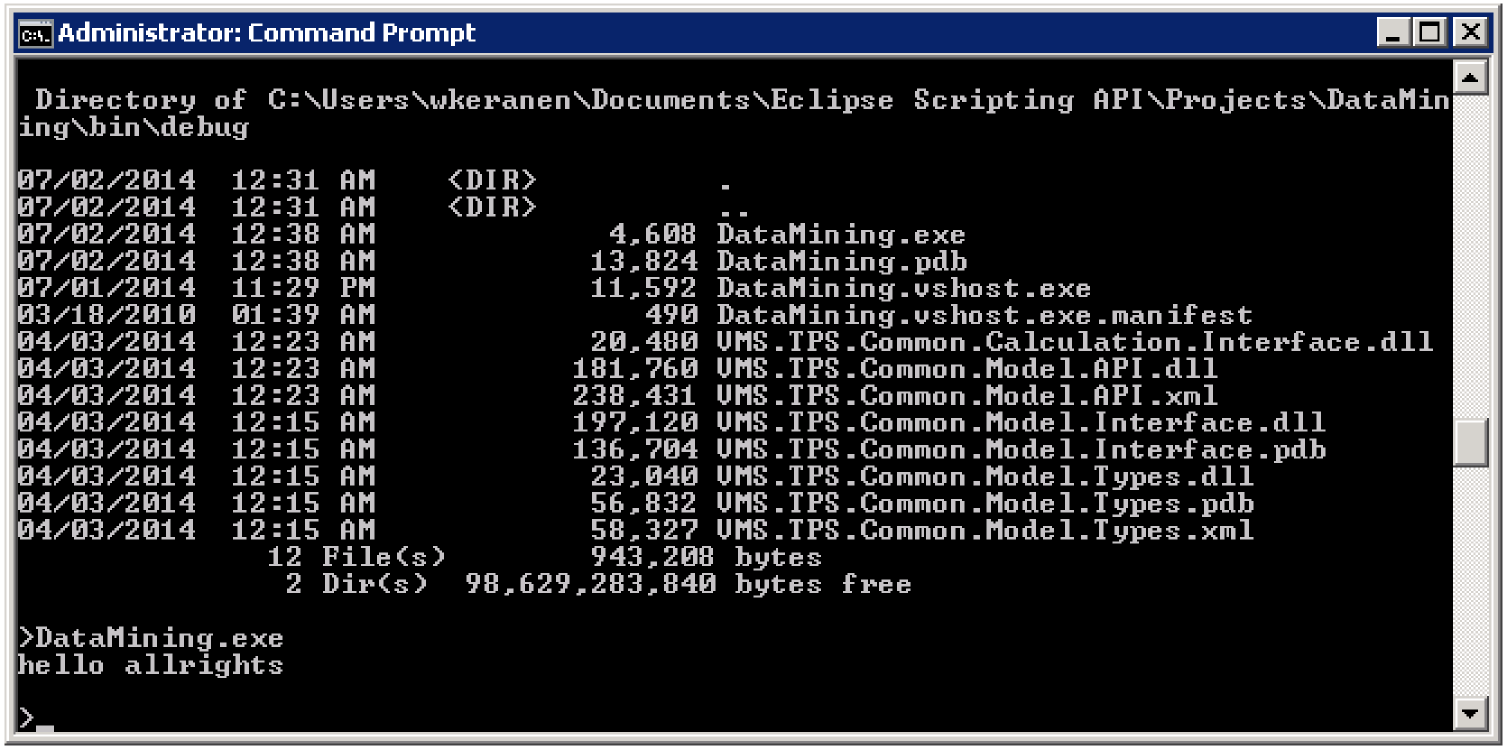
Standalone Executable Context

- Patient.Courses is a list (IEnumerable<T>)
- Loop over Courses, Course.PlanSetups to find the right context.

```
Patient patient = app.OpenPatient(ps);
   foreach (Course c in patient.Courses)
   {
       foreach (PlanSetup p in c.PlanSetups)
        {
         }
```

Printing to screen in console app

```
static void Execute(Application app)
    {
        Console.WriteLine("hello " + app.CurrentUser.Id);
    }
```



Administrator: Command Prompt

```
 Directory of C:\Users\wkeranen\Documents\Eclipse Scripting API\Projects\DataMin
ing\bin\debug

07/02/2014  12:31 AM    <DIR>          .
07/02/2014  12:31 AM    <DIR>          ..
07/02/2014  12:38 AM             4,608 DataMining.exe
07/02/2014  12:38 AM            13,824 DataMining.pdb
07/01/2014  11:29 PM            11,592 DataMining.vshost.exe
03/18/2010  01:39 AM               490 DataMining.vshost.exe.manifest
04/03/2014  12:23 AM            20,480 VMS.TPS.Common.Calculation.Interface.dll
04/03/2014  12:23 AM           181,760 VMS.TPS.Common.Model.API.dll
04/03/2014  12:23 AM           238,431 VMS.TPS.Common.Model.API.xml
04/03/2014  12:15 AM           197,120 VMS.TPS.Common.Model.Interface.dll
04/03/2014  12:15 AM           136,704 VMS.TPS.Common.Model.Interface.pdb
04/03/2014  12:15 AM            23,040 VMS.TPS.Common.Model.Types.dll
04/03/2014  12:15 AM            56,832 VMS.TPS.Common.Model.Types.pdb
04/03/2014  12:15 AM            58,327 VMS.TPS.Common.Model.Types.xml
              12 File(s)        943,208 bytes
               2 Dir(s)  98,629,283,840 bytes free

>DataMining.exe
hello allrights

>_
```

Exercise 2, Step 2.

- Step 2: Create a loop or series of loops that:

1) Find all plans that are in a course called "Varian".

2) Print out the "patient/course/plan ids" and Plan Approval Status for each found plan.

Exercise 2, Step 3.

- Step 3:

Add code to print the max dose for each plan that has dose calculated.

Bonus points if you can print the max dose in absolute dose! ☺

Exercise 2, Step 4.

- Step 4:

Add code to print the beam ids of each plan and meterset weights of each control point in each beam.

Print the additions in the same loop.

Well done, scripter!