Introduction to ASP.NET vNext

Bangkok .NET Users Group

Jetabroad (Thailand) Offices February 3rd 2015

Overview

Ace

Runtime and .Net Framework

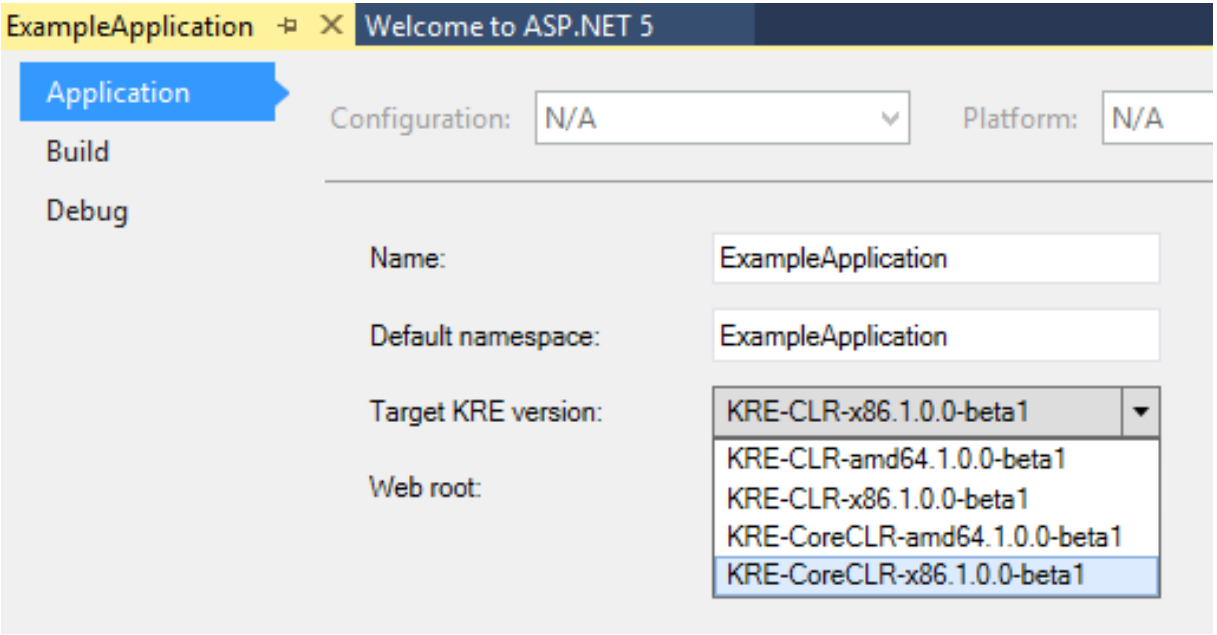
- Ability to deploy the framework along with the application
- Upgrading framework can be done per app

Runtime and Framework

Being able to run on three runtimes:

- Core CLR (Cloud-Optimized Runtime)
- Full .Net CLR
- Cross-Platform CLR

Runtime and .Net Framework



Runtime and .Net Framework

- kre (k runtime engine)
  - Bootstrap and run an application
- kvm (k version manager)
  - To install/upgrade/get/switch kre(s) on machines.

```
PS SQLSERVER:\> kvm list

Active Version     Runtime Architecture Location                    Alias
------ -------     ------- ------------ --------                    -----
       1.0.0-beta1 CLR     amd64        C:\Users\Ace\.kre\packages
       1.0.0-beta1 CLR     x86          C:\Users\Ace\.kre\packages
       1.0.0-beta1 CoreCLR amd64        C:\Users\Ace\.kre\packages
       1.0.0-beta1 CoreCLR x86          C:\Users\Ace\.kre\packages
       1.0.0-beta2 CLR     amd64        C:\Users\Ace\.kre\packages
  *    1.0.0-beta2 CLR     x86          C:\Users\Ace\.kre\packages default
       1.0.0-beta2 CoreCLR amd64        C:\Users\Ace\.kre\packages
       1.0.0-beta2 CoreCLR x86          C:\Users\Ace\.kre\packages
```

Runtime and .Net Framework

- System.Web assembly is monolithic.
- No matter how small changes ,means shipping a new version of the entire assembly.
- Removes the dependency on monolithic framework assemblies.

Runtime and .Net Framework

- System.Web functionality has been moved into smaller pieces of NuGet packages.
- Upgrading dependencies can be done for specific packages.

Runtime and .Net Framework

```
"dependencies": {
    "EntityFramework.SqlServer": "7.0.0-beta1",
    "EntityFramework.Commands": "7.0.0-beta1",
    "Microsoft.AspNet.Mvc": "6.0.0-beta1",
    //"Microsoft.AspNet.Mvc.WebApiCompatShim": "6.0.0-beta1",
    "Microsoft.AspNet.Diagnostics": "1.0.0-beta1",
    "Microsoft.AspNet.Diagnostics.Entity": "7.0.0-beta1",
    "Microsoft.AspNet.Identity.EntityFramework": "3.0.0-beta1",
    "Microsoft.AspNet.Security.Cookies": "1.0.0-beta1",
    "Microsoft.AspNet.Server.IIS": "1.0.0-beta1",
    "Microsoft.AspNet.Server.WebListener": "1.0.0-beta1",
    "Microsoft.AspNet.StaticFiles": "1.0.0-beta1",
    "Microsoft.Framework.ConfigurationModel.Json": "1.0.0-beta1",
    "Microsoft.Framework.CodeGenerators.Mvc": "1.0.0-beta1",
    "Microsoft.Framework.Logging": "1.0.0-beta1",
    "Microsoft.Framework.Logging.Console": "1.0.0-beta1",
    "Microsoft.VisualStudio.Web.BrowserLink.Loader": "14.0.0-beta1"
},
```

Runtime and .Net Framework

```
"Microsoft.VisualStudio.Web.BrowserLink.Loader": "14.0.0-beta1",
"Mi"
```

Search remote NuGet packages for 'Mi'

- EntityFramework.Migrations
- Microsoft.AspNet.Diagnostics
- Microsoft.AspNet.Diagnostics.Entity
- Microsoft.AspNet.FeatureModel
- Microsoft.AspNet.FileSystems
- Microsoft.AspNet.Hosting
- Microsoft.AspNet.Http
- Microsoft.AspNet.Http.Extensions
- Microsoft.AspNet.HttpFeature

Runtime and .Net Framework

- kpm (k package manager)
  - Responsible for all operation involved with packages in an application

```
PS C:\Windows\system32> d:
PS D:\> kpm pack -o d:\websites\dfs --overwrite --no-source --runtime kre-clr-x86.1.0.0-beta2 --configuration Debug
```

Hosting

- Designed as cross platform and host agnostic.
- No longer dependent on a system installation of .NET.
- More control over http pipelining. Better performance.

Hosting

- IIS

```
"dependencies": {
        "Microsoft.AspNet.Server.IIS": "1.0.0-alpha4"
    }
```

- Self-Hosting

```
"dependencies": {
        "Microsoft.AspNet.Server.WebListener": "1.0.0-alpha4"
            }
```

Hosting

- Http Pipelining

```
using System;
using Microsoft.AspNet.Builder;
using Microsoft.Framework.DependencyInjection;

namespace MvcApplication
{
    public class Startup
    {
        public void Configure(IApplicationBuilder app)
        {
            app.UseServices(services => { services.AddMvc(); });
            app.UseMvc();
        }
    }
}
```

Hosting

- k
  - k build: Build an application
  - k run: Run an application
  - k web: Host and Run a web application

```
PS C:\Windows\system32> d:
PS D:\> cd D:\DevZone\DFS\Jetabroad.Dfs\src\Jetabroad.Dfs.Control
PS D:\DevZone\DFS\Jetabroad.Dfs\src\Jetabroad.Dfs.Control> k web
Started
```

Dependency Injection

- Built-in DI for configuring services/libraries.
- Supports BYOC (Bring Your Own Container).

Demo

Chaow

Front-End vNext

Working with JavaScript and CSS

Will

Old style: Microsoft-Only Tools

Hosting

NuGet

Transformation

ASP.NET Bundling

New style: Standard JavaScript Tools

Hosting

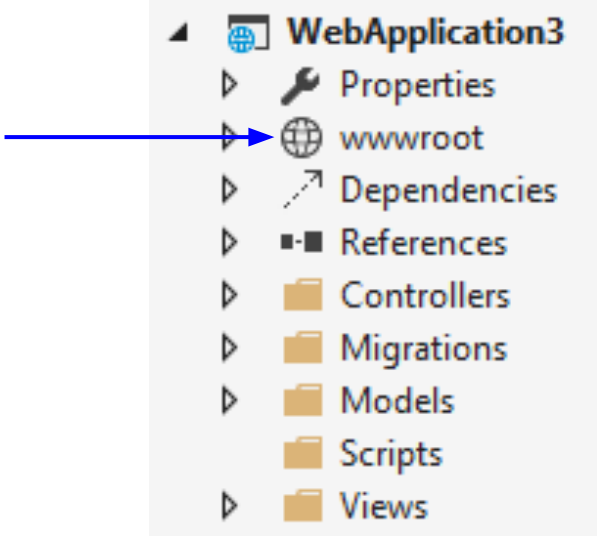
Bower

Transformation

Grunt

New Directory: wwwroot

- Anything in `wwwroot` will be web-accessible.
- Libraries are put **inside** `wwwroot`.
- Custom JS and CSS are put **outside** `wwwroot`
  - Only the minified versions will go inside `wwwroot`



Using Bower

- List library in bower.json
- Run package restore
  - Packages are downloaded to `wwwroot`

Bower: Front-End Package Manager

bower.json*

http://json.schemastore.org/bower

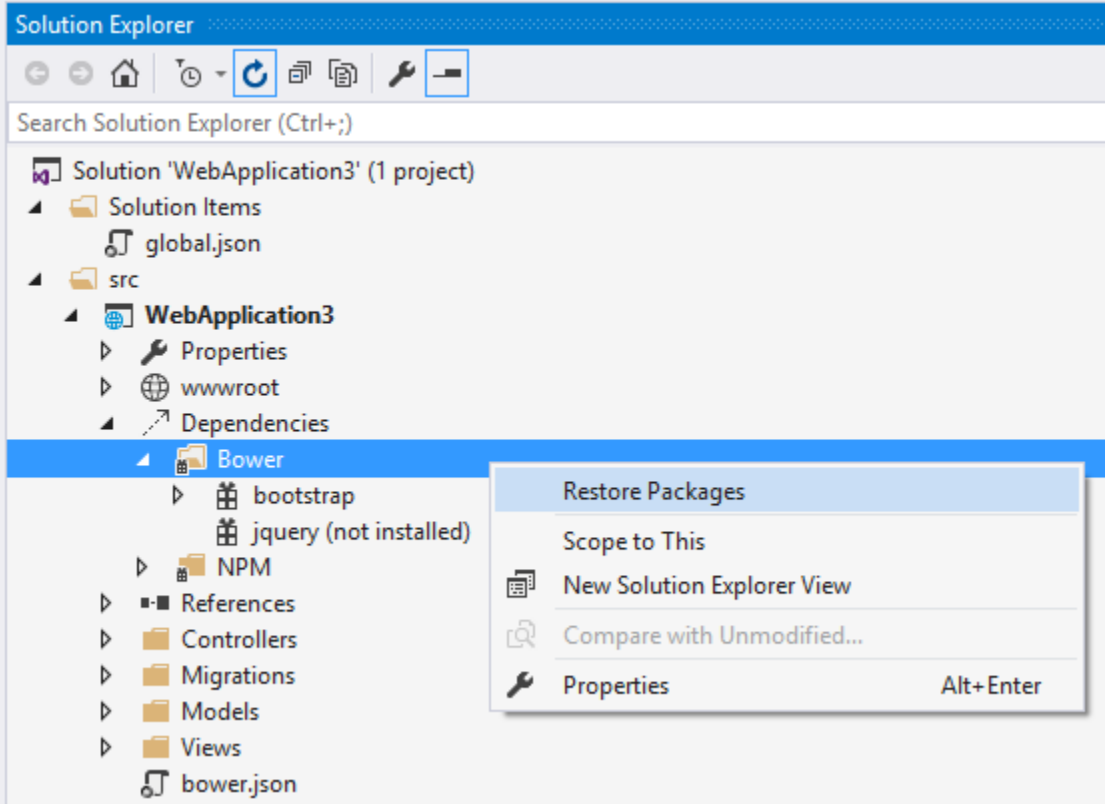
```
{
    "name": "WebApplication3",
    "private": true,
    "dependencies": {
        "bootstrap": "~3.0.0",
        "jquery": "2.1.
    },
    "exportsOverri
        "bootstrap
            "js":
```

2.1.3 — The currently latest version of the package

^2.1.3

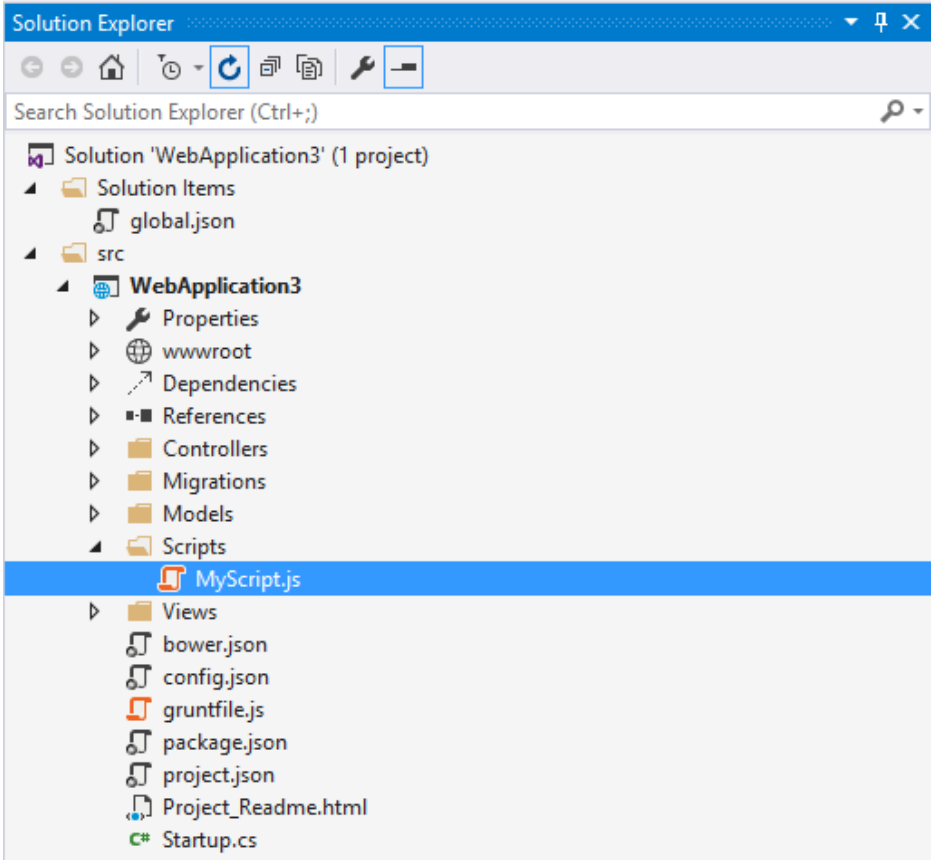
~2.1.3

Bower: Front-End Package Manager



Using Grunt

1. List task in in `package.json`
2. Run package restore
3. Configure task in `gruntfile.js`
4. Run task via Task Runner Explorer


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'WebApplication3' (1 project)
Solution Items
global.json
src
WebApplication3
Properties
wwwroot
Dependencies
References
Controllers
Migrations
Models
Scripts
MyScript.js
Views
bower.json
config.json
gruntfile.js
package.json
project.json
Project_Readme.html
Startup.cs

package.json gruntfile.js

http://json.schemastore.org/package

```
{
    "version": "0.0.0",
    "name": "WebApplication3",
    "devDependencies": {
        "grunt": "^0.4.5",
        "grunt-bower-task": "^0.4.0",
        "grunt-contrib-uglify": "^0.7.0"
    }
}
```

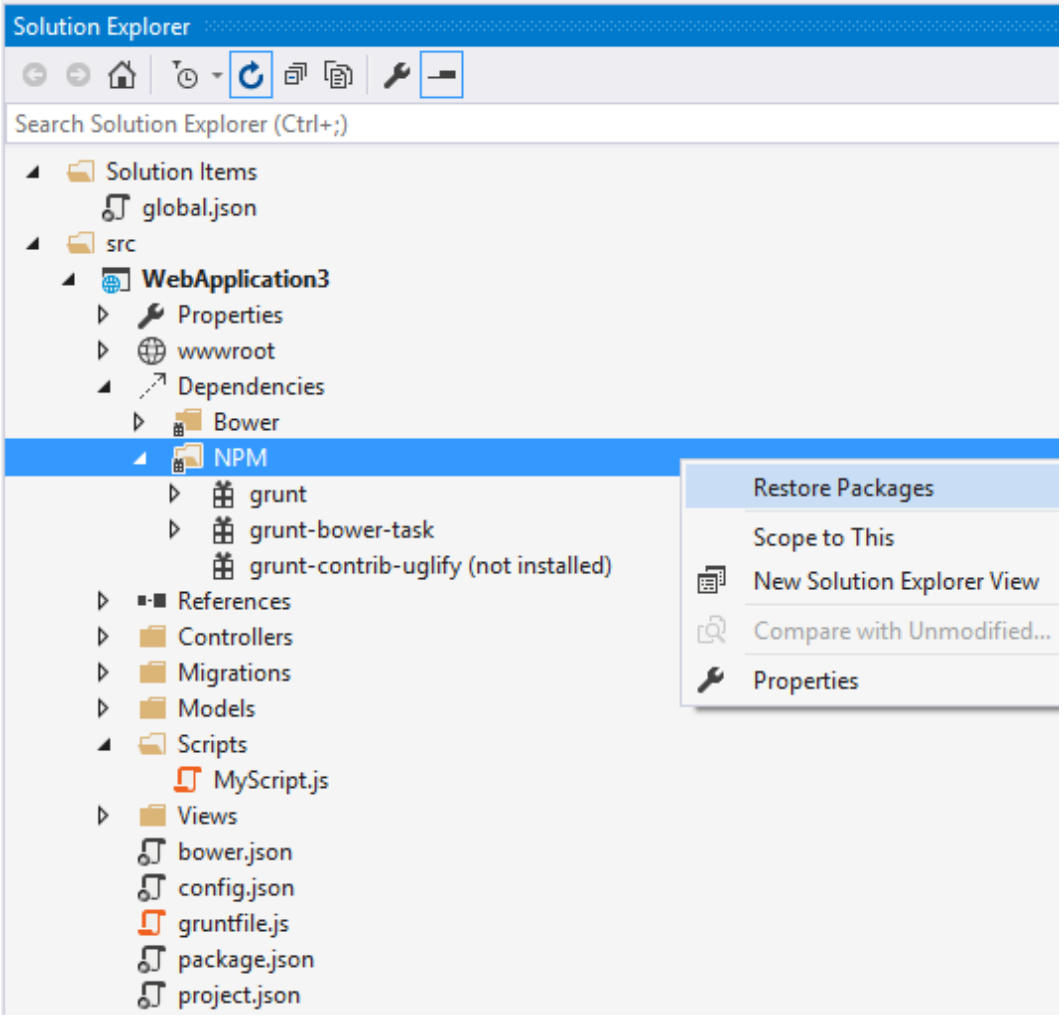
grunt-contrib-uglify

Type: string

grunt-contrib-uglify

Minify files with UglifyJS.

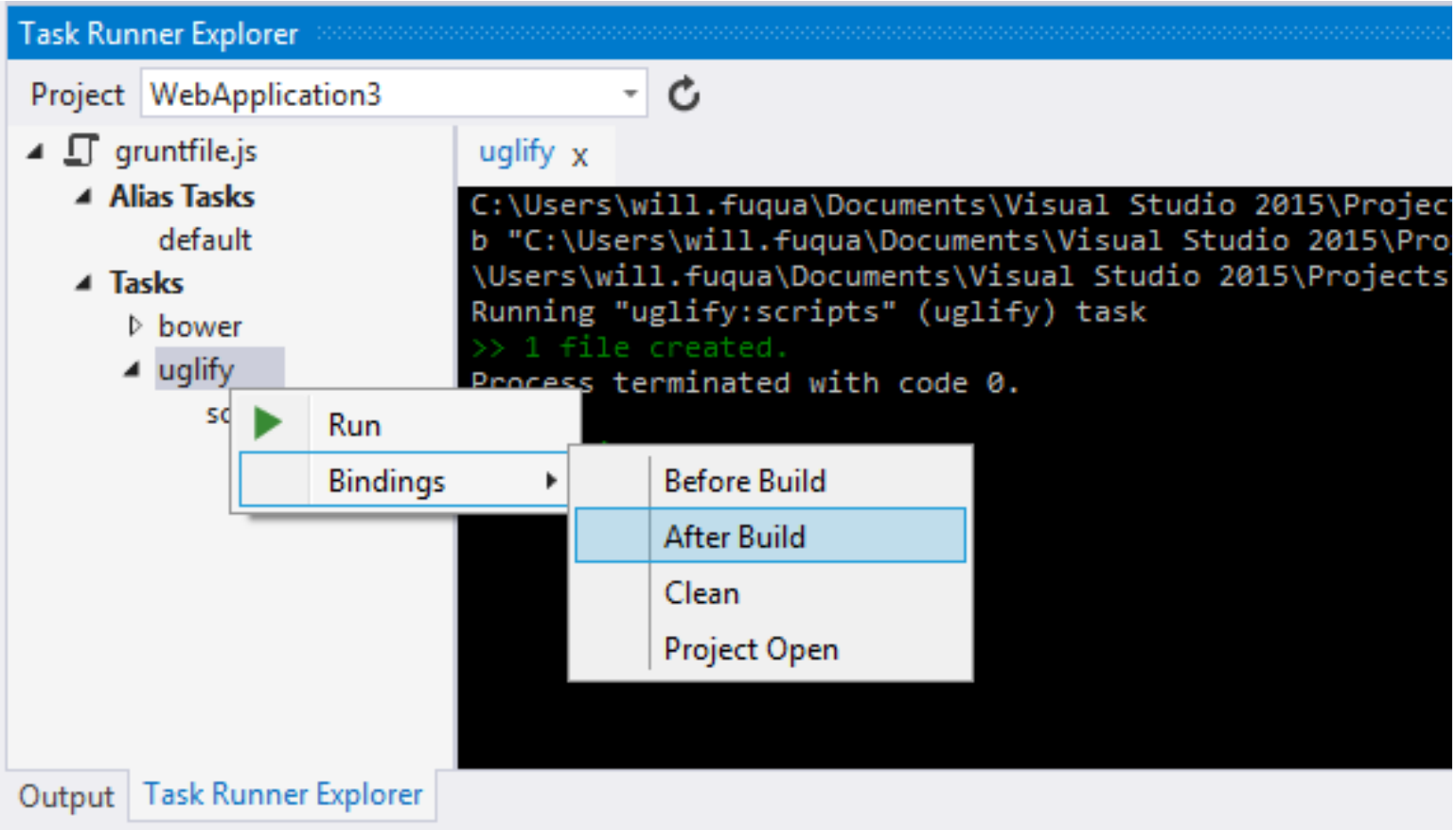
Latest: 0.7.0
Author: Grunt Team
License: n/a
Homepage: https://github.com/gruntjs/grunt-contrib-uglify


Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution Items
global.json
src
WebApplication3
Properties
wwwroot
Dependencies
Bower
NPM
grunt
grunt-bower-task
grunt-contrib-uglify (not installed)
References
Controllers
Migrations
Models
Scripts
MyScript.js
Views
bower.json
config.json
gruntfile.js
package.json
project.json
Restore Packages
Scope to This
New Solution Explorer View
Compare with Unmodified...
Properties

package.json | gruntfile.js

<global>

```
module.exports = function (grunt) {
    grunt.initConfig({
        bower: {
            install: {
                options: {
                    targetDir: "wwwroot/lib",
                    layout: "byComponent",
                    cleanTargetDir: false
                }
            }
        },
        uglify: {
            scripts: {
                files: {
                    'wwwroot/dest/output.min.js': ['Scripts/*']
                }
            }
        }
    });

    grunt.registerTask("default", ["bower:install"]);

    // The following line loads the grunt plugins.
    // This line needs to be at the end of this this file.
    grunt.loadNpmTasks("grunt-bower-task");
    grunt.loadNpmTasks('grunt-contrib-uglify');
};
```


Task Runner Explorer
Project
WebApplication3
gruntfile.js
Alias Tasks
default
Tasks
bower
uglify
Run
Bindings
Before Build
After Build
Clean
Project Open
uglify x
C:\Users\will.fuqua\Documents\Visual Studio 2015\Projec
b "C:\Users\will.fuqua\Documents\Visual Studio 2015\Pro
\Users\will.fuqua\Documents\Visual Studio 2015\Projects
Running "uglify:scripts" (uglify) task
>> 1 file created.
Process terminated with code 0.
Output
Task Runner Explorer

Great, but how do I debug?

Source Maps!

Supported in all browsers

(even IE!)

```
uglify: {
    options: {
        sourceMap: true
    },
    scripts: {
        files: {
            'wwwroot/dest/output.min.js': ['Scripts/*']
        }
    }
}
```

Sources | Content scripts | Snippets

(index) | output.min.js × | MyScript.js

- (no domain)
- localhost:32286
  - Scripts

```
function sayHello(a){alert("Hello "+a)}sayHello("vNext");
//# sourceMappingURL=output.min.js.map
```

Watch Expressions

Call Stack — Async

Not Paused

Sources | Content scripts | Snippets

(index) | output.min.js | MyScript.js ×

- localhost:32286
  - Scripts
  - css
  - dest
  - lib
  - (index)
- localhost:35310

```
function sayHello(name) {
    alert("Hello " + name);
}
sayHello("vNext");
```

Watch Expressions

Call Stack — Async

sayHello	MyScript.js:2
(anonymous function)	MyScript.js:5

Paused on a JavaScript breakpoint.

Scope Variables

Local

```
a: "vNext"
this: Window
```

Global — Window

Breakpoints

MyScript.js:2

```
alert("Hello " + name);
```

DOM Breakpoints

XHR Breakpoints

Event Listener Breakpoints

Bower: Package manager for client-side libraries

Grunt: Task runner for the client-side

Bower: Over 25,000 libraries

Grunt: Over 4,000 tasks

C# 6.0

The New Features

Dew

Member declaration features

- Initializers for auto-properties

```
public class Customer {
    public string First { get; set; } = "Jane";
    public string Last { get; set; } = "Doe";
}
```

- Getter-only auto properties

```
public class Customer {
    public string First { get; } = "Jane";
    public string Last { get; }
    public Customer(string last) { Last = last; }
}
```

Member declaration features (cont)

```
public class Customer {
        private string first = “Jane”;
        private string last = “Doe”;
        public string First {
                get { return this.first; }
                set { this.first = value; }
         };
                public string Last {
                get { return this.last; }
                set { this.last = value; }
         }
}
```

```
public class Customer {
        public string First { get; set; } = "Jane";
        public string Last { get; set; } = "Doe";
}
```

Expression bodied function members

- Expression bodies on method-like members

```
public Point Move(int dx, int dy) => new Point(x + dx, y + dy);
public static Complex operator +(Complex a, Complex b) => a.Add(b);
public static implicit operator string(Person p) => p.First + " " + p.Last;
public void Print() => Console.WriteLine(First + " " + Last);
```

- Expression bodies on property-like function members

```
public string Name => First + " " + Last;
public Customer this[long id] => store.LookupCustomer(id);
```

Expression bodied function members (cont)

```
public Point Move(int dx, int dy) {
        return new Point(x + dx, y + dy);
}
```

```
public Point Move(int dx, int dy) => new Point(x + dx, y + dy);
```

```
public static Complex operator +(Complex a, Complex b) {
        a.Add(b);
}
```

```
public static Complex operator +(Complex a, Complex b) => a.Add(b);
```

```
public static implicit operator string(Person p) {
        return p.First + " " + p.Last;
}
```

```
public static implicit operator string(Person p) => p.First + " " + p.Last;
```

```
public void Print() {
        Console.WriteLine(First + " " + Last);
}
```

```
public void Print() => Console.WriteLine(First + " " + Last);
```

Expression bodied function members (cont)

```
public string Name { get { return First + " " + Last; } }

public Customer this[long id] {
        return store.LookupCustomer(id);
}
```

```
public string Name => First + " " + Last;

public Customer this[long id] => store.LookupCustomer
(id);
```

Import features

- No more

```
Console.WriteLine("Hello World");
Math.Sqrt(9);
```

- Simpler way

```
using static System.Console;
using static System.Math;

WriteLine("Hello World");
Sqrt(9);
```

Parameterless constructors in structs

- C# 5.0 default behavior
- Sample

```
struct Point
{
    public int XCoordinate { get; set; }
    public int YCoordinate { get; set; }
    public Point(int x, int y) : this()
    {
        XCoordinate = x;
        YCoordinate = y;
    }
}
```

Usage:

```
var p1 = new Point(150, 240);
var p2 = new Point(150, 240);

// print 150
Console.WriteLine(p1.XCoordinate);
// print 240
Console.WriteLine(p1.XCoordinate);
```

Parameterless constructors in structs

```
struct Point
{
    public int XCoordinate { get; set; }
    public int YCoordinate { get; set; }
    public Point(int x, int y) : this()
    {
        XCoordinate = x;
        YCoordinate = y;
    }
}
```

```
struct Point
{
    public int XCoordinate { get; set; }
    public int YCoordinate { get; set; }
    public Point() : this(100, 50) { }
    public Point(int x, int y) {
        XCoordinate = x;
        YCoordinate = y;
    }
}
var p1 = new Point();
// Print 100
WriteLine(p1.XCoordinate);
```

Null-conditional operators

- Checking for null has never been easier

```
int? length = customers?.Length;                // null if customers is null
int? orders = customers?[0].Orders.Count;       // null if customers is null
```

- Taking a step further

```
int length = customers?.Length ?? 0;
if (predicate?.Invoke(e) ?? false) { ... }
PropertyChanged?.Invoke(this, args);
```

Null-conditional operators (cont)

```
int? length = customers == null ? (int?) null : customers.Length;
int? orders = customers == null
                        ? (int?) null
                        : customers[0].Orders == null
                                ? (int?) null
                                : customers[0].Orders.Count;

int? length = customers?.Length;
int? orders = customers?[0].Orders?.Count;

int length = customers?.Length ?? 0;
if (predicate?.Invoke(e) ?? false) { ... }
PropertyChanged?.Invoke(this, args);
```

String interpolation

- No more

```
var greeting = string.Format("Hello {0} Age: {1}", user.Name, user.Age.ToString("D3");
Console.WriteLine("GUID: {0}", Guid.NewGuid().ToString("N"));
```

- Simpler way

```
var greeting = $"Hello {user.Name} Age: {user.Age:D3}";
WriteLine($"Guid: {Guid.NewGuid() :N}");
```

nameof expressions

- Ordinary string literals for this purpose is error prone
- Use nameof expression

```
if (user.Username == null) throw new ArgumentNullException(nameof(user.Username));
WriteLine(nameof(person.Address.ZipCode)); // prints "ZipCode"
```

nameof expressions (cont)

```
if (username == null) throw new
ArgumentNullException(“Useraname”);
```

```
(if username == null) throw new
ArgumentNullException(nameof(username));
```

```
var s = string.Format("Name: {0}, Value: {1}",
“Useraname", c.Username);
```

```
var s = $"Name: {nameof(c.Username)}, Value:
{c.Username}";
```

Index Initializers (cont)

```
var jsonObject = new Dictionary<string, string> {
        {"Source", "Jetabroad Thailand"},
        {"Token", Guid.NewGuid().ToString()},
};
```

```
var jsonObject = new Dictionary<string, string> {
        ["Source"] = "Jetabroad Thailand",
        ["Token"] = Guid.NewGuid(),
};
```

Exception filters

- VB has them, F# has them
- C# has them too

```
try { ... }
catch (MyException e) if (myfilter(e)) {
        ...
}
```

Usage:

```
private static bool Log(Exception e) {
        /*** log it by any logger ***/
        return false;
}

try {
        …
}
catch (Exception e) if (Log(e)) {
        ...
}
```

Await in catch and finally blocks

- In C# 5.0 await keyword was not allowed in catch and finally blocks
- The code looks like:

```
Resource res = null;
try {
    res = await Resource.OpenAsync(...);        // You could do this.
} catch(ResourceException e) {
    await Resource.LogAsync(res, e);            // Now you can do this ...
} finally {
    if (res != null) await res.CloseAsync();    // ... and this.
}
```

ASP.NET vNext build with TeamCity and Octopus Deployment.

O

Agenda

- Teamcity
  - Prerequisites
  - Continuous Integration
  - Deployment
- Octopus Deployment

Teamcity - Prerequisites

- Create an user account to run a Teamcity Build Agent.
- Change the TeamCity Build Agent service user account.
- Install a KVM and KRE into service user account.
- Install node.js, npm, grunt-cli and Git for Windows.

TeamCity – Continuous Integration

- Use `KPM RESTORE` command to restore packages.
- Use `KPM BUILD` command to build an application.
- Use `K TEST` command to run the unit test.

KPM RESTORE commar

```
[root]
        [src]
                [src_folder1]
                        project.json
                [src_folder2]
                        project.json
                        [src_folder2_1]
                                project.json
                        [src_folder2_2]
                                project.json
        [test]
                [test_folder1]
                        project.json
                        [test_folder1_1]
                                project.json
        global.json
```

```
@echo off
cd %teamcity.build.workingDir%

SETLOCAL
CALL kvm use default -runtime CLR -amd64

CALL kvm list
```

```
@powershell -NoProfile -ExecutionPolicy unrestricted -Command "Get-ChildItem %mr.
SourceFolder% global.json -rec -erroraction 'silentlycontinue'  | Select-Object -Expand
DirectoryName | Foreach { cmd /C cd $_ `&`& CALL kpm restore }; exit $Lastexitcode"
```

KPM BUILD Command

```
@echo off
cd %teamcity.build.workingDir%

SETLOCAL

CALL kvm use default -runtime CLR -amd64

CALL kvm list

@powershell -NoProfile -ExecutionPolicy unrestricted -Command "Get-ChildItem %mr.
SourceFolder% project.json -rec -erroraction 'silentlycontinue' | Foreach { kpm build
$_.FullName --configuration %mr.Configuration% }; exit $Lastexitcode"
```

KPM BUILD Command (Cont…)

```
[root]
        [src]
                [src_folder1]
                        project.json
                [src_folder2]
                        project.json
                        [src_folder2_1]
                                project.json
                        [src_folder2_2]
                                project.json
        [test]
                [test_folder1]
                        project.json
                        [test_folder1_1]
                                project.json
        global.json
```

- c:\[root]\[src]\[src_folder1]
- c:\[root]\[src]\[src_folder2]
- c:\[root]\[src]\[src_folder2]\[src_folder2_1]
- c:\[root]\[src]\[src_folder2]\[src_folder2_2]
- c:\[root]\[test]\[test_folder1]
- c:\[root]\[test]\[test_folder1]\[test_folder1_1]

K TEST Command

```
@echo off
cd %teamcity.build.workingDir%

SETLOCAL
CALL kvm use default -runtime CLR -amd64

CALL kvm list

@powershell -NoProfile -ExecutionPolicy unrestricted -Command "Get-ChildItem %mr.
SourceFolder% project.json -rec -erroraction 'silentlycontinue' | Where-Object { $_.
FullName -like '*test*' } | Select-Object -Expand DirectoryName | Foreach { cmd /C cd $_
`&`& k test -teamcity }; exit $Lastexitcode"
```

K TEST Command (Cont…)

```
[root]
        [src]
                [src_folder1]
                        project.json
                [src_folder2]
                        project.json
                        [src_folder2_1]
                                project.json
                        [src_folder2_2]
                                project.json
        [test]
                [test_folder1]
                        project.json
                        [test_folder1_1]
                                project.json
        global.json
```

- c:\[root]\[test]\[test_folder1]
- c:\[root]\[test]\[test_folder1]\[test_folder1_1]

TeamCity – Deployment

- Use `KPM RESTORE` command to restore packages.
- Use `KPM BUILD` command to build an application.
- Use `KPM PACK` command to create a deployment package.

KPM PACK Command

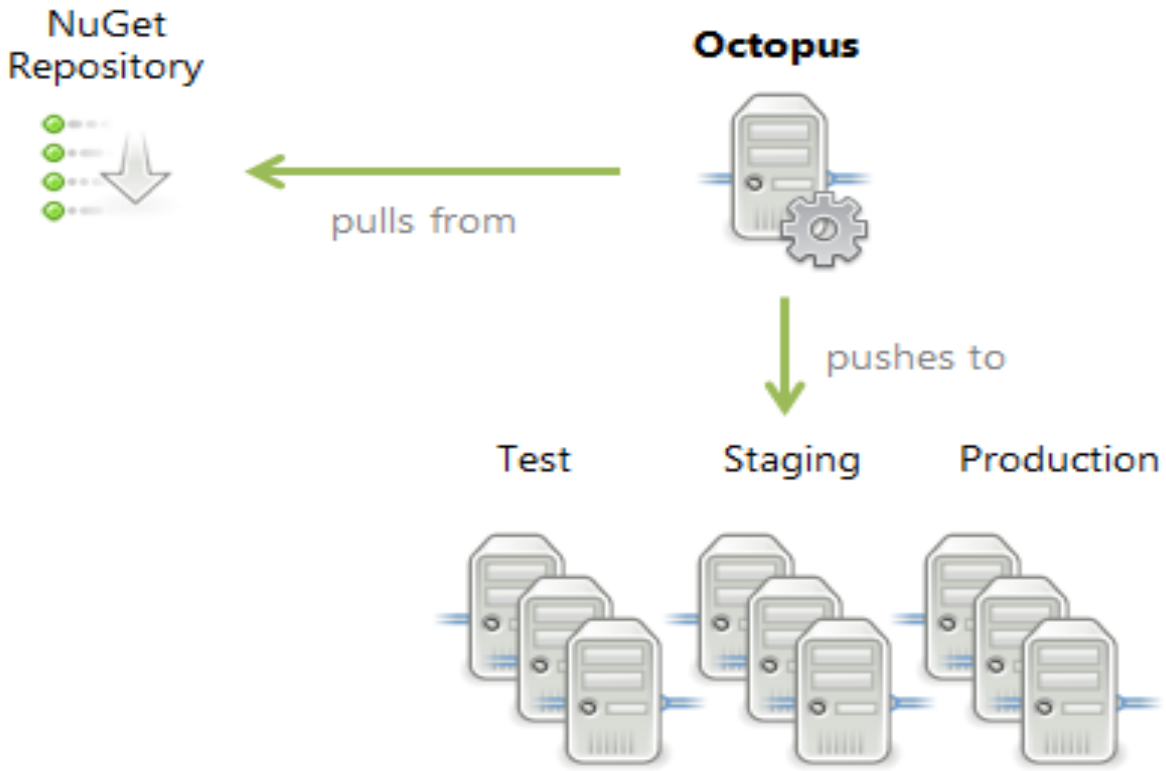
```
@echo off
cd %teamcity.build.workingDir%

SETLOCAL
@powershell -NoProfile -ExecutionPolicy unrestricted -Command "del %teamcity.build.
workingDir%\artifacts\* -Force -Recurse"

CALL kvm use default -runtime CLR -amd64
CALL kvm list
CALL kpm pack "%mr.ProjectJson%" -o artifacts --no-source --overwrite --configuration %
mr.Configuration% --runtime KRE-CLR-amd64.1.0.0-beta1

@echo on
echo "##teamcity[publishArtifacts 'artifacts => %mr.ArtifactsName%']"
```

KPM PACK Command (Cont…)

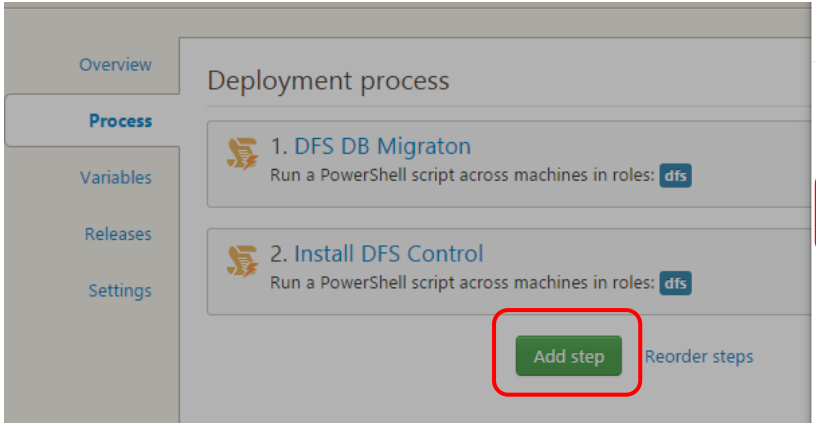
```
[root]
    [src]
        [src_folder1]
            project.json
        [src_folder2]
            project.json
            [src_folder2_1]
                project.json
            [src_folder2_2]
                project.json
    [test]
        [test_folder1]
            project.json
            [test_folder1_1]
                project.json
    global.json
```

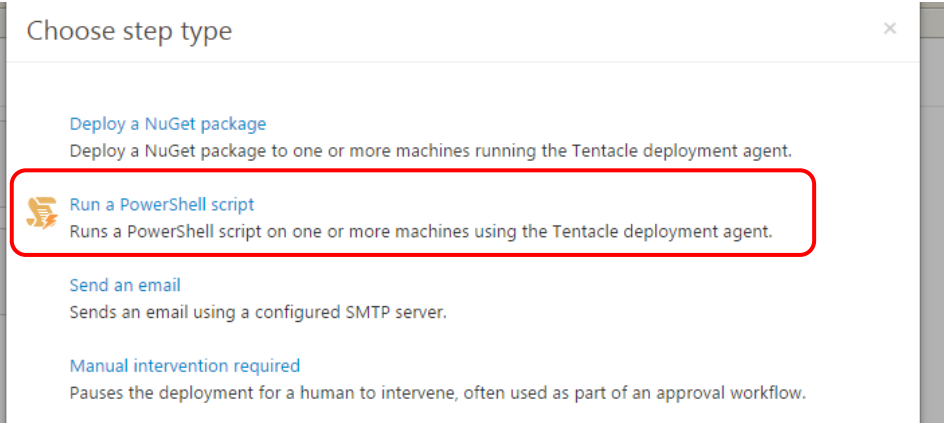
- C:\[root]\[src]\[src_folder1]
- src_folder1_application.zip

Octopus Deployment



Octopus Deployment (Cont…)





Deployment PowerShell Script

- Copy the application deployment zip packages from TeamCity server.

```
Write-Debug "Copying files."

cp -Path "\\TeamCity\Full Build\$teamcityBuildId\Jetabroad.Dfs.DataAccess.*.zip" -Destination
$dfsCorePath
```

- Extract the application deployment zip packages to target folder.

```
Write-Debug "Extract Dtfs.DataAccess to target folder $dfsDbMigrationPath ."

& C:\Programs\7za.exe x $dfsCoreZipFile -o"$dfsDbMigrationPath"
```

Deployment PowerShell Script (Cont…)

- [Optional] Change the application settings in the Config.json file.

```
$configFile = ls "$dfsDbMigrationPath\approot\packages\Jetabroad.Dfs.DataAccess" "config.json" -Recurse -ErrorAction SilentlyContinue | % { $_.FullName }
$configObj = (Get-Content $configFile) -join "`n" | ConvertFrom-Json
Write-Debug "Change the database connection string $dfsDbConnectionString"
$configObj.Data.DefaultConnection.ConnectionString = $dfsDbConnectionString
```

- [Optional] Run EF command to apply any database migration.

```
Write-Debug "Run the database migration command."
cd $dfsDbMigrationPath
.\ef.cmd migration apply -c DfsDbContext
```