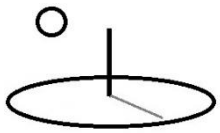

DevTreks Source Code

Last Updated: September 15, 2018 First Released: November 10, 2015

Author: Kevin Boyle, President, DevTreks

Version: 2.1.6

URLs

https://www.devtreks.org

https://github.com/kpboyle1/devtreks2.1 (.NET Core 2.1)

https://devtreks1.blob.core.windows.net/resources/db216.zip

folder holding 2.1.4 uploaded data files that exceeded the 500KB limit for db storage

https://devtreks1.blob.core.windows.net/resources/network_carbon.zip

https://github.com/kpboyle1/devtreksapi1

A. Introduction

This reference documents the source code for DevTreks. The software uses a "plain old html and javascript" front end (**1***) with a web server or cloud server back end. "Plain old" means browsers that support both javascript and html5. Older browsers will not fully work with this product. The underlying html technology targets mobile devices, but also works well on tablets and desktops.

DevTreks is a multitier ASP.NET Core database application. The web project, DevTreks, uses an MVC pattern. The data layer, DevTreks.Data, uses an EF Core data repository pattern. EF data models are stored in the DevTreks.Models project. ASP.NET Identity models are stored in the DevTreks web project's Data folder. Localization strings are stored in the DevTreks.Exceptions and DevTreks.Resources projects. The DevTreks.Extensions folder holds projects that use a

Managed Extensibility Framework pattern. Each project holds a separate group of calculators and analyzers.

The Social Budgeting tutorial explains how to manage networks, clubs, members, and data services. The Calculators and Analyzers tutorial explains how calculators and analyzers work.

B. Kindred Spirits

The spirit employed in building the software is as important as the source code itself. That spirit is contained in the motto at the top of the page. The source code is designed to assist with important societal issues throughout the world –climate change mitigation, food system health, health care effectiveness, natural resources conservation, and public infrastructure investment. Many of these issues, such as climate change or health care, deal with public goods that need to be delivered efficiently if society is to benefit. Others, such as ag production, deal with improving livelihoods in specific economic sectors. The spirit extends to the people who may use this source code. Four kindred spirits that have the needed "gusto" are more desirable than four million conventional users that lack the needed spirit (**2* and 5***).

C. License

Appendix A contains the software license. This version of the MIT License was chosen because of its simplicity and permissiveness. It has pros and cons. On one hand, it allows people to do much of what they want to do. On the proverbial other hand, permissive licenses promote branches of the software to go off in unpredictable directions, similar to Android. The author is familiar with licenses that work well because they require all source code changes be "shared" (i.e. GIS software licenses). Their advantage is that the software goes in one predictable direction. That might be appropriate at a future date, but not yet.

The software is copyrighted by the author because he started the project prior to the formation of the non-profit DevTreks. The project arose from a prototype the author developed while working for the USDA. DevTreks has not accepted, and does not accept, donations (but the recent trend

towards “$28” donations is consistent with this source’s business ethos (**3***)). The only exception is the software development tools.

The current version uses 2 third party, open source, software modules, Jace and MathNet, with licenses that can be found on codeplex and/or github. The code uses standard open source code projects that include JQuery and Bootstrap.

D. ReadMe

Appendix B contains the ReadMe.txt file for the current source code release. Highlights include:

1. This release upgraded to AspNet Core 2.1, fixed bugs found during further 2.1.4 testing, and upgraded the calculator patterns.

E. Development Platforms

The version 2.1.0 refactor was built using Visual Studio 2017, Professional Edition. Version 2.1.6 was deployed using Version 15.8+. The development database uses Sql Server 2017 Express (Version 17.7). The Azure database uses their latest version. The main programming languages include C#, javascript, and T-SQL.

Version 2.1.0 upgraded from the Net Framework 4.6.1 libraries referenced as “net461” in projects to .NET Core 2.0 in the ASP NET web project and .NET Standard 2.0 in project class libraries. The result should be cross platform source that runs on Windows, Linux, and Mac servers. DevTreksStatsApi uses “netcoreapp1.0” and is cross platform. All testing to date has been limited to Windows servers.

F. Deployment Platforms

All releases are deployed on two servers. The first is the localhost web server contained in Windows desktop operating systems. The second is a Microsoft Azure cloud server. The

software has never been deployed to a regular web server, but with minor tweaks, it can be deployed to regular IIS web servers as well. The DevTreksStatsApi WebApi app is deployed on cross platform virtual machines. Version 2.1.0 is the first cross platform release, but requires testing on Linux and Mac servers.

The References include documentation for publishing and deploying to various platforms. **Appendix C** explains how to deploy to a new Azure App Service web app. **Appendix D** explains how deploy on a desktop or web server to localhost:5001.

The software is designed for Internet scale use. The scope of the software extends to multiple economic sectors, including agriculture, building construction, health care, food systems, and climate change. While the software can be deployed to a personal desktop for personal use, serious users of this software must be prepared to work at appropriate scales and scopes. Outside of some international technology hubs and server farms, the author is not familiar with people who have experience caring, thinking, and acting, at the intended scales and scopes.

In fact, the author stopped attending professional economist meetings because he thought that most participants were, and are, "doing it wrong". As a self-assessment exercise in "doing it right", interpret the following question "Other than institutional factors, is there any technical reason that natural resource damage assessments can't be completed online, stored uniformly online, and easily accessed online by people and machines, for every major potential disaster in the world?". After perusing recent DevTreks tutorials, feel free to substitute the terms "Health Technology Assessment" or "Conservation Technology Assessment", for "natural resource damage assessments".

For issues of global importance, such as climate change or efficient health care delivery, professionals and government officials, need to learn how to care, think, and act, at appropriate scales and scopes. That usually requires hiring the right people who know how to work hard "doing it right" (**3***).

G. Containers and Fee-based Subscription Deployments

The sibling reference file in this tutorial, Containers, explains more about subscription-based containers. Version 2.1.6 investigated Containers further and decided not to use them at the current time.

H. Tests and Perfection

The author stopped running formal code tests early in the project cycle because the truly meaningful tests were taking longer to set up than actually writing the source code. The remaining tests definitely returned "green lights", but, given the time requirements for the important tests, didn't impress the author. He decided that, based on his "project team", it would be better to base tests on the tutorials that are prepared for every major software feature. Each tutorial contains URLs demonstrating what a feature does, such as running a calculation. The URLs are also added to site maps. Those URLs are periodically tested to discover source code flaws and bugs. They are also upgraded periodically with new advances.

This is an acknowledged flaw that would be the first major improvement to work on if developers are hired at some point. For the current release, as long as the source code truthfully backs up the tutorials, and the URLs in the tutorials work, some level of imperfection may be acceptable. At least for this imperfect developer.

I. Source Code Set Up

The references found in the Reference section explain everything needed to understand how to use this source code and how to deploy a resultant web application. Make sure to first read the .NET Core documentation prior to the ASP.NET and EF Core documentation. Visual Studio 2017 simplified compiling and deploying the software –earlier Visual Studio releases will not work with this software.

The appsettings.json file and the Startup.cs file in the DevPacks web site folder show that either a localhost or azure platform is targeted by commenting in and commenting out appropriate settings. User secrets are tested by commenting in and out settings in Startup.cs.

J. File Storage

Files are stored in the file system for web servers. Files are stored in both the file system and blob for azure servers. Both servers place potential limitations on the total amount of space devoted to storage. In addition, storage generally costs money. Temporary files are stored in the wwwroot\resources\Errors, wwwroot\resources\temp, and an azure container named "temp". Files in those locations should be deleted during periodic maintenance. Depending on the space limitations imposed by Internet hosts and the number of members and clubs, it's technically feasible to bump up against storage limitations. The same holds true for database storage. Right now the probability of running out of storage space appears low, but a more permanent storage solution has to be considered for more scalable deployments.

K. Release Schedule

New releases occur between 1 month and 4 months. It usually takes more than 1 release to get any new feature fully working. Developer burnout is avoided by making releases when the bulk of a new feature works. Oftentimes, it makes sense to forgo downloading monthly releases until documentation clearly demonstrates that a new feature fully works. **Appendix E** lists changes associated with each release.

The author sees no clearly defined end to this type of software product –technically, thousands of algorithms remain to be built (although finding people who recognize the implications of automating, or deprecating, microeconomics might be a tad dificil (**3***)).

L. Database Updates

Possibly the hardest part about keeping third party deployments synchronized with the monthly releases is updating required resources that are stored in the database. The most prominent of these resources are new versions of calculators and analyzers (Linked Views), and stylesheets. Almost every release includes either a new or updated Linked View or stylesheet. Updating the source code, or binaries, alone won't keep those resources synchronized.

Appendix F lists links to each release's db changes. Some of these changes will be sql scripts, most will be links to updated stylesheets, linked views, or modified TEXT datasets. Source code users trying to keep third party deployments synchronized, should expect to work with both the source code and the db updates. As mentioned, a better alternative is to participate in the open source project or request a fee-based subscription.

Production web sites, such as an Azure deployment, require deploying a cleaned up default database, DevTreksDesk. The database is cleaned up by deleting unnecessary content, such as test data. Care is needed not to delete "default rows" found in most tables. These default rows, which can be identified by their name (i.e. Default Input), or Id (i.e. 1, numbers that aren't generated randomly), are used by the "Add Default" user interface workflow to insert new default rows into tables. We suggest first getting the default database working on the production server prior to deleting unnecessary data. A future release may include a cleaned up default database.

M. International Use (4*)

Although this version can be deployed anywhere in the world, including the numerous Azure regions, full scale international use requires full support from IT staff (**3***).

Most UI strings are stored in resource files. The DevTreksResources.es.resx file in the DevTreks.Resources project demonstrates how to use resource files to translate the UI strings. Version 2.0.0.beta2 began fully supporting localization of those strings –but it's not entirely clear how several localization configurations are used (i.e. see the Startup.cs file). Although it's not hard to add missing strings to the localization projects, some developers may prefer to participate in the open source project so that other developers can also benefit from their improvements.

Translation of the majority of site content (i.e. the results of running calculations) requires translating the stylesheets found in each MEF extension. That may not be the best approach. Alternative approaches include using the Media View of calculated results to display translated summaries, using the Story Telling feature with translated base story stylesheets to explain

translated results, using the Resources feature to store pdfs and images of translated site content, or developing a separate, complementary site, such as a blog site, that summarizes the results of calculations.

N. Project Participants (5*)

The author may welcome professional software developers or information technologists who desire to assist with the project. First, review the source code, compile the binaries, go through some tutorials, experiment using the code, and assess your individual "gusto". If you are enthusiastic about making a contribution (i.e. specializing in unit tests, focusing on the missing climate change algorithms), send a brief summary about how you think you can assist to the following email. The author is the developer and gets busy with development work –a reply will not be immediate.

devtrekkers@gmail.com

Summary (6*)

This source code may help people to improve their lives and livelihoods.

References

https://docs.microsoft.com/en-us/aspnet/core/migration/20_21?view=aspnetcore-2.1

https://docs.microsoft.com/en-us/dotnet

Latest ASP.NET Core 1 documentation: https://docs.asp.net/en/latest/

Latest EF Core 1 documentation: https://docs.efproject.net/en/latest/

Footnotes

1. Three primary data formats are used throughout DevTreks –xml, text, and html. The author is well aware that most development "these days" takes place using json. The

author started this project before json became popular and sees no particular reason for switching because: 1. The front end client serves up static html files –xml or json is never served up to the client, 2.. The www.w3.org considers xml and xslt to be mature, but institutional, Internet data formats (see https://www.w3.org/TR/xslt-30/). It's true that there could still be performance advantages associated with using json formats –but mostly for data owners (who are assumed to be primarily interested in ensuring the transparency, or static html delivery, of their content).

2. The Social Budgeting tutorial documents that, although all content is delivered through social networks, those networks are not typical social networks. They are designed for professionals who prefer to develop and deliver sound scientific content.
3. Most conventional institutions, especially those working in the government and research arenas, misallocate staff. They still hire staff with skills appropriate for previous decades but not with the IT skills urgently needed to address today's problems. The needed new skills include being able to ask and answer the right questions and having the character to understand the significance of "$28" donations. Those institutions need to be reformed and their dinosaurs gracefully retired.
4. Sound, global, scientific data has to settle on an official, or at least primary, language. English has been chosen for this purpose.
5. The author is not concerned about the current social media fad with the numbers of "followers" or "likers". Nor should you -with this source code, a more appropriate concern is with improving lives and livelihoods by automating, or "deprecating", microeconomics (i.e. why exactly is physics used to explain firm behavior?). The Social Budgeting tutorial explains that DevTreks takes the long view about how technology evolves to accomplish these tasks. That 'long view' is measured in technological development and adoption time, not geological time. The Social Performance tutorial points to the right, Integrated Valuation, direction.
6. This footnote is last because DevTreks recognizes that some software developers (and countries) do not like using proprietary platforms and databases, such as Microsoft tools. Even when those tools are open source. After attending a couple of Open Source Development conferences in Portland, OR, USA several years ago, the author concluded

that, from an economist perspective, every major platform and database is proprietary. It comes down to developer preference.

References Note

We try to use references that are open access or that do not charge fees.

Improvements, Errors, and New Features

Please notify DevTreks (devtrekkers@gmail.com) if you find errors in these references. Also please let us know about suggested improvements or recommended new features.

A video tutorial explaining this reference can be found at:

https://www.devtreks.org/commontreks/preview/commons/resourcepack/Deployment/348/none

Appendix A. License

THE SOFTWARE.

Except as contained in this notice, the name of DevTreks and Kevin P. Boyle shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from DevTreks and Kevin P. Boyle.

DevTreks and the DevTreks logo are trademarks of the nonprofit organization, DevTreks.

Appendix B. ReadMe.txt

Version: 2.1.6, September 13, 2018

Introduction

DevTreks is a multitier ASP.NET Core 2 database application. The web project, DevTreks, uses an MVC pattern. The data layer, DevTreks.Data, uses an EF Core 2 data repository pattern. EF data models are stored in the DevTreks.Models project. ASPNET Identity models are stored in the DevTreks web Project's Data folder. Localization strings are stored in the DevTreks.Exceptions and DevTreks.Resources projects. The DevTreks.Extensions folder holds projects that use a Managed Extensibility Framework pattern. Each project holds a separate group of calculators and analyzers.

Always visit the What's New link on the home site for the latest news. The What's New text file lists tutorials that have been upgraded recently. Those tutorials are usually associated with the current release. The Source Code tutorial explains how the source code works. The Social Budgeting tutorial explains how to manage networks, clubs, and

members to deliver social budgeting data services.

The Calculators and Analyzers tutorial explains

how calculators and analyzers work.

home site

https://www.devtreks.org

source code sites

https://github.com/kpboyle1/devtreks2.1 (.NET Core 2.1)

database.zip file

https://devtreks1.blob.core.windows.net/resources/db216.zip

What's New in Version 2.1.6

1. This release upgraded to AspNet Core 2.1, fixed bugs found during further 2.1.4 testing, and upgraded the calculator patterns.

Server version: Sql Server 2017 Express, Version 17.7

connection string

Server=localhost\SQLEXPRESS;Database=DevTreksDesk;Trusted_Connection=True;

DevTreks default member login

Name: kpboyle1@comcast.net

Pwd: public2A@

system administrator

SqlExpress 2017 databases can be accessed using a Windows OS logged in user –these haven't been tested with the new db server and aren't critical for accessing the db in SSMS

User: devtreks01_sa or sa

Pwd: public

Appendix C. Publish and deploy to an Azure App Service web app

DevTreks

Make sure to use the correct release settings in the web project's appsettings and Startup files. The Sql Azure database connection string must be in the ReleaseConnection and the blob storage connection string must be in the ReleaseStorageConnection. Don't commit appsettings.json to github before removing both strings.

The application is published to a staging deployment slot in an Azure App Service web app. Visual Studio is used for the deployment. The staging slot is then swapped out with the production web site. Version 2.0.0 is the first version that has been deployed to an Azure App Service web app –previous versions used an Azure Cloud Service. The only disadvantage identified so far with the new web app is that the settings used to upload large videos can no longer be used, and large videos can't be uploaded. DevTreks recommends using Microsoft's AzCopy tool to upload large files to Azure storage, with syntax similar to:

AzCopy /Source:*fullfolderpath* /Dest:*fullblobcontainername* /DestKey:*key* /Pattern:*fullfilename*

DevTreksStatsApi

Data Sciences Machine

Version 2.0.2 added this ASP.NET Core 1 WebApi app so that statistical scripts could be run on remote virtual machines, including Windows, Linux, and Mac servers. DevTreks is the client that consumes the statistical result returned in the response. Although any statistical or mathematical library can be used on the virtual machine, the current source only supports Math.Net, R, Python, Julia, and Azure Machine Learning.

The tests completed as of the reference date reflect deployment of the WebApi app to an Azure Data Sciences Virtual Machine that has been deployed to a Windows server (i.e. devtreksapi1.southcentalus.cloudapp.azure). Further documentation can be found in the

Conservation Technology Assessment (CTA) 2 and 3 references. The general process is explained in the following URL.

https://aspnet-aspnet.readthedocs-hosted.com/en/latest/publishing/iis.html

Because this virtual machine already had Visual Studio installed, the WebApi app was actually compiled and published directly on the virtual machine. Quite a few mistakes were made in setting up the IIS proxy server to Kestrel. Linux servers use proxy servers such as NGINX. The instructions must be followed closely. Although not clearly documented, the IUSER\DefaultAppPool must be given read/write permissions on the wwwroot/resources folder. The author tests with full permissions.

For testing purposes, a relatively inexpensive virtual machine server package ($50/month), employing limited processing and storage capacity, was chosen. Although the CTA 2 reference documents that Linux web servers that cost around $10/month can also be used, the potential money that can be saved using CTAs far outweighs server costs. Linux remains to be tested.

Appendix D. Publish and deploy to localhost

DevTreks

While an Azure web app uses blob storage, a web server release uses the file system for storage. By default, a localhost deployment uses the Kestrel web server found in ASPNET core web applications. The ASP.NET documentation in the references explain more about deployment to other commercial web servers, including IIS and Apache.

ASP.NET Core 1 applications can be deployed several different ways –using the .NET Core dotnet publish command, employing MSBuild directly, or using the publish feature in Visual Studio. "Free" versions of all of those tools exist. Deployment instructions can be found in the References. This release used Visual Studio.

1. Make sure to use the correct release settings in the web project's appsettings.json file. The ReleaseConnection string must be the connection string to the local DevTreks database. The local DevTreks database does not hold the same content as the Azure database, For example, tutorials are not added to the local db. Clicking on What's New with the local db returns a blank page of html. That can be corrected by following the same tutorial data structure as found on the Azure site (i.e. use Resource base elements to store the tutorials in the Commons Network).
2. Set up a file system folder at C:\DevTreks.2.1.0 (or any convenient folder). Publish using a custom profile pointing at this folder with a Release configuration. Delete any files found in the wwwroot/resources folder before publishing.
3. Review the following documentation about using Kestrel to run AspNet Core web apps from https://localhost:5001 : https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel
4. Use a command prompt to run the script, "dotnet DevTreks.dll", in the step 1 folder holding the published application. The command window will return a message stating that the Kestrel server is listening at port http://localhost:5000 and https://localhost:5001.

5. Open a web browser and go to https://localhost:5001. Localhost uses a default aspnet core SSL certificate which web browsers will not recognize as secure. Manually set the web browser to accept the "unsecure" URL. Somewhat of a hack, but browsers don't consider any of the development certificates to be secure outside of the development environment. The fix is to use a 3rd party SSL certificate.
6. All media files, including DataURL TEXT files, have to be previewed by their owner before being used. The Preview runs code that copies the media and data files from the database to the file server.
7. Version 2.1.0 onwards noticed that some browsers are caching TEXT datasets. Even when a new TEXT file is uploaded and stored properly in the file system, the URL to the file returns the older, cached version. Browser history has to be deleted to verify the new TEXT files.
8. Run DevTreks normally, ensuring that localhost:5001 is used in any URL, such as Data URLs.
9. When finished, use CNTRL+C to close the DOS command window.

When testing on the development host (i.e. http://localhost:55009/) developers may want to first run the dotnet DevTreks.dll script so that resources, such as datasets and images, can load from localhost:5001 too. Make sure to first run step 4's script so that Kestrel is running. Many calculations and analyses use that host to reference data URL and media URL resources.

DevTreksStatsApi

The deployed localhost:5000 DevTreksStatsApi app can be tested by running the script, "dotnet DevTreksStatsApi.dll", from the root of the deployed application (i.e. c:\\DevTreksStatsApi.2.0.2). Then paste the link, http://localhost:5000/api/statscript, in a browser to verify that the default GetAll() controller action runs successfully. The response from the host is a default json string holding a StatScript object's default properties. The remaining tests use a new Client (console app) that has been added to the DevTreks solution.

Appendix E. Release Changes

Version 2.1.6, September 13, 2018

- The web project has been upgraded to an entirely new AspNet Core 2.1 project with appropriate migrations. Microsoft plans to provide full support for 2.1 for the next 3 years (i.e. long term support), which is considered the useful life for the existing code.
- Security has been strengthened by upgrading to the AspNet 2.1 Identity pattern, including requiring “https” on localhost too. With 1 minor exception, the new Identify has not been customized (i.e. the Identify Razor class package has not been altered). The result is stronger authentication, including compliance with EU requirements. The existing network-club-member authorization pattern has been changed to require new members to join existing clubs. Newly registered members must ask club admins to add them to clubs (i.e. they no longer automatically have their own club).
- The 2.1.6 source has been added to a new github branch because the source was built from an entirely new Visual Studio 15.8+ solution. Experience suggests cleaner source results from using a new solution rather than migrating from the existing source.
- Bugs found in the recent tutorials upgrade were fixed.
- The calculator patterns were upgraded and strengthened.
- Version 2.1.6 couldn’t be upgraded to the latest javascript and html libraries (i.e. JQuery 3.3.1) because JQuery-Mobile hasn’t been updated recently. We’ll keep an eye out for the future of these supporting technologies. Future releases will consider upgrading to alternative javascript/html libraries (i.e. bootstrap, angular) for long term “code sustainability”.

Version 2.1.4, June 19, 2018

- The database has been updated to SqlServer 2017, Version 17.7.
- Some instability has been noticed when saving calculations. Version 2.1.6 attributed this to the Azure host traffic.

Version 2.1.4, June 01, 2018

- This release fixed bugs associated with the upgraded calculator patterns introduced in the last release. The fixes have been pushed to github.
- Three of the Social Performance references have been updated to reflect the changes. All of the SPA URLs were tested with the latest code and passed (i.e. except the recalcitrant Example 6B on the cloud).

Version 2.1.4, May 17, 2018

- This release focuses on sustainability, disaster risk management, consequential digital activism, deprecating microeconomics, next generation jobs, and the quest for meaningful lives.
- Machine learning algorithm development has just started and will continue with the next release.
- Quite a few changes took place with how algorithms are run and supported –mainly aimed at machine learning algorithms. For example, Stock Calculators now use the Score.JointURL, rather than the Score.DataURL property to hold their calculations. Many of the new patterns are documented in the Social Performance 3 reference. These new patterns have introduced some instability in this release (i.e. Example 6B broke) and require updating some older calculations.
- This release includes new, but tentative, support for Accord. Both Accord and MathNet are now included via Nuget packages. MathNet's source had been included in previous releases.

Version 2.1.2, December 1, 2017

- This release upgraded 4 algorithms documented in the Social Performance Analysis tutorial. Specifically, this release supports fuller sustainability assessments, involving the integration of Product LCIA, Organization LCIA, Social LCA, Hotspots Analyses, Life Cycle Costs and Benefits, and Cost Effectiveness Analysis.
- The Version 2.1.0 refactor was further tested and minor code improvements made. Identity security was retained because it went through a major refactor in Version 2.0.0

and minor refactor in Version 2.1.0 and because climate change looms. Linux and Mac server testing are not current priorities.

- Version 2.1.4 will focus on upgrading the Conservation Technology Assessment techniques. Specifically, new, open source, machine learning, libraries may be supported.

Version 2.1.0, October 02, 2017

- The September 20 release contained bugs which were fixed and updated on github last week. Localhost and cloud tests were carried out using the URLs in many tutorials. The tests passed and the tutorials updated to show the results. The "netframework4.6.1" github branch is now considered obsolete. So on to Version 2.1.2.

Version 2.1.0, September 20, 2017

- The Version 2.1.0 refactor upgraded the ASPNET web project to.NET Core 2.0 and the class libraries to .NET Standard 2.0. The refactor includes the upgrade of several patterns, resulting in some probable instability until more complete testing can be done. At least 1 more version release is needed before stability is restored.
- Linux and Mac server testing have not taken place yet. It's very likely that some conventions, especially dealing with file system storage, may need tweaking or refactoring before the software is 100% cross platform.
- The Managed Extensibility Framework (MEF 1) Pattern used by Extensions was upgraded to a new, simpler, MEF 2 pattern (i.e. using System.Composition). In addition, 1 of the 2 Event patterns used to run calculators and analyzers was eliminated because it used a lot of duplicate code. The remaining pattern had to be tweaked to run on the new NetCore libraries.
- The Asynchronous patterns were overhauled. The partial patterns used in previous releases were upgraded to fuller patterns, requiring many changes to many method signatures. Specifically, byref and output method parameters were largely eliminated (but the majority were not actually needed). The Monitoring and Evaluation or ME2, and Resource Stock or SB1, Extensions highlight the improvements –now they mostly run asynchronously. The large number of changes require more comprehensive testing.

- The Resource Stock Extension, SB1, was refactored –the existing pattern experimented with new asynchronous events, but used a lot of duplicate code. The duplicate code has been eliminated.
- Although the FoodNutrition and HealthCare Extensions were refactored, and appear to work fine, they are not actively supported. The Malnutrition Extension, MN1, which uses the USDA ARS SR database, takes precedence over the "Food Facts" Extension. The HealthCare Extension is only included, so far, as an example of domain-specific tools in the Health Care tutorial.
- A lot of minor code changes were made, most involving the refactor. These changes are only documented in the source code itself. The overall code base was largely compatible with the new core libraries –most code was running successfully after 2 weeks of refactoring.
- Identity security will not be substantially changed for this release –the only tests involved logging in an existing member. A refactor will be investigated for the next release.
- DevTreksStatsApi will not be upgraded until the next release. This release only upgraded local R and Python script execution (i.e. Examples 1 in the CTA 02 and 03 references). AML testing is also put off until the next release.
- The DevPacks application refactor will not be made until the next version release.

Bugs and Miscellaneous Improvements:

Remaining Todos: Azure cloud testing; several suspect code changes or bugs

Two new examples were added to the Social Performance Analysis tutorial. These examples demonstrate conducting Cost Effectiveness Analysis, analyzing climate change using Natural Capital Care Perspectives, and using QASYs as replacements for QALYs and DALYs.

The Social Performance Analysis tutorial provides clearer documentation for using Monitoring and Evaluation to integrate Performance Monitoring and Impact Evaluation. Substantially more

testing and development are needed before Footnote 5 is fully addressed, but the Social Performance Analysis tutorial points to the right, Integrated Valuation, direction.

Version 2.0.8, May 10, 2017

- **Social Performance Analysis Tools**: This release includes 4 new algorithms designed to measure Social Performance. A new reference, Social Performance Analysis, documents how these algorithms work. The reference can be found in the Performance Analysis tutorial.
- **Tutorials:** Several references in several tutorials are being updated to address the changes associated with the new tools. The Life Cycle Calculation reference has changed in order to comply with one of the new algorithms.

Bugs and Miscellaneous Improvements:

The existing CTAP and new RCA algorithms returned datasets with column names that interfered with Math Expressions (i.e. QTM). The Math Expression automatically removes column names when parsing formulas (i.e. I4.QTM becomes I4). The column names causing the interference were changed.

The Monitoring and Evaluation CTAP algorithm for carrying out Multi-Criteria Assessment no longer returns a mistaken error message, even when no errors occur.

Version 2.0.6, November 23, 2016

- The Version 2.0.4 MEF Extension upgrades were tested and improved to work with all of the Conservation Technology Assessment (CTA) algorithms. The CTA references were updated with M&E examples.
- Several algorithms in the Technology Assessment tutorials were upgraded to work with M&E calculations and because the additional tests revealed new improvements with some of those calculations. Major changes were not made to these algorithms because this released focused on the M&E tools.

Bugs and Miscellaneous Improvements:

- The Visual Studio/ASP.NET, Sql Server Development Tools, and Azure SDK, development tools were updated to their November, 2016 updates.

Version 2.0.4, November 09, 2016

- The MEF Extension, ME2, holding Monitoring and Evaluation Calculators and Analyzers, was upgraded to support the measurement of risk and uncertainty in M&E indicators. The Monitoring and Evaluation tutorials were upgraded to document the changes. A major advantage to the upgrade is that all of the CTA algorithms, documented in the Technology Assessment tutorials, can also be used to conduct Monitoring and Evaluation calculation and analysis.
- The MEF Extension, SB1, holding Resource Stock Calculators and Analyzers, was changed by no longer rerunning base element Resource Stock calculations during analyses. The Analyzers don't change the original base element calculations, so running calculations twice is unnecessary. The same pattern is now also used with the M&E Analyzers. Analyses now run much faster. The consequences of this change has not been fully tested with DevPacks yet.
- Several references in several tutorials were updated, or will be updated, to address the changes in the M&E and Resource Stock tools.

Bugs and Miscellaneous Improvements:

- The bug, or more specifically, error message, documented in the Version 2.0.0 release notes related to connection strings resurfaced in the NPV calculators but for a different reason. It likely affected new deployments to clean machines. The stylesheets used with those calculators make function calls that can involve retrieving data, such as lists of data, from the file/blob system and database. The following changes were made to further deal with Version 2.0.0+ bugs associated with the retrieval:
    - Stylesheet calls that involved storing resources in file or blob storage weren't properly setting the uri.URIDataManager.PlatformType property. As a result,

resources weren't being saved correctly. A new condition was added to DevTreks.Data.FileStorageIO.URIAbsoluteExists() to double check that property.

- All function calls to Helpers.FileStorageIO.GetXmlReader(uri, docPath) were checked and, when needed, the condition for dealing with a null reader result was strengthened by not generating an unhandled exception. That means calculators, such as the NPV calculators, can still be run, even when the resources used in those calculators (i.e. lists of Units, Currencies, Interest Rates) don't load successfully. The latter condition should be, technically, impossible (except when bugs like unhandled exceptions exist).
- Future releases will try to carry out more clean-machine tests. Specifically, the wwwroot/resources folder will be emptied prior to deployment to localhost and the Azure staging slot. [Full Azure clean-machine tests may be prohibitive at this stage.]

Version 2.0.2, October 04, 2016

- New postcompile scripts were added to each MEF Extension's project.json file. The script copies the calculator/analyzer dll to wwwroot/Extensions after the dll is compiled. MEF uses that path to find and run the calculators and analyzers. This technique is not currently cross platform, but every alternative explored has flaws. The Version 2.0.0 technique is being retained for potential cross platform use. That technique copies the dlls when data owners preview media resources using a Release build.
- A new ASP.NET WebApi web app runs R and Python scripts as documented in Conservation Technology Assessment (CTA) references, CTA 2 and 3. The source code has been added to the devtreksapi1 github repository. A new client has been added to DevTreks.Data.Helpers.WebServerIO to post a request to the API and process the response. The DevTreks.Extensions.Algorithms, Script1 and Script2, were upgraded.
- The CTA reference in the Technology Assessment 1 tutorial for the R statistical package, CTA Algorithm 2, documents the changes for Version 2.0.2, including the upgrade from

the regular R statistical package to Microsoft R Open. The cloud deployment now includes the option of using a cross platform Statistical Virtual Machine.

- The CTA reference in the Technology Assessment 1 tutorial for the Python statistical package, CTA Algorithm 3, documents the changes for Version 2.0.2, including the upgrade to Anaconda 4 with Python 3.5.2. The cloud deployment now includes the option of using a cross platform Statistical Virtual Machine.
- The CTA reference in the Technology Assessment 1 tutorial for the Azure Machine Learning (AML) statistical package, CTA Algorithm 4, documents the changes for Version 2.0.2, including new subalgorithms and examples demonstrating how to use AML with R and Python algorithms.
- The Conservation Technology Assessment 1 reference has been updated with algorithm6, Julia. Although most of the source code has been stubbed out to support that open source statistical library, it will not be tested for this release. For this release, the point is to reemphasize the importance of economies of scope (and scale).

Bugs and Miscellaneous Improvements:

- Visual Studio, Azure SDK, Sql Server SSDT, ASP.NET and Net Core 1.0 were updated to their latest releases on September 15, 2016. Within the solution, project.json files were updated to reference the new assemblies. (i.e. to 1.0.1).
- The reference to Bundle.Minifier.Core in the web project's project.json file was moved to the tools section of project.json. The prepublish command, dotnet bundle, uses that package to automatically bundle and minify the needed css and js files.
- The Preview panel has been upgraded to display an image or video of a base element's linked views (i.e. stored using the MediaURL property of calculators and analyzers). The Calculators and Analyzers reference has been upgraded to explain this change more thoroughly.
- A new AppSetting, PlatformType, has been added as a property to ContentURI.URIDataManager. All AppSettings are set in Startup.cs. Numerous method calls to GetPlatformType() were eliminated. FileStorageIO.PLATFORM_TYPES added

"none" to the enum list. DevTreks.Data.Helpers.FileStorageIO.GetPlatformType added an additional condition to account for azure machine learning urls.

- Command buttons with the text "Download Resource" were changed to "Download" so that they don't need 2 lines to display on some mobile devices.
- DevTreks.Data.Helpers.GetAppSettingsString() set the appsetting argument to string.Lower() to match the enum.
- DevTreks.Extensions.Algorithms was cleaned up by removing some deprecated algos. Test algos that might still be used in the future were retained (i.e. Bayesian algos).
- Each project's output path was cleaned up by removing folders holding older versions of Net Framework.
- Running the Make Base command button, on the Views panel, for Inputs and Outputs that don't have children series returns an error. The Make Base command button shouldn't be used except when an Input or Output has children –only Input and Output Series are used in budgets. The fix is to add at least 1 child series. The error message will be retained –some thought is required to use this software.
- In order to test an OLS regression using R and an AML web service, a new nuget package, Microsoft.AspNet.WebApi.Client 5.2.2, had to be added to the DevTreks.Data project. That package may not be cross platform yet.

Version 2.0.0, August 31, 2016

- Version 2.0.0 refactored **t**he source code for use on Windows, Linux, and Mac servers. This version has been refactored using the official ASP.NET and EF Core 1 libraries that were released June 27, 2016. The source code was added to the public github repository, devtreks, for the first time.
- All tutorials were upgraded in August, 2016 which entailed considerably more testing and the discovery of the 7 bugs fixed in the August 11 and 31 updates on github. Does that mean the source is bug free? No, but it does mean that the refactored source is basically sound.

- Connection strings can include characters that could break string array storage and manipulation. For example, the character @ initially broke a Stylesheet parameter call to Data.AppHelpers.Resources.GetResourceArraysAsync and its related StylesheetHelpers.WriteSelectListForNewView. Although that bug is fixed, it's possible that additional connection string characters could break those types of functions.
- The following error message occurred when first attempting to publish to Azure: "Could not load file or assembly 'Microsoft.DotNet.ProjectModel, Version=1.0.0.0, Culture=neutral, PublicKeyToken=adb9793829ddae60'. The system cannot find the file specified)". This message was corrected by clearing the nuget cache. To do so, download the nuget cli tool (nuget.exe), move it to the solution folder and run the following script from a DOS command prompt. The script removed all "old" cached nuget packages. Each project.json file was then changed slightly (i.e. by adding a blank line) so that new project references would load.

  nuget locals all –clear

Appendix F. DB URLs and SQL Scripts

Scripts will only be included for db changes. Scripts will not be included for changes to stylesheets or link views that are stored in Resources or Linked Views. These can be easily downloaded, then uploaded to other servers, using the URLs in this Appendix. Scripts will not include mundane changes, such as updating an image stored in a Resource. Good practice is to review new and updated references and to duplicate any new or updated examples on your own servers.

Appendix C points out that Version 2.0.0.beta2 starting using localhost:5000 rather than localhost.

May 17, 2018. Version 2.1.4

New Social Performance Analysis 3 examples were added to the localhost database. Several of the data files used in this tutorial exceeded the 500KB limit for database storage. Meaning they can't be automatically downloaded from the db to the file system by previewing the resources uri. The following zip file contains the missing data files –if testing, some of these files must be manually uploaded to the same resource. Future releases may decide not delete these files from the source folders prior to the release.

https://devtreks1.blob.core.windows.net/resources/network_carbon.zip

December 1, 2017. Version 2.1.2

New Social Performance Analysis examples were added to the localhost database. Those examples, including URLs, are documented in the Social Performance reference in the Performance Analysis tutorial.

September 08, 2017. Version 2.1.0

New Social Performance Analysis examples were added to the localhost database. Those examples, including URLs, are documented in the Social Performance reference in the Performance Analysis tutorial.

May 10, 2017. Version 2.0.8

The following 2 stylesheets changed because Life Cycle calculations were upgraded in this release. In addition, new Social Performance Analysis examples were added to the localhost database. Those examples are documented in the Social Performance reference in the Performance Analysis tutorial.

Replace this Life Cycle Cost calculator stylesheet (if needed, switch to the Reconstruction Science club)

https://devtreks1.blob.core.windows.net/resources/network_commercial/resourcepack_469/resource_979/LC1CostCalculator1.xslt

For the Life Cycle Cost calculator stylesheet in this group of stylesheets (pre May 11, 2017 dbs only)

https://localhost:5001/buildtreks/edit/commercial/resourcepack/Life Cycle Input Calculators and Analyzers/471/none

Replace this Life Cycle Benefit calculator stylesheet

https://devtreks1.blob.core.windows.net/resources/network_commercial/resourcepack_472/resource_988/LC1BenefitCalculator1.xslt

For the Life Cycle Benefit calculator stylesheet in this group of stylesheets

https://localhost:5001/buildtreks/edit/commercial/resourcepack/Life Cycle Output Calculators and Analyzers/472/none

November 22, 2016. Version 2.0.6

No changes to the database, stylesheets, or linked views took place, but additional M&E examples were added to the localhost database. Those examples are documented in the M&E and CTA tutorials.

November 07, 2016. Version 2.0.4

All of the M&E 2 Linked Views were updated. They can be found at the following URLs which are owned by the Food Nutrition club in the HomeTreks network group.

https://www.devtreks.org/hometreks/preview/farmworkers/linkedviewgroup/Monitoring and Evaluation Calculators/53/none/

https://www.devtreks.org/hometreks/preview/farmworkers/linkedviewgroup/ Monitoring and Evaluation 2 Analyzers/61/none/

All of the M&E 2 stylesheets were updated. They can be found at the following URL which is owned by the Food Nutrition club in the HomeTreks network group.

https://www.devtreks.org/hometreks/preview/farmworkers/resourcegroup/Monitoring and Evaluation 2 Styles/143/none

October 03, 2016. Version 2.0.2

The calculators in the following LinkedViewPack were upgraded to new versions to test the new MediaURL displays on the Preview panel. Similar Media URLs must also be added to all other base Linked View calculators and analyzers. These data services are owned by the Carbon Emission Reducers club in the GreenTreks network group (i.e. if needed, switch default clubs).

https://www.devtreks.org/greentreks/preview/carbon/linkedviewgroup/Stock Calculators/63/none/

July 8, 2016. Version 2.0.0

The EF files in the web project's Data folder were replaced with equivalent files taken from building a new MVC /EF project and then copying those files. The new files included the EF changes documented in the EF reference for a "rc2 to netcore" migration. Note that the EF Core 1 technologies are still officially in "preview" status.

Entity Framework appears to have trouble with table row insertions when the row inserted uses a foreign key = 0. That was the default NetworkId foreign key used by Service table insertions, or the Add New Service workflow in Service Agreements. A new Network has been added to table Network using the following script. This is now the default Network used by Service insertions. Have to admit –I don't like foreign keys = 0 either. The default database on codeplex has been updated with the updated Network table. Relatedly, storing Network connection strings in the database is no longer considered best practice and many of these Network properties are not currently used, but it wouldn't be hard to tweak them to support separate databases for separate Networks.

```
SET IDENTITY_INSERT Network ON

GO

-- Attempt to insert an explicit ID value of 1

INSERT INTO Network (PKId, NetworkName, NetworkURIPartName, NetworkDesc,
AdminConnection, WebFileSystemPath, WebConnection, WebDbPath, NetworkClassId,
GeoRegionId) VALUES(1, 'Default Network', 'defaultnetwork', 'Default Network when inserting
data', 'none', 'none', 'none', 'none', 1, 1)

GO

SET IDENTITY_INSERT Network OFF

GO
```

June 10, 2016. Version 2.0.0.beta2

The development database now uses Sql Express 2016, RTM. Azure uses RTM 12. Table AccountToLocal was further refactored. All the models and dbcontext were regenerated after the refactor using EF Core 1 scaffolding. The following 6 stored procedures have been refactored to support the new localization pattern (i.e. by changing the localaccountgroup condition in the sps):

GetLinkedViewsPage, GetLinkedViewsBaseId, GetLocalsByClubId, GetAncestorNamesForAdmins, UpdateLinkedViewIsDefault, InsertLinkedViews

For Azure, DevTreks deleted all of the existing ASP.NET Identity tables and allowed EF Core 1 to automatically create new ones (see the web project's Data folder and Startup.cs file). Existing Members, have been registered as new users and then manually configured to use the new ASP.NET log in (i.e. by copying the Id field of the AspNetUsers table to the AspNetId field of table Member). Table AccountToLocal was changed manually.

May 26, 2016. Version 2.0.0.beta1

This major refactor should first be tested using the codeplex database. The ASP.NET Identity tables have all changed. The AccountToLocal table is the only other table that was refactored. Existing databases will not work without refactoring, preferably using EF Core 1 migrations or scaffolding. The story-telling stylesheets are the only stylesheets that were upgraded and can be downloaded at the following url. Note that these data services are owned by the Core Economics Data club in the AgTreks network group (i.e. if needed, switch default clubs).

https://www.devtreks.org/agtreks/preview/crops/resourcegroup/Story Support Group/60/none/