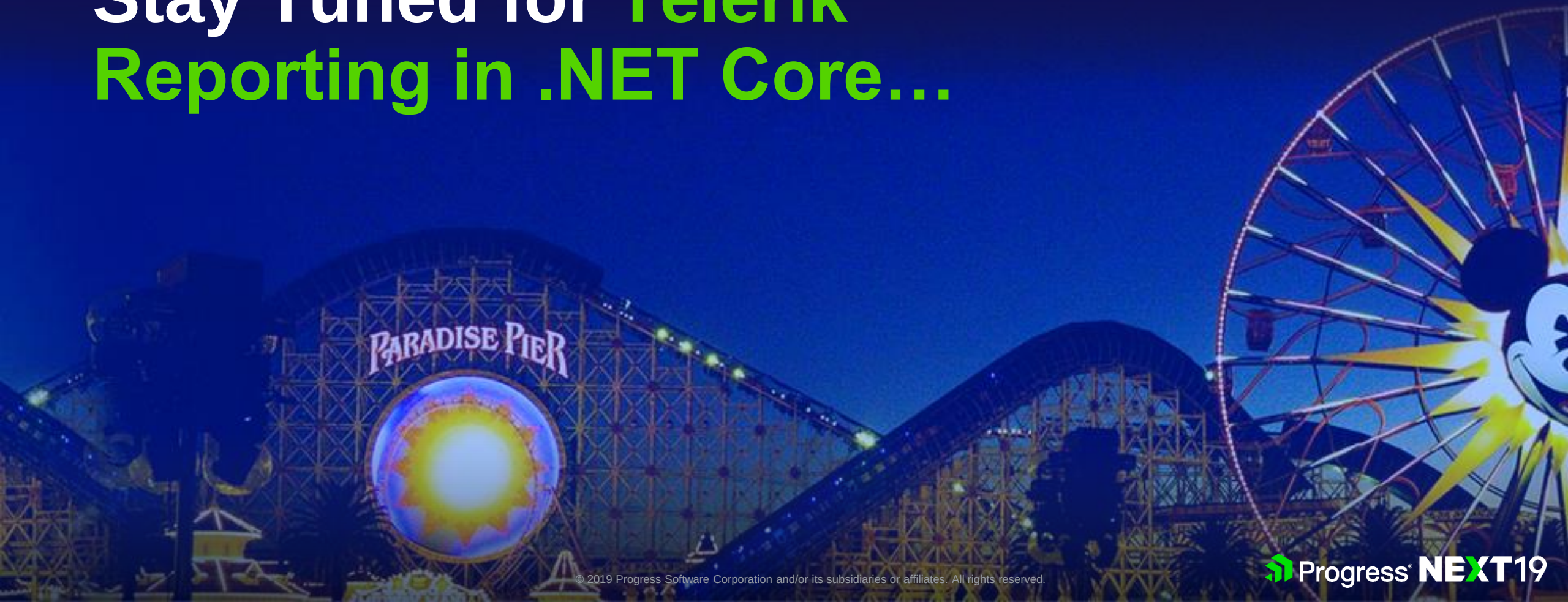
Stay Tuned for Telerik Reporting in .NET Core…



Progress NEXT19

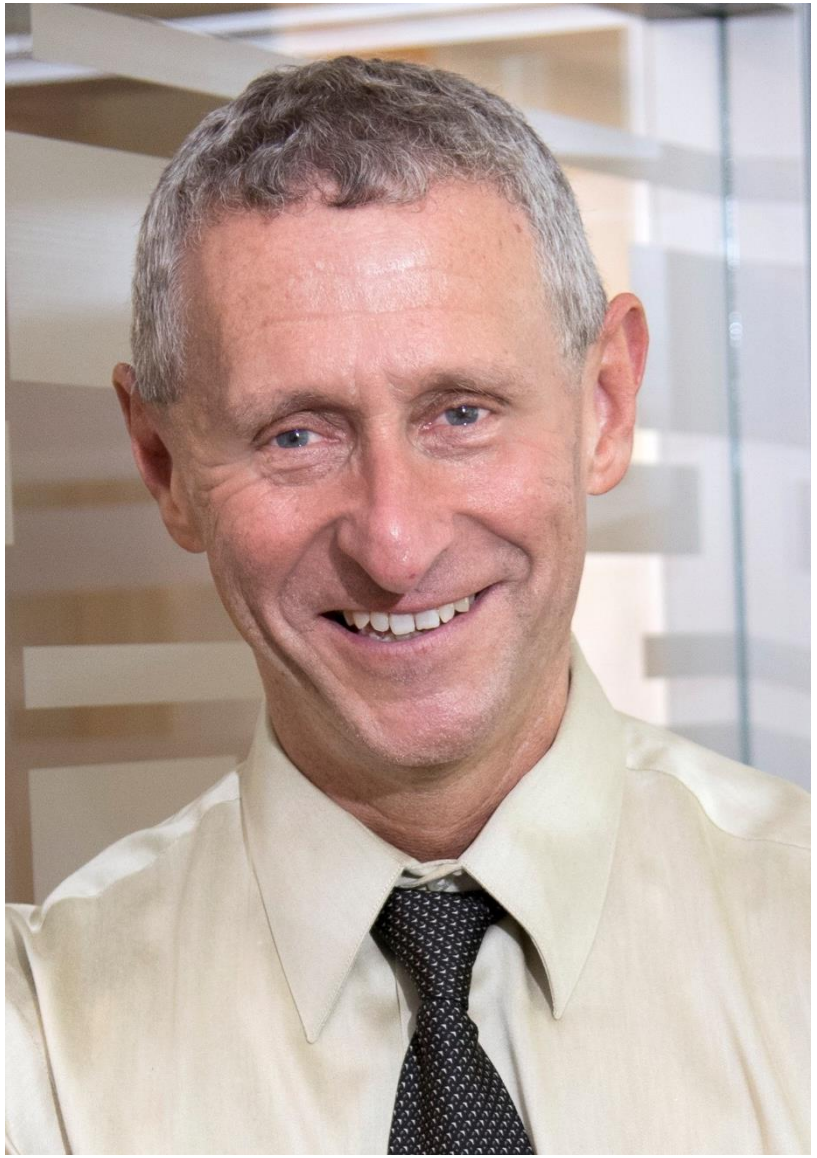
Telerik Reporting in .NET Core

“The Final Frontier”

Richard Zaslaw
Senior Sales Engineer – Developer Tooling
May 8, 2019

Introductions

- Senior Sales Engineer in the Progress Developer Tooling division
- Over 7 years with Telerik and Progress supporting our entire portfolio of desktop, web and mobile UI related products.
- Currently focusing efforts on the Progress/Telerik Report Solution and our Kendo UI framework (Angular, React, JQuery, .NET)



Agenda

- Progress Telerik Reporting Solution
- Designing Full-Featured Reports
  - Standalone Report Designer
- Publishing Telerik Reports
  - Telerik Reporting REST Service
  - HTML5 Report Viewer (JavaScript SPA apps, .NET (Webforms, MVC and ASPNET.Core)
- .NET Standard 2.0 and .NET Core Evolution: *"The Final Frontier"*
  - **New Feature!** Cross-Platform Support (Linux, macOS and Windows)
- Displaying Reports in .NET Core 3.0 Desktop Apps
  - .NET Core 3.0 Desktop Integration

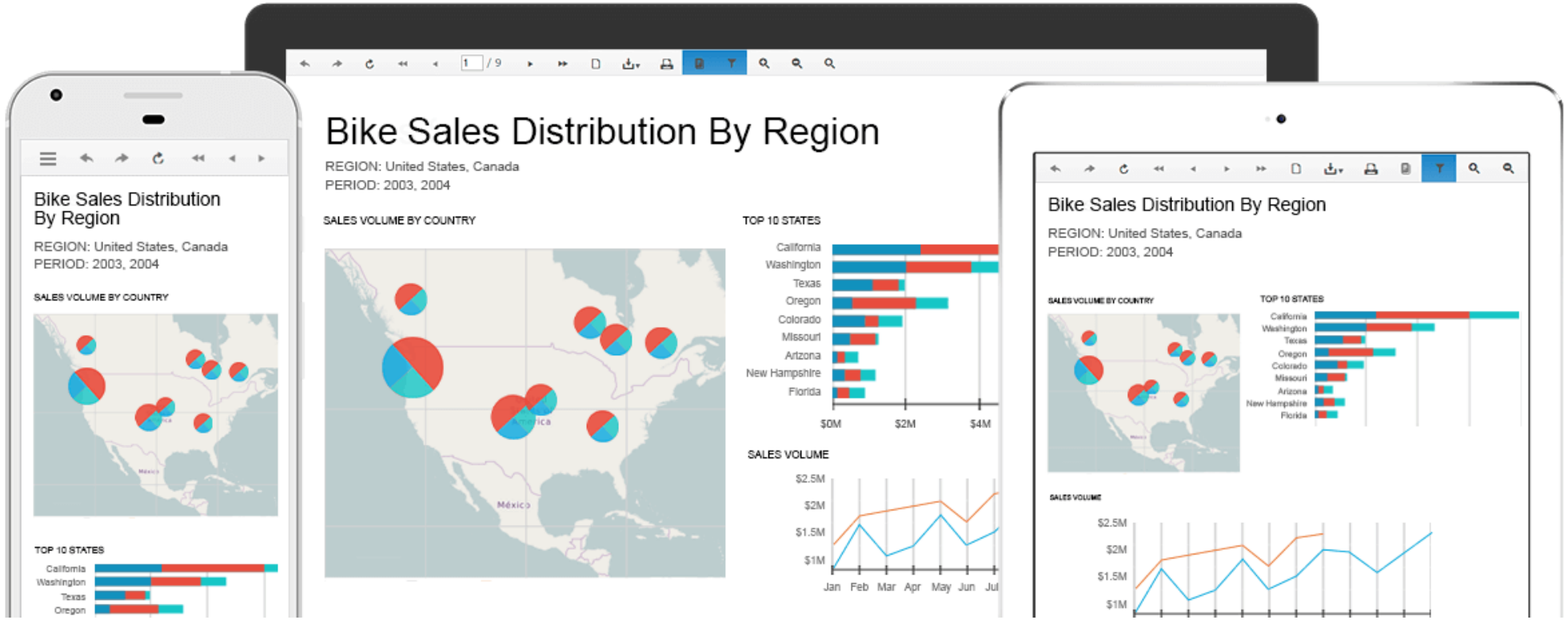
Progress®
NEXT19

Progress® Telerik®

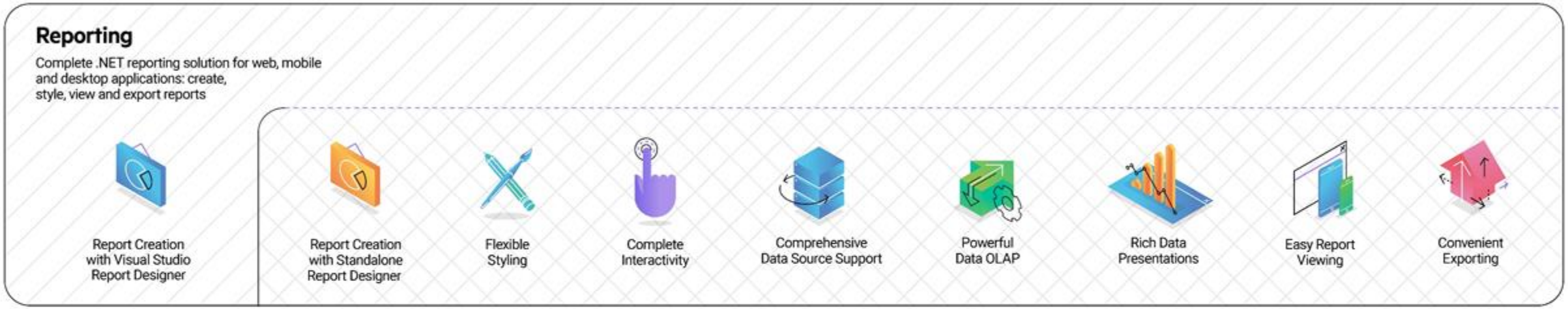
Reporting Solution

Telerik Reporting

- **Lightweight** .NET reporting platform designed to satisfy the reporting needs of the modern business
- **Mature** product, with a complete feature set for creation and delivery of richly styled, fully interactive reports.



Telerik Reporting



Telerik Reporting + Telerik Report Server

Reporting

Complete .NET reporting solution for web, mobile and desktop applications: create, style, view and export reports

Report Creation with Visual Studio Report Designer

Report Creation with Standalone Report Designer

Flexible Styling

Complete Interactivity

Comprehensive Data Source Support

Powerful Data OLAP

Rich Data Presentations

Easy Report Viewing

Convenient Exporting

ADD-ON

Report Server

Complete .NET report management solution: create, store, manage, schedule and view reports from a single web interface

Easy Report Storage and Access

Streamlined User Management (LoB)

Custom Report Scheduling (LoB)

Relevant Data Alerts (LoB)

Universal FREE Viewing (LoB)

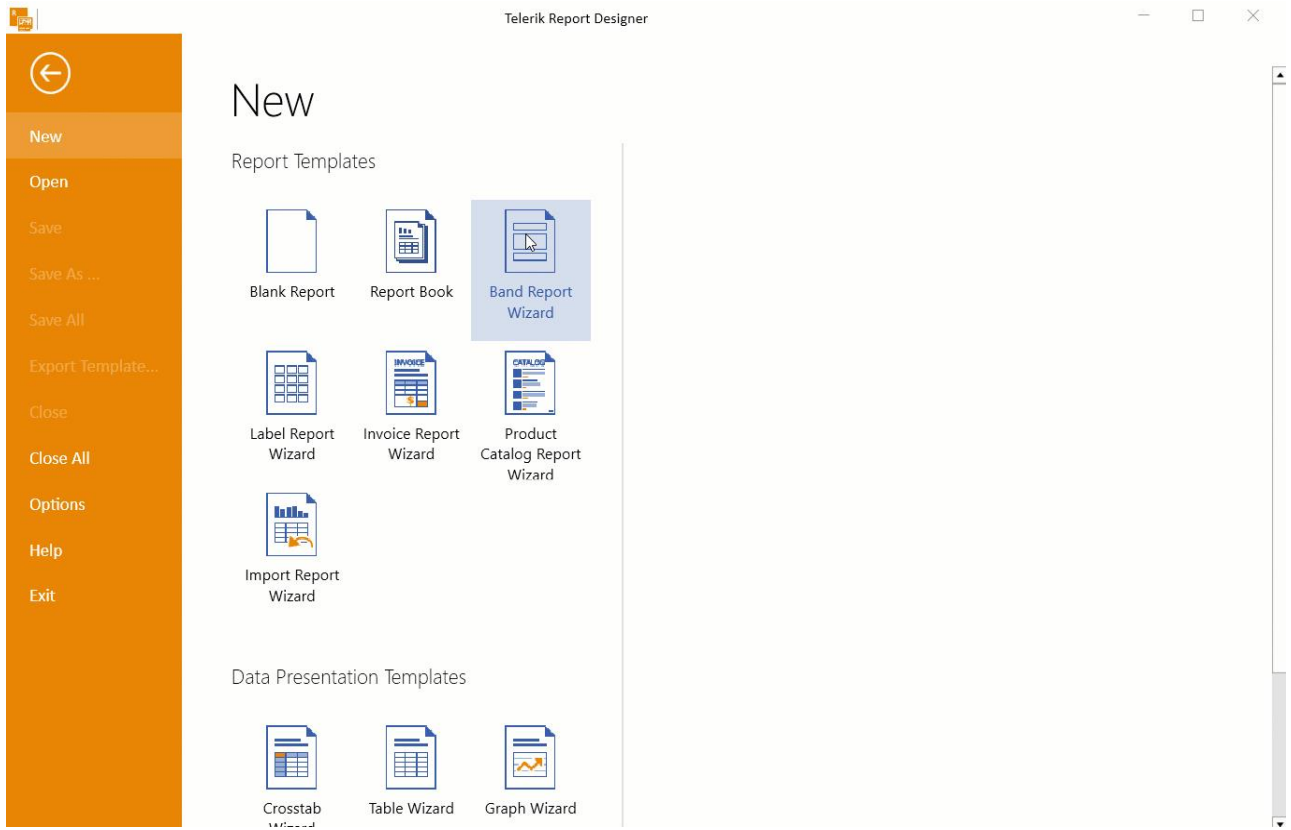
*LoB – Line of Business feature

Progress®
NEXT19

Designing Full-Featured Reports

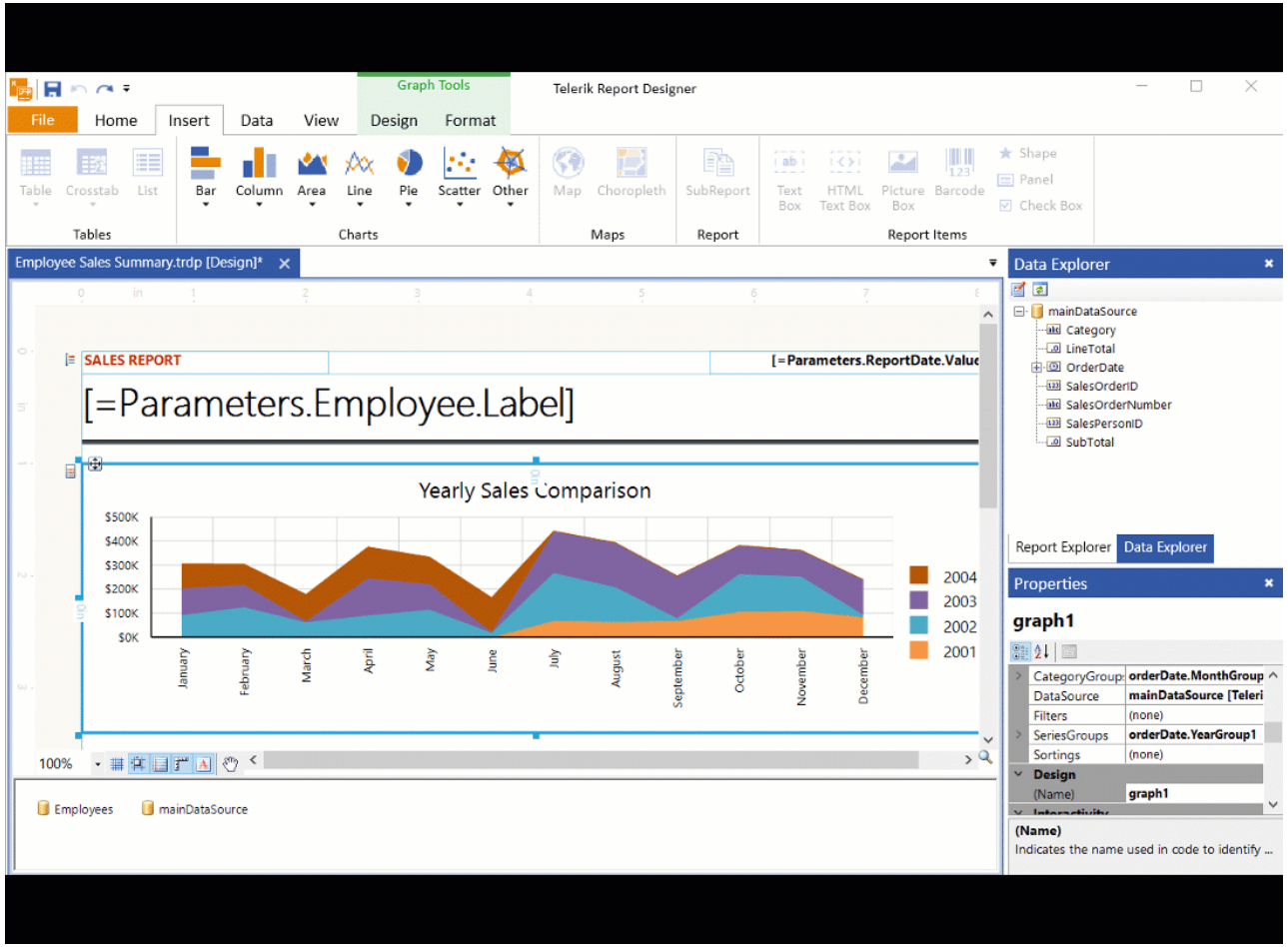
Rapid Report Development

- Telerik Report Wizard
  - Select predefined layout
  - Configure data source
  - Choose professionally styled theme
  - Codeless, intuitive report creation
  - Preview report
- DataSource Components
  - SQLDataSource
  - CubeDataSource
  - EntityDataSource
  - CSVDataSource
  - WebServiceDataSource
  - ObjectDataSource



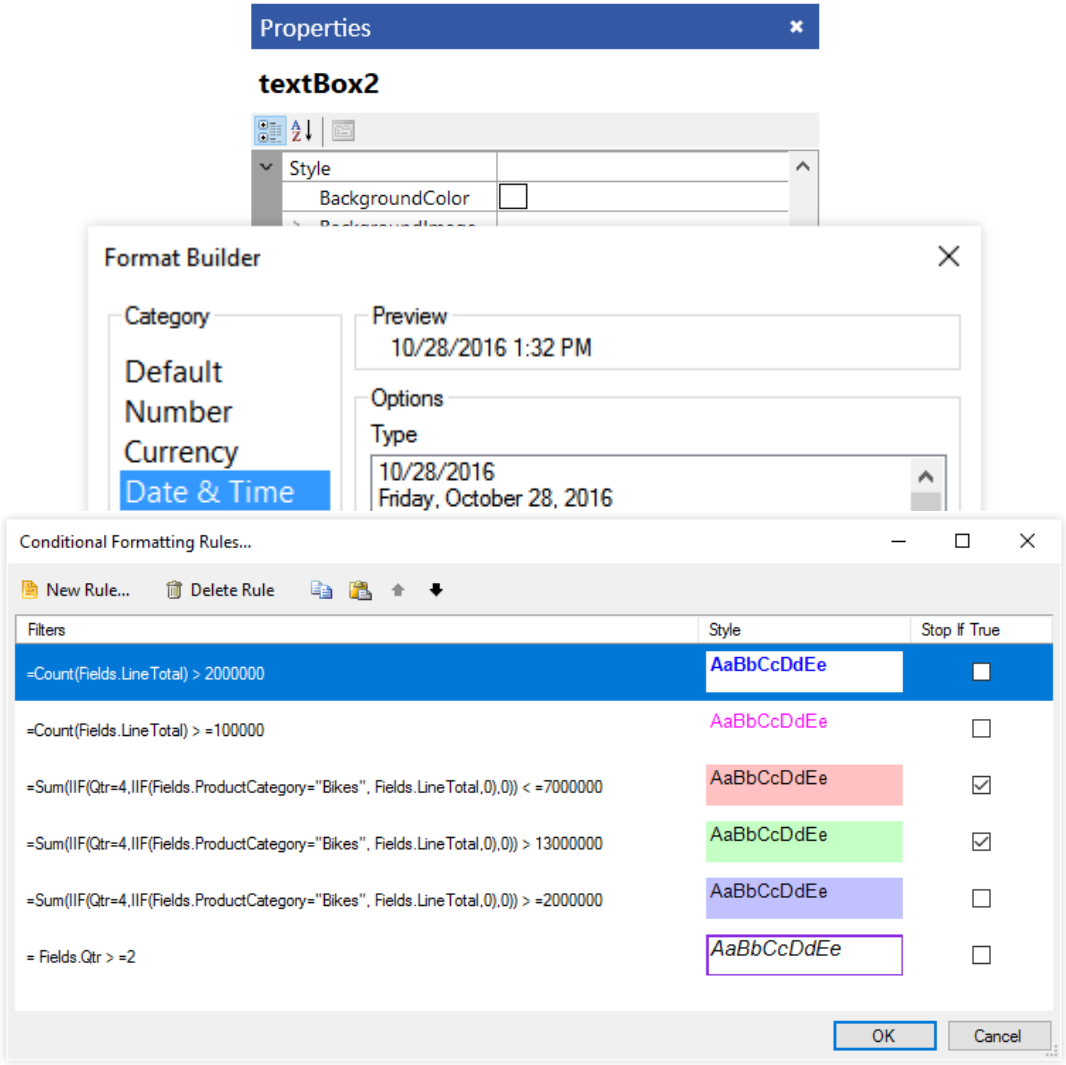
Standalone Report Designer

- **Desktop** designer enables **power users** to modify existing or build new reports
- Codeless, drag and drop designer
- Add/position/style report items (TextBox, Charts, Tables) + implement data-binding
- Preview changes as you design
- Pixel-perfect rendering
  - Designer canvas simulates graph paper and represents the actual layout of the report elements as they would appear on a printed sheet of paper
- **.trdp** file format (XML-based/ Packaged)



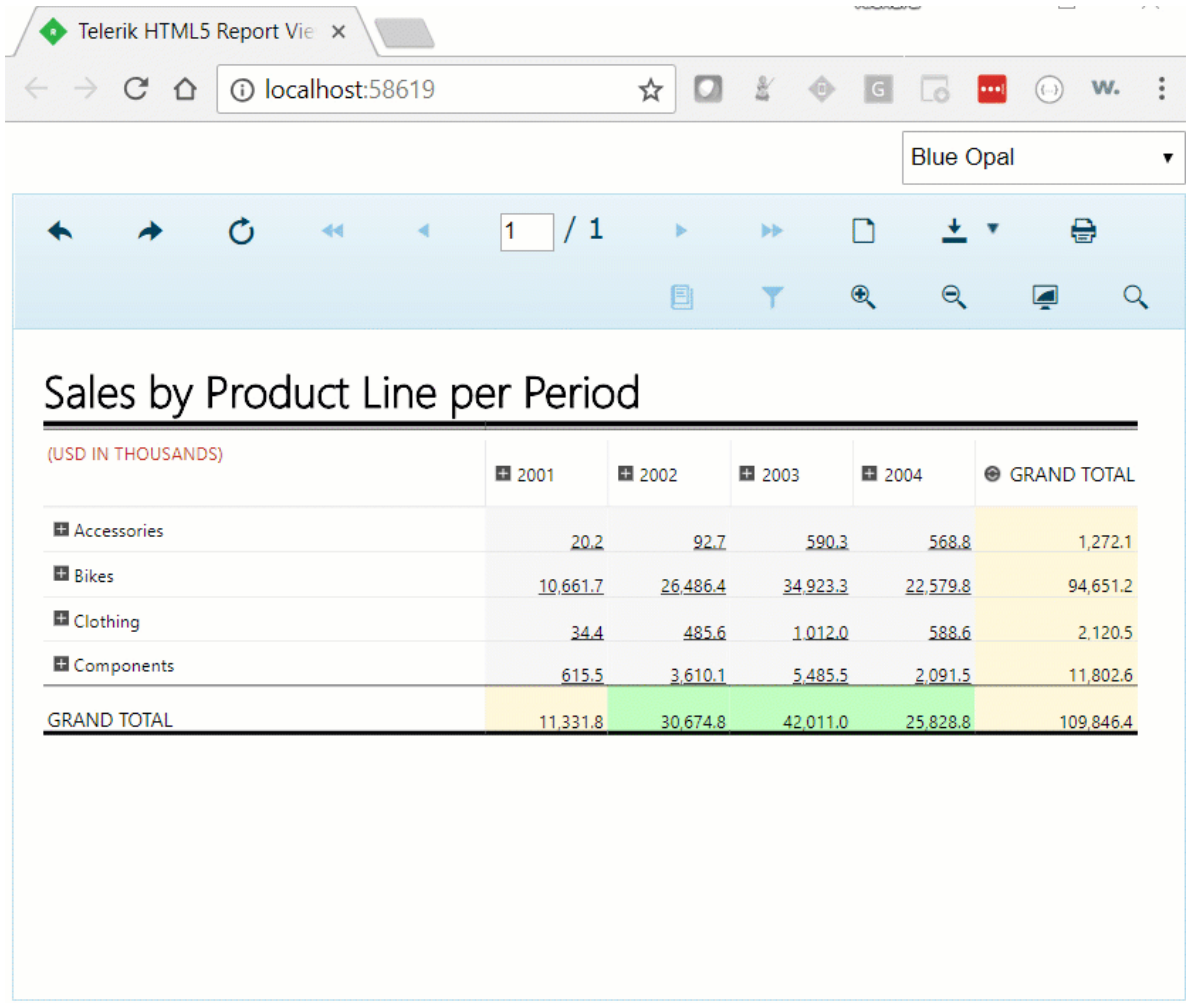
Styling Reports

- CSS-like styling mechanism
  - Quick, detailed visual customization of all properties of a report item
  - Copy and reuse styles across report items
- Export stylesheets and apply them
- Format Builder
  - Set the proper formats for dates, numbers and currency
- Conditional formatting
  - Powerful reporting expression engine with data conditions
  - Control text color, font and background



Interactive Report Features

- Parameterized Reports
  - Filter data based on built in parameters
- Drill-Down Report Action
  - Toggle visibility of report items
  - Collapse/expand column or row collections
- Drill-Through Report Action
  - Link to another report
- Customizable Tooltips
- Table of Contents
  - Bookmarks
- Hyperlinks
  - Open URL in Browser



Progress®
NEXT19

Publishing Telerik Reports

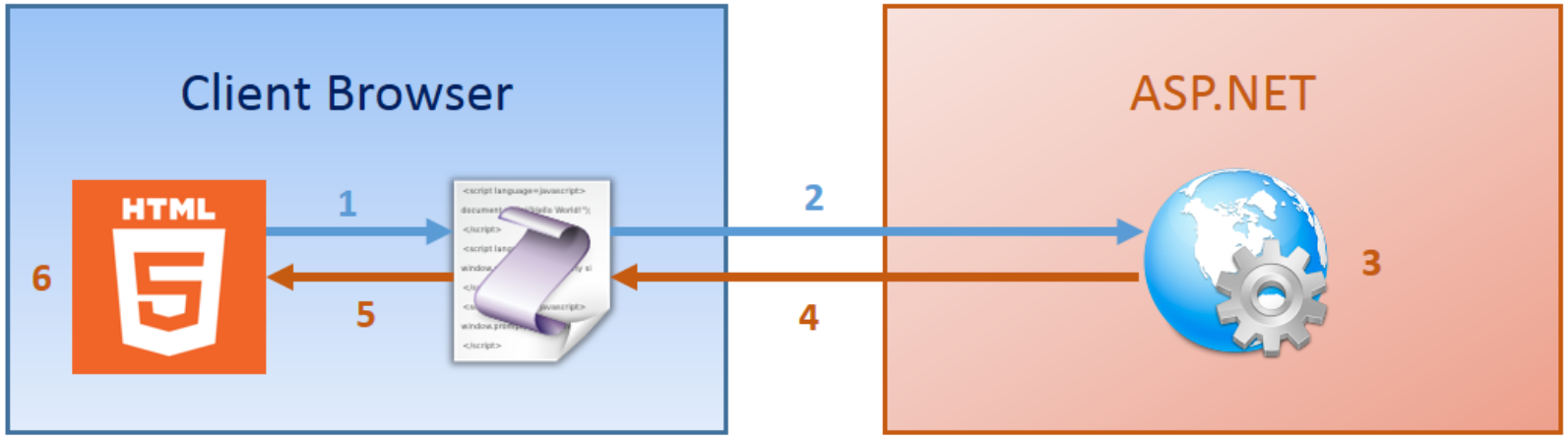
Telerik Reporting REST Service

Telerik Reporting REST Service

- A set of services that allows report generation from client applications
- Serves requests from an embedded HTML5 Report Viewer to a backend server app where Reporting Engine and report definitions reside
- The service provides a simple Application Programming Interface (API) to our Telerik Report Engine using HTTP requests
  - The client of the service requests a report by a JSON object called client report source.
  - The service then resolves this to a .NET report source. To do that it uses a REST Service Report Resolver.
  - Serves the Telerik Report to the HTML5 Report Viewer widget as a web accessible resource
- Primary implementation uses the **ASP.NET WebAPI Framework**, **but options are available.**

Telerik Reporting REST Service

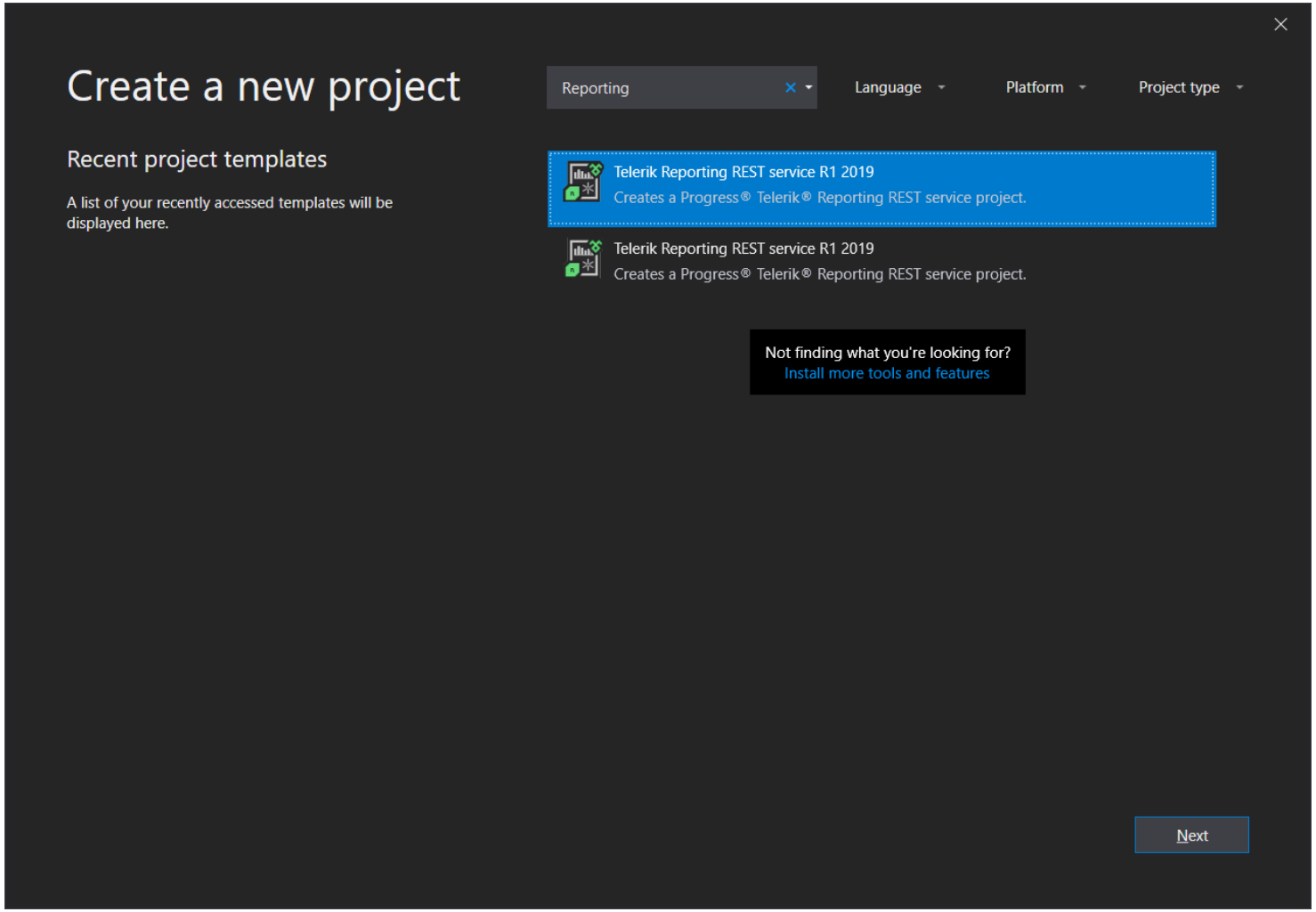
How it works…



1. HTML5 based web page calls a JavaScript method to request a report
2. JavaScript method issues a RESTful call to the Reporting ASP.NET Web API endpoint
3. ASP.NET Web API controller leverages Report Rendering Engine to render the requested report
4. ASP.NET Web API controller returns the Report artifacts to the calling JavaScript method
5. JavaScript method sends report artifacts to the HTML5 page for rendering
6. Client browser renders the HTML5 report

Configuring a Telerik REST Service

- “New -> Project -> Reporting -> Telerik Reporting REST Service”
- Creates .NET Server App with Telerik Reporting Engine and “ReportsController.cs”
- Sets default path to report definitions to a “Reports” folder in the app
- Adds resolver for trdx/trdp report definitions and a fallback resolver for .NET class reports

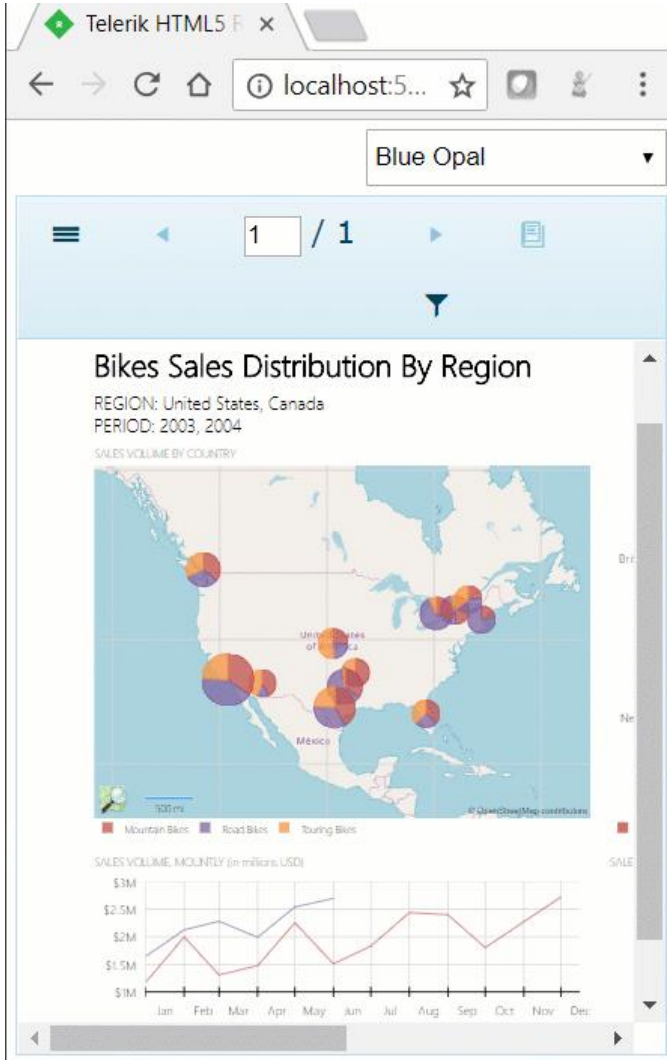


* Wizard based approach works for full .NET Framework apps only. .NET Core apps require manual configuration

HTML5 Report Viewer

HTML5 Report Viewer

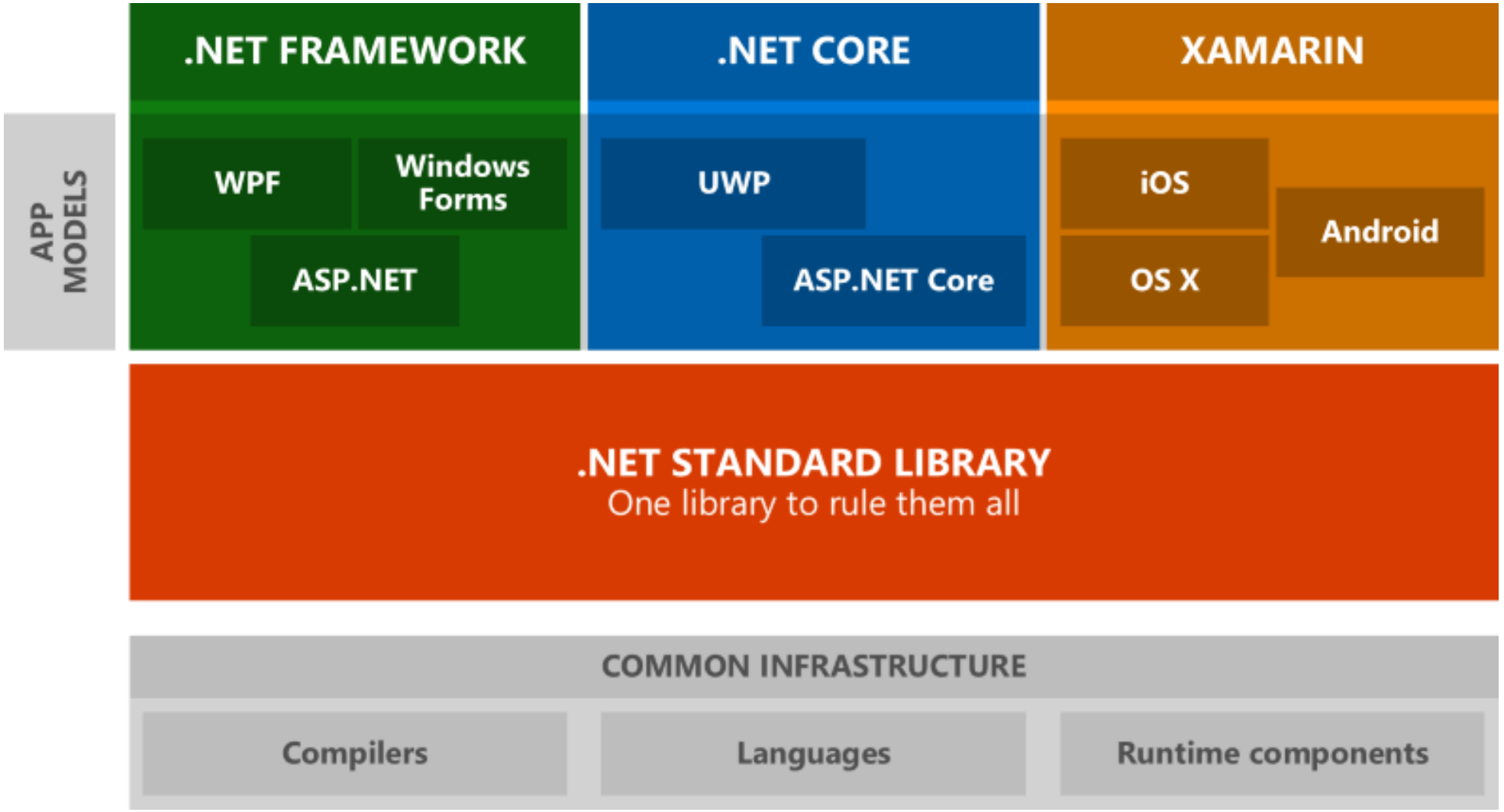
- Pure HTML5/JavaScript/CSS3 jQuery-based component that enables displaying reports in an HTML page
  - Fully customizable with included HTML templates
  - Extensive styling capabilities and theming options
  - Component wrapper for Angular (new)
- Serve reports in all leading modern desktop and mobile browsers
  - HTML5 compliant desktop web browsers
  - Touch-enabled mobile devices running iOS, Android, Windows Phone and WinRT
- Baked-in responsive layout adapts itself to the device display for mobile-friendly viewing experience



“The Final Frontier”

.NET Standard 2.0 and the Evolution of .NET Core

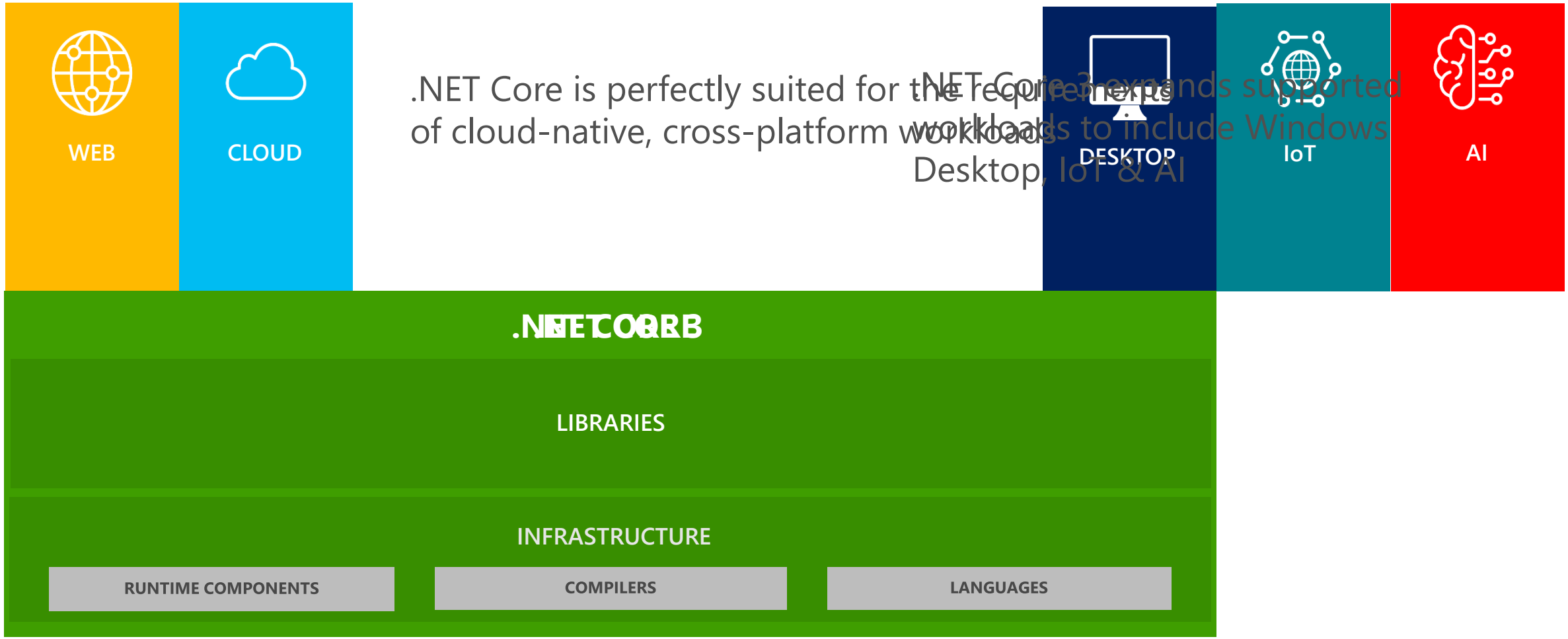
.NET Standard 2.0 Library



.NET Standard 2.0 Library

- .NET Standard is a set of fundamental APIs that are available in all .NET implementations
- Designed as a formal specification, it determines a base set of functionalities for the .NET ecosystem.
- It is supported in .NET Core 2.0+, in the .NET Framework 4.6.1+, and in Visual Studio 15.3 and newer
- This unifies the .NET implementations and prevents future fragmentation
- It replaces Portable Class Libraries (PCLs) as the tool for building .NET libraries that work everywhere.
- Spoiler! Telerik Reporting now provides a version built on .NET Standard 2.0

.NET Core 3



The Evolution of .NET Core

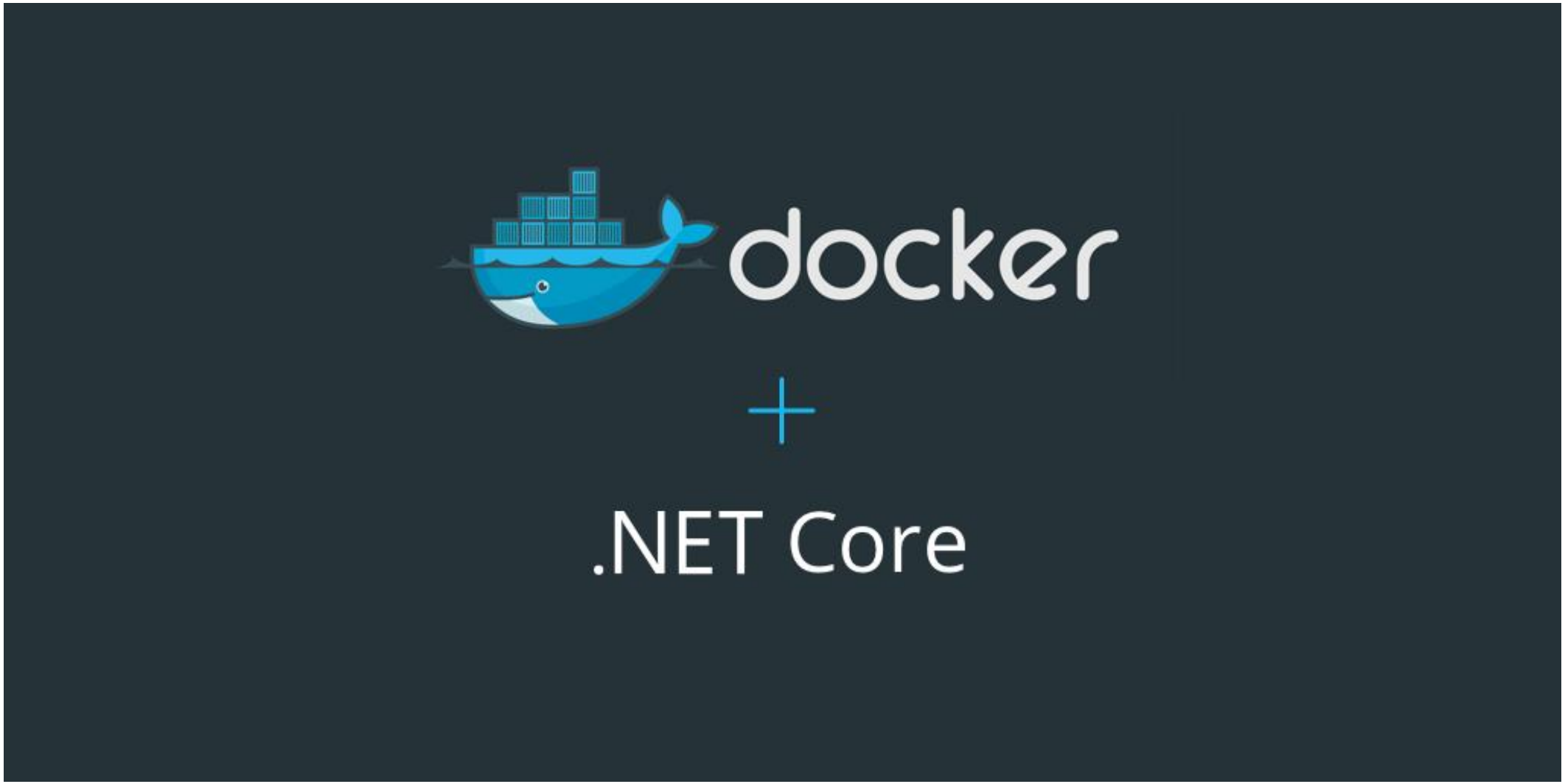
- .NET Core is a cross-platform, open source implementation of .NET guided by the .NET Foundation (Progress was an early corporate sponsor!)
- Microsoft is *"highly recommending"* that all Greenfield development be done using .NET Core
  - Microsoft is just going to be making highly compatible targeted improvements to .NET Framework after version 4
- Brownfield apps can be converted to run in .NET Core
  - Take all of those methods which are .NET Framework specific, wrap them in a micro service (or a number of micro services) and put them somewhere (Dependency inversion)
  - Re-implement your application stack in .NET Core, making it smaller and faster
  - Host them on less expensive Linux or Windows servers
- .NET Core 3 will add support for development of desktop application software, artificial intelligence/machine learning and IoT apps

New!
Telerik Reporting REST Service
with
Cross-Platform Support

.NET Standard 2.0 → Core 2.0+ Support

- Support Introduced with R1 2019 release (January)
- New! .NET Standard 2.0 versions of Telerik Reporting assemblies
  - Available via NuGet packages from our Telerik NuGet private feed
  - Or add assembly references to local .dll files in **\Bin\netstandard2.0** installation folder
  - The external dependencies are resolved through NuGet package dependencies in order to provide better dependency management and code portability
- Deploy the Telerik REST Service (Report Library & Reporting Engine) in .NET Core
  - Run on Linux, macOS and Windows machines
  - **Can be implemented in a Docker Container**

Demo – Telerik Reporting in .NET Core 2



.NET Core 2.0+ Reporting - Requirements

- On Windows, our Reporting Engine still relies on the GDI+ library for rendering
  - It provides the fastest and most convenient way to process text and images
  - The .NET Core runtime manages to resolve the GDI+ calls natively
- Linux and macOS deployments require the install of an additional **libgdiplus** library
  - Mono implementation of GDI+ API for non-Windows operating systems

.NET Core 2.0+ Reporting – Supported Features

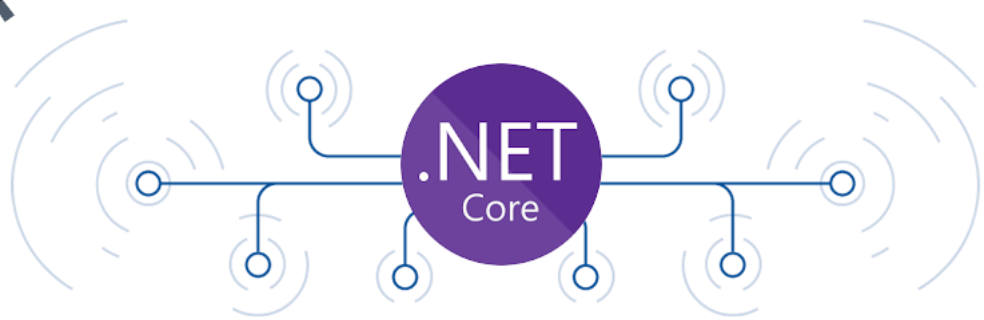
- All major Telerik Reporting functionalities work in .NET Core CLI and web apps
  - All rendering extensions except MHTML and XPS are supported on Windows.
  - HTML, PDF and OpenXML-based renderings are supported on Linux/macOS.
  - All report items except the obsolete Chart item are supported.
- The supported report definition types:
  - .trdx and .trdp (XML based) created using Standalone Report Designer
  - .cs and .vb report classes, created in Visual Studio Designer (limitations apply)
  - KB to assist with using reports from an existing .NET Framework 4+ report library in a .NET Core application

Progress®
NEXT19

Displaying Reports in .NET Core 3.0 Desktop Apps

.NET Core 3.0 Desktop Integration

Telerik UI for Winforms / WPF Both Now Run in .NET Core 3.0



Telerik Reporting in .NET Core 3.0 WinForms Apps

- Prerequisites:
  - Visual Studio 2019 or newer
  - .NET Core SDK 3 Preview 3 or newer
  - Windows Forms App (.NET Core) project
- Add the following assembly references (available in Telerik Reporting installation Bin directory) or NuGet package references (available in Telerik Private NuGet Repository):
  - Telerik.Reporting (located in Bin\netstandard2.0)
  - Telerik.ReportViewer.WinForms (located in Bin\netcoreapp3.0)
- Instantiate UriReportSource and set the proper UriReportSource.Uri relative path
- Instantiate and setup the WinForms Report Viewer

Telerik Reporting in .NET Core 3.0 WPF Apps

- Prerequisites:
  - Visual Studio 2019 or newer
  - .NET Core SDK 3 Preview 3 or newer
  - WPF App (.NET Core) project
- Add the following assembly references (available in Telerik Reporting installation Bin directory) or NuGet package references (available in Telerik Private NuGet Repository):
  - Telerik.Reporting (located in Bin\netstandard2.0)
  - Telerik.ReportViewer.Wpf (located in Bin\netcoreapp3.0)
  - Telerik.ReportViewer.Wpf.Themes (located in Bin\netcoreapp3.0)
  - Telerik.Windows.Controls (located in Bin\WpfViewerDependencies)
  - Telerik.Windows.Controls.Input (located in Bin\WpfViewerDependencies)
  - Telerik.Windows.Controls.Navigation (located in Bin\WpfViewerDependencies)
  - Telerik.Windows.Data (located in Bin\WpfViewerDependencies)

Telerik Reporting in .NET Core 3.0 WPF Apps

- Merge the required resources to the application in the App.xaml

- Add the following namespaces to the target Window declaration:
  - xmlns:tr="http://schemas.telerik.com/wpf"
  - xmlns:telerikControls="clr-namespace:Telerik.Windows.Controls;assembly=Telerik.Windows.Controls“
  - xmlns:telerikReporting="clr-namespace:Telerik.Reporting;assembly=Telerik.Reporting"

- Add the WPF Report Viewer declaration to the target Window and set the proper UriReportSource.Uri relative path

Thank You!

Richard Zaslaw

richard.zaslaw@progress.com

https://www.linkedin.com/in/richard-zaslaw-telerik/

You might also like to attend

Data Integration: The REST of the Story

Tony Lavinio

Room Congressional A @ 3:45 PM


Progress®
NEXT19