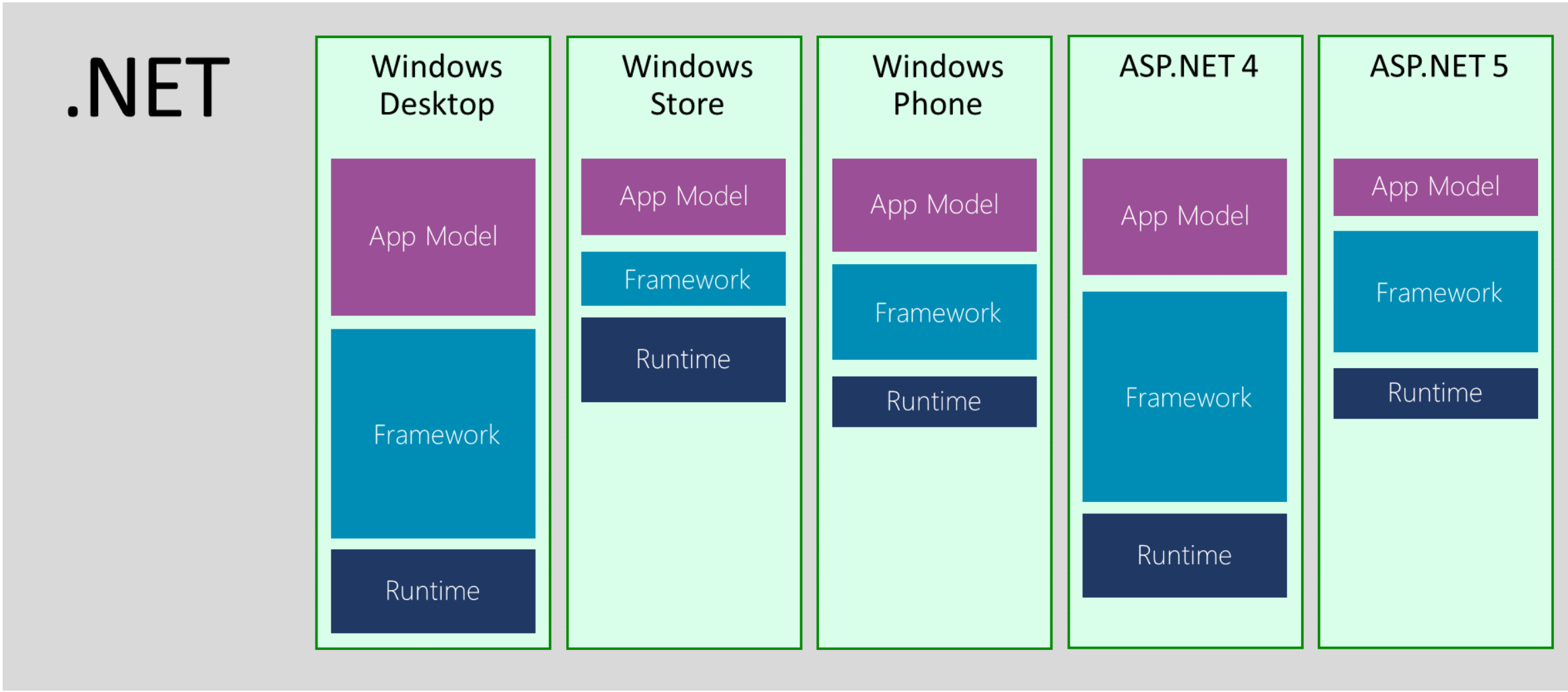
Kentico

Create a .NET Core app

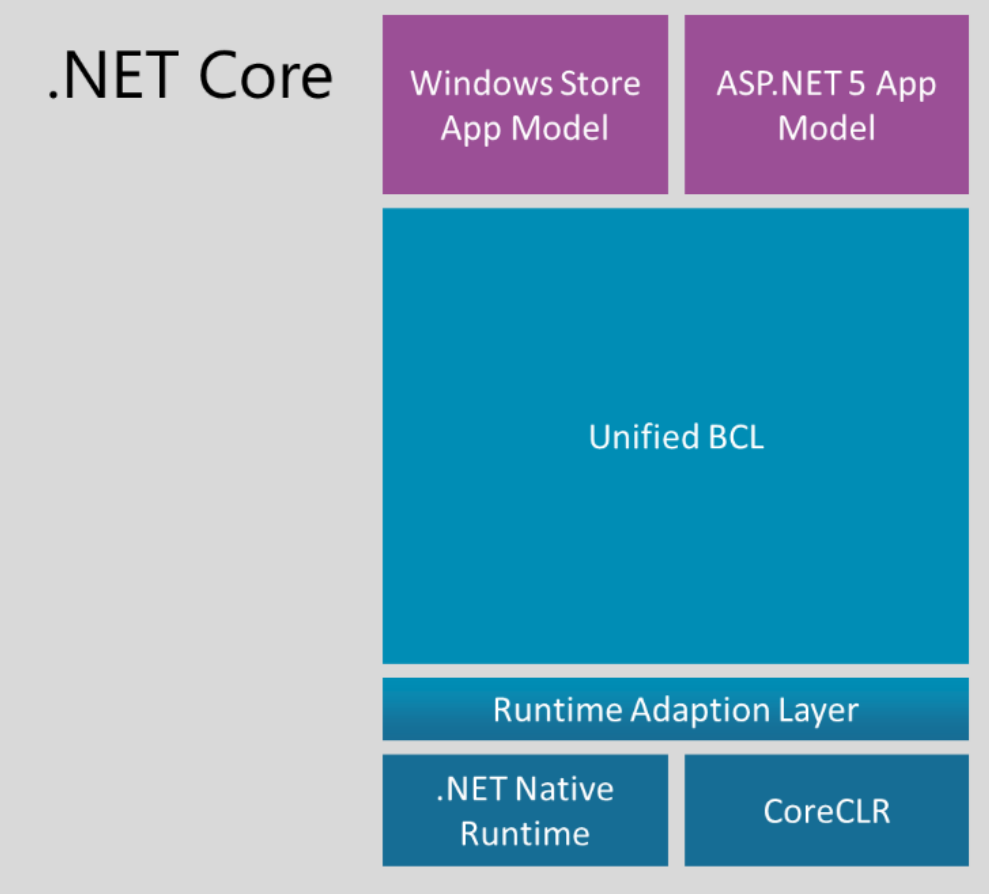
Tomas Hruby

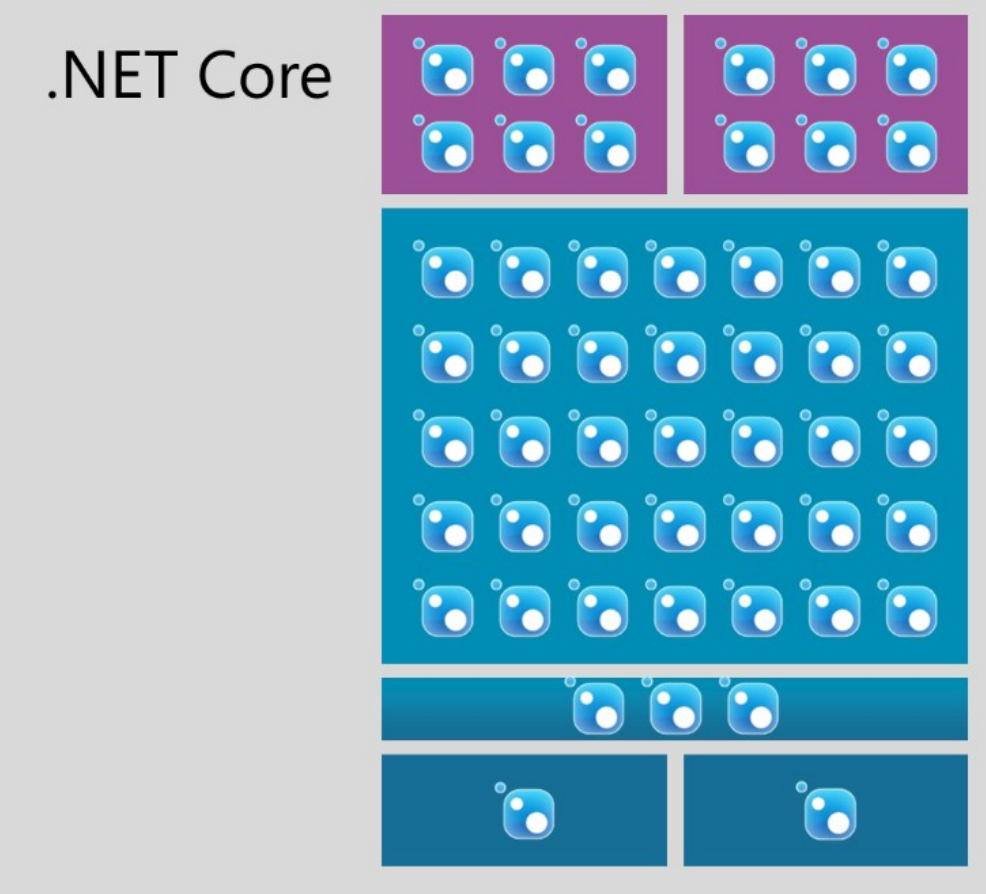
2016

Before


.NET
Windows Desktop
App Model
Framework
Runtime
Windows Store
App Model
Framework
Runtime
Windows Phone
App Model
Framework
Runtime
ASP.NET 4
App Model
Framework
Runtime
ASP.NET 5
App Model
Framework
Runtime

Now



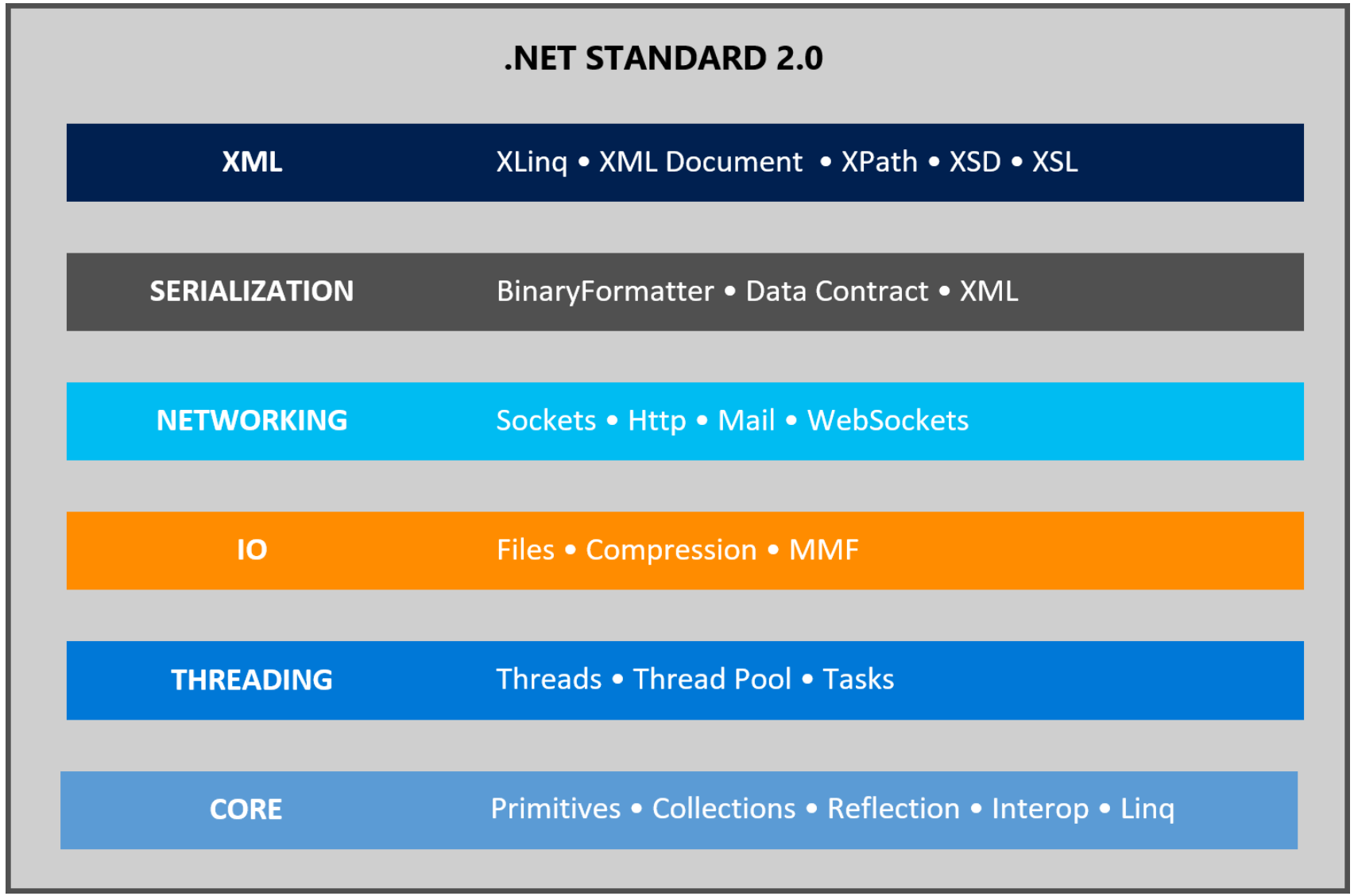


https://github.com/dotnet/core
https://github.com/dotnet/coreclr

.NET Standard

.NET Platform	.NET Standard							
	1.0	**1.1**	**1.2**	**1.3**	**1.4**	**1.5**	**1.6**	**2.0**
.NET Core	→	→	→	→	→	→	1.0	vNext
.NET Framework	→	4.5	4.5.1	4.6	4.6.1	4.6.2	vNext	4.6.1
Xamarin.iOS	→	→	→	→	→	→	→	vNext
Xamarin.Android	→	→	→	→	→	→	→	vNext
Universal Windows Platform	→	→	→	→	10.0	→	→	vNext
Windows	→	8.0	8.1					
Windows Phone	→	→	8.1					
Windows Phone Silverlight	8.0							

.NET Standard



Kentico

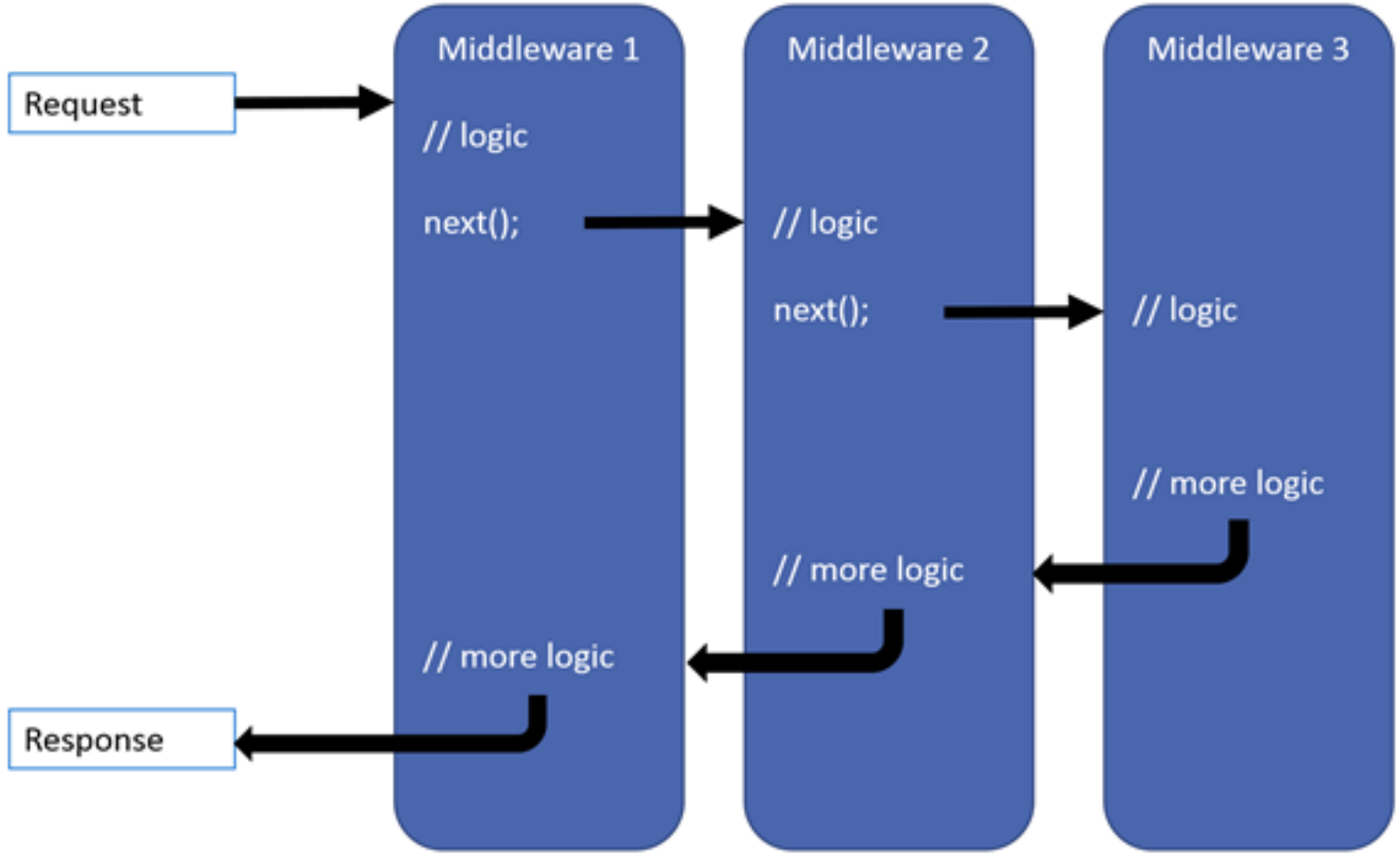
It’s Demo time…

https://docs.asp.net/en/latest/fundamentals/index.html

Kentico

Adding middlewares

ASP.NET middlewares



Kentico

It’s Demo time…

Kentico

Developing a full-blown web app

Dependency Injection

- **Is available everywhere — Startup, Middleware, Services**
- **Enables you to modularize your application**
- **https://docs.asp.net/en/latest/fundamentals/dependency-injection.html**

App Configuration

- **Configuration is build on the project startup (Startup.cs)**
- **Combine configuration from multiple sources**
- **https://docs.asp.net/en/latest/fundamentals/configuration.html**

Logging

- Access logger everywhere while maintaining IOC
- Switch loggers application-wise (Startup.cs)
- Specify level of details for every namespace
- https://docs.asp.net/en/latest/fundamentals/logging.html

Routing

- Map URI to Controller
- URI – controller, action, query

Unit tests

- **xUnit**
  - https://xunit.github.io/docs/getting-started-dotnet-core.html
- **NSubstitute**
  - http://nsubstitute.github.io/help/getting-started/

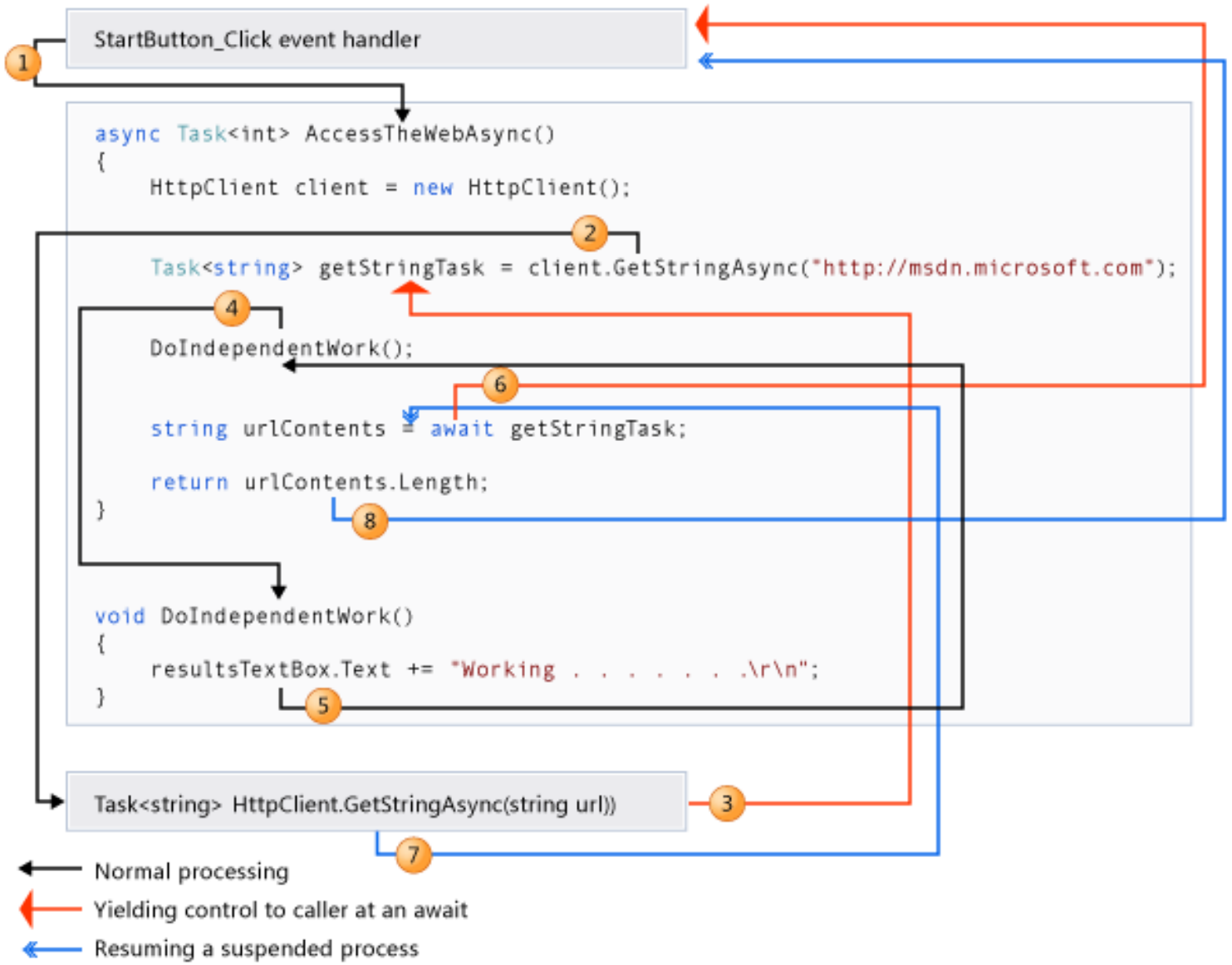
Kentico

Asynchronous programming

Tomas Hruby

2015

Kentico


StartButton_Click event handler
1
async Task<int> AccessTheWebAsync()
{
HttpClient client = new HttpClient();
2
Task<string> getStringTask = client.GetStringAsync("http://msdn.microsoft.com");
4
DoIndependentWork();
6
string urlContents = await getStringTask;
return urlContents.Length;
}
8
void DoIndependentWork()
{
resultsTextBox.Text += "Working . . . . . .\r\n";
}
5
Task<string> HttpClient.GetStringAsync(string url))
3
7
Normal processing
Yielding control to caller at an await
Resuming a suspended process

Async-Await

- Put async-await only across **the whole** application
- **Don't mix** manipulation with threads with async-await
  - Blocking context threads causes deadlocks
  - To use async code from normal method use **Task.Run**
- Method can continue in a **different thread** after awaiting
  - Don't depend on the thread context
  - HTTP context copies context information to new thread from the thread pool

https://msdn.microsoft.com/en-us/library/hh191443.aspx
http://blog.stephencleary.com/2012/07/dont-block-on-async-code.html

Kentico

REST API design

Basics revisited

- Stateless HTTP communication (Request-Response)
- URL identifies an object
  - ~/api/users/ngk67
  - ~/api/users/ngk67/roles
  - Use plural for endpoints (/api/users, /api/projects)
- Method identifies action (GET, POST, PUT, PATCH, DELETE)
  - GET ~/api/users (get all users)
  - DELETE ~/api/users/ngld456 (delete user with given ID)
- Request body represents the object identified by URI
  - POST ~/api/users (object in request body is inserted as user)
- Server answers with HTTP Status Code and response body containing requested object(s)

HTTP Status Codes

Method	URL pattern	Statuses	(Notes)
GET	/api/users	200	
GET	/api/users/{id}	200, 404	
POST	/api/users	201, 400	201 – Created
PUT	/api/users/{id}	200, 404, 400	400 – Bad request
DELETE	/api/users/{id}	204, 404	204 – No content

401 – Unauthorized
500 – Internal Server Error
429 – Too many requests (response should contain time of renewal)

http://www.restapitutorial.com/httpstatuscodes.html

Filtering & Paging

- Filtering using query parameters
  - GET ~/api/students?semester=spring
- Filtering using query objects
  - GET ~/api/students?filter={“subjects”=[“PE”,“maths”],“age”=18}}
  - REST calls remain the same even if new filter is added (forward compatible)
  - Easy to write by hand in browser URL bar
- Paging request
  - GET ~/api/students?pagesize=10&page=3
- Paging responses
  - Pagination info in header (X-Total-Count, X-Page-Size, X-Current-Page)
  - Pagination envelope
    - { “total”: 40, “pageSize”: 10, “page”: 1, “data”: [...] }
  - Cursor based pagination (link to next page)

Versioning

- Using URL prefixes
  - ~/api/v1/users
- Using HTTP headers
  - X-API-Version: 7.1.3

Special actions

- Don’t use special query parameters (as flags). Use special endpoints
  - ~/api/students/:student_id/disable
  - ~/api/users/resetpassword