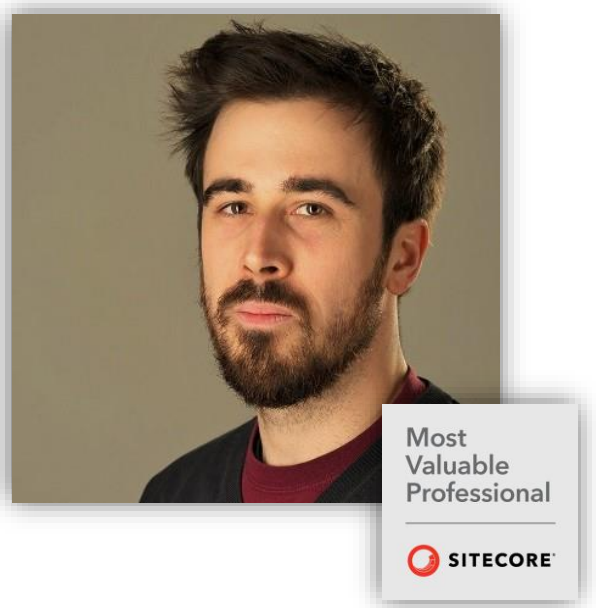
DISCOVERING SITECORE 10 ASP.NET CORE SDK

by Tamás Tárnok

About me

- Senior Software Developer at ALLWIN Informatics
- Working on Sitecore projects since 2014
- Sitecore Technology MVP since 2018
- Blog: https://trnktms.com
- Twitter: https://twitter.com/trnktms



Agenda

1. Architecture overview
2. How to start with ASP.NET Core SDK
3. Basic features
4. DEMO
5. Pros/Cons

Layout Service – rule them all

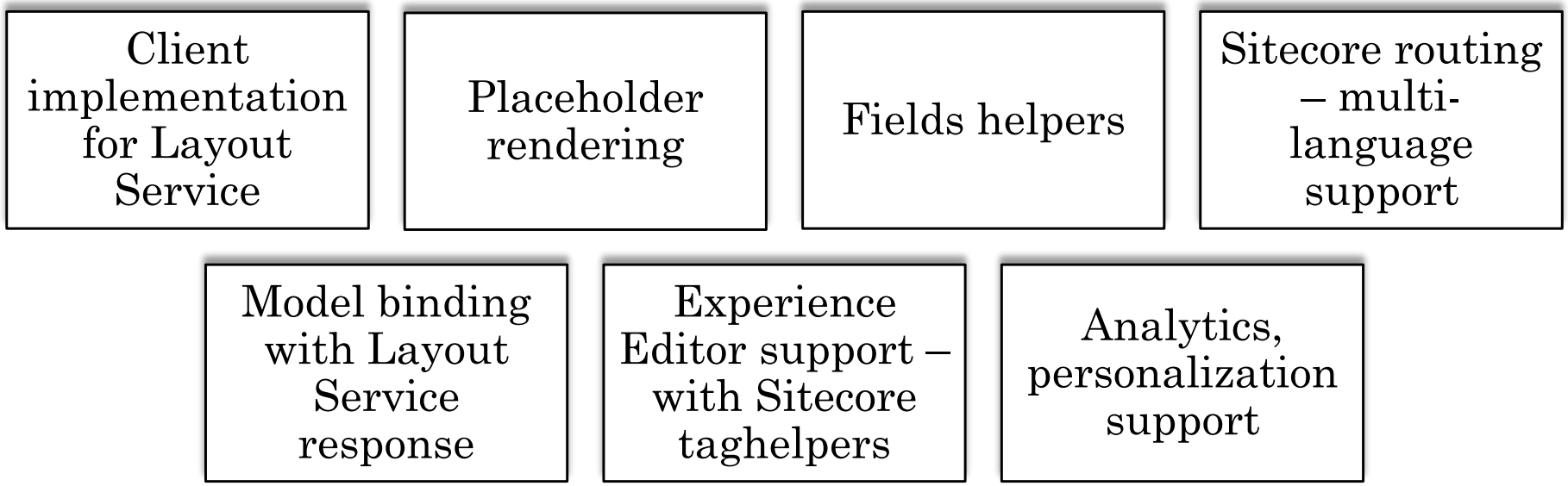
- Layout Service introduced with the release of JSS
- It's service which represents a page in JSON instead of HTML
- This gives the possibility to implement any SDK around it – platform independent

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Example<title>
        <link rel="stylesheet" href="sty
    </head>
    <body>
        <h1>
            <a href="/">Header</a>
        </h1>
        <nav>
            <a href="one/">One</a>
            <a href="two/">Two</a>
            <a href="three/">Three</a>
        </nav>
```

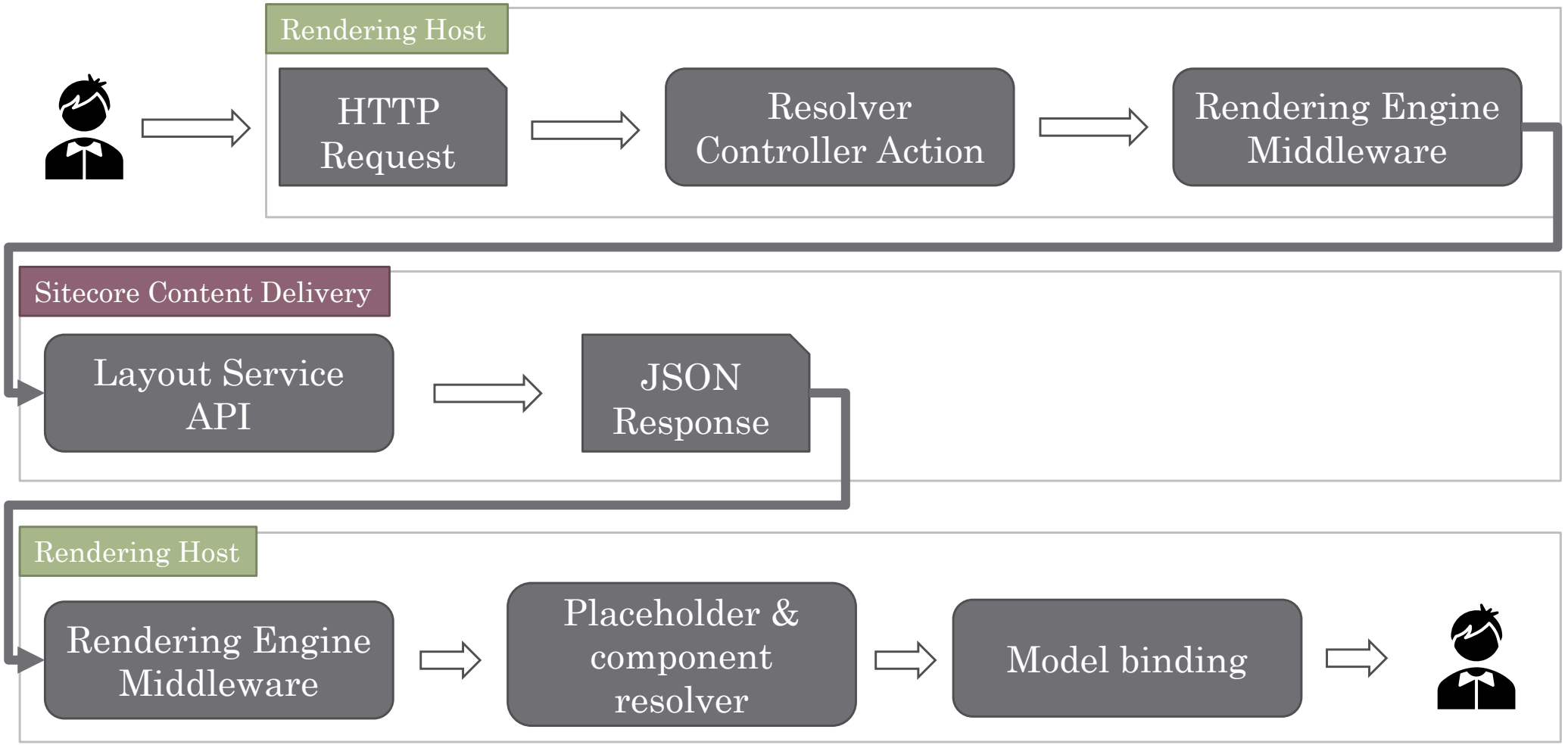
```json
{
    "id": "1242160000000072038",
    "description": "3",
    "website": "3",
    "numberOfEmployees": "3",
    "phone": "3",
    "name": "account3",
    "shippingAddress": {
        "country": "3",
        "stateOrProvidence": "3",
        "city": "3",
        "postalCode": "3",
        "street1": "3"
    },
    "billingAddress": {
        "country": "3",
        "stateOrProvidence": "3",
        "city": "3",
        "postalCode": "3",
        "street1": "3"
    }
}
```


.NET
Core

Sitecore ASP.NET Core SDK



The main process of a request



How to use/install

- **On the Sitecore instance**

Using prebuilt docker image from Sitecore

OR

Install the Sitecore Headless Rendering package

- **On the Rendering Host instance**

```
dotnet new sitecore.aspnet.gettingstarted -n MyProject
```

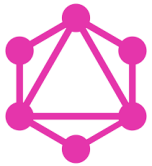
OR

~~Installing the necessary Nuget packages one-by-one~~

Contents resolver

- Implementation should be in the Sitecore project
- It is responsible for the JSON response for a rendering
- Sitecore has the following resolvers by default:
  - Context Item Children Resolver
  - Context Item Resolver
  - Datasource Item Children Resolver
  - Datasource Resolver
  - Folder Filter Resolver
  - Sitecore Forms Resolver
- Will be needed to implement our own to provide the data we need in the Rendering Host
- Chained Contents Resolver exists, implemented by Nick Hills – go to GitHub

GraphQL



- GraphQL is a query language to ask what you need from an endpoint
- Instead of Contents Resolver, possible to use GraphQL to put the data together for a component
- Until now Sitecore does not support it OOTB in the Rendering Host
- But Göran Halvarsson published a nice blogpost how you can make it work – go to the post

Model binding

- The concept is similar to Glass Mapper
- Main difference – models in the rendering host should be in sync with JSON and these are not 1-1 relation between template and model
- Possibilities to bind your models in the Rendering Host:

Name	Code
ModelBoundView	`.AddModelBoundView<MyModel>("MyView")`
ViewComponents	`await _viewModelBinder.Bind<MyModel>(this.ViewContext)`

Rendering Host caching

- Caching is always important to use in Sitecore projects – HTML caching, item caching, etc.
- The same caching mechanism is still used on the Sitecore instances
- The JSON response can cached and works like the good old HTML caching
- At the moment Sitecore has no OOTB caching in the ASP.NET Core SDK, it will come soon
- Until then, I implemented a PoC how output caching could be implemented – go to the post

Translation API

- No translation API client side implementation yet in the SDK out-of-the-box
- But the backend implementation is ready - /sitecore/api/jss/dictionary/<site>/<language>/?sc_apikey=<apikey>
- Currently there is a cache in it this API which causes a not intermediate update after publish #CS0214257
- Until then, I implemented a PoC how this can be implemented in the Rendering Host – go to the post

DEMO

Pros

- Using new technology – HR and devs are happy
- Better scaling
- Rendering Host is more light weight than a full Sitecore CD
- Faster debugging – just use F5 in Visual Studio
- Faster deployment – no need to deploy on a Sitecore instance
- Platform independent hosting

Cons

- Infrastructure a bit more complex because of the extra Rendering Host role
- Less shared knowledge than with Sitecore MVC **yet**
- No official output caching **yet**

Sources

- How to Develop with Sitecore ASP.NET Core Rendering SDK by Nick Wesselman
- Sitecore Headless Development
- #headless channel on Sitecore Chat
- #headless tag on Sitecore StackExchange

Thank you!

Questions?