Custom MVC apps:
"Adding custom functionality for browsers"

Pre-Conference Session, IBUC 2018

Statistics Netherlands

A brief history [1]

- Blaise 5.0.x:
  - Asp.NET on webserver
  - Business logic on webserver and webc lient
  - Controls 'lived' on the webserver
  - Every action resulted in a complete postback of the webpage
  - Customization takes place on webserver (C#) and webclient (HTML, Javascript and CSS)

Microsoft

ASP.NET

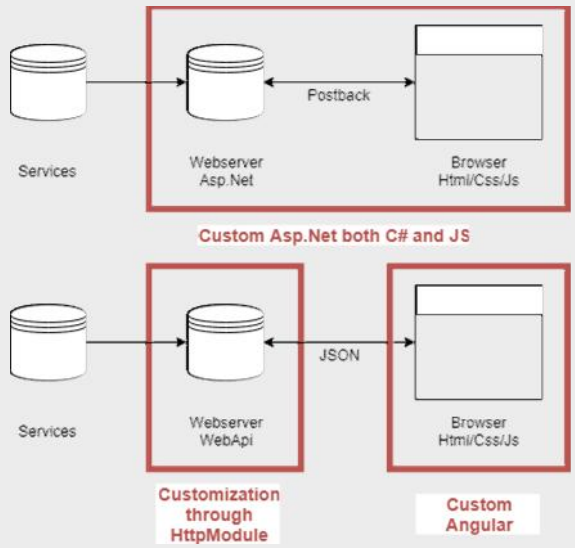
A brief history [2]

- Blaise 5.2.0+:
  - Web API on webserver, only pass-through of data
  - No business logic on webserver, all logic in webclient
  - Controls 'live' in the webclient
  - Every action results in a JSON postback
  - Runtime customization only needed in webclient
  - Business logic customization at domain-level accessible through httpmodules (Authorization, Authentication, Redirecting)

A brief history [3]

- Blaise 5.2.0:
  - Web runtime used AngularJS (1.x)
  - Primary languages Javascript, CSS and HTML
  - We never got to customization
  - Only the 'old' apsx runtime was customizable
- Blaise 5.2.5+:
  - Start using the new Angular2+
  - Primary languages Typescript, SCSS and HTML
  - Use of Typescript instead of Javascript which made our runtime customizable, strongly-typed and testeable
  - Use of SCSS made our CSS cleaner

Schematics



Angular runtime

- Every part of the application we try to make customizable (components, directives, pipes and services)
- Documentation intensively in development
- Blaise 5.2.5 based on Angular 4.2.4
- Blaise 5.3.0 based on Angular 5.1.2
- Blaise 5.4.x based on Angular 5.2.x
- Blaise 5.5.x will be based on Angular 7.x / CLI
- Blaise 5.6.x propably based on Angular 8.x / CLI
- We will try to keep up with Google

Why do you need customization [1]

1. Need for custom layout
   - When our templates don't suffice your needs
   - E.g. use Google Maps for storing coördinates in a field of type String
   - E.g. when you want to add animations to your survey
2. Need for custom events or actions
   - When you want to act on or react to events we don't provide
   - E.g. you want to change te format of a certain value before sending it to the DataEntryService

Why do you need customization [2]

3. Need to access external API's
   - Lookup external API's instead of Blaise resources
4. Authentication, Authorization
   - Make sure the client is who he says he is
   - Make sure the client is authorized to access the survey
   - This is beyond the scope of this presentation

What do we provide

- General Application
  - An empty shell as a starter boilerplate for your development
- Technical documentation for the current release on https://help.blaise.com/angular
- Technical documentation for previous releases on https://help.blaise.com/angular/[version] (only 5.2.5+) e.g. https://help.blaise.com/angular/5.4.1

What do you need to develop

- Software you need
  - NodeJS >= v8.x
  - NPM >= v5.x
  - An IDE like:
    - Visual Studio Code + extensions
    - Visual Studio
- Knowledge you need
  - Blaise 5
  - Angular
  - Typescript / Javascript

Demo

Anatomy of the General Application [1]

- Files
  - Package.json
    - General configuration variables
    - Dependencies
      - Fix the path to @blaise/core here
  - README.md
    - General information on what to do
  - Tsconfig.json
    - TypeScript compiler options
  - Webpack.config.js
    - Bundler options

Anatomy of the General Application [2]

- Directories
  - Config
    - Various helper/config files
  - Datamodel
    - A place to hold your datamodel (not compulsory)
  - Debug
    - Where the debug version of the application goes
  - Dist
    - Where the production version of the application goes
  - Src
    - Where the source of your custom application go

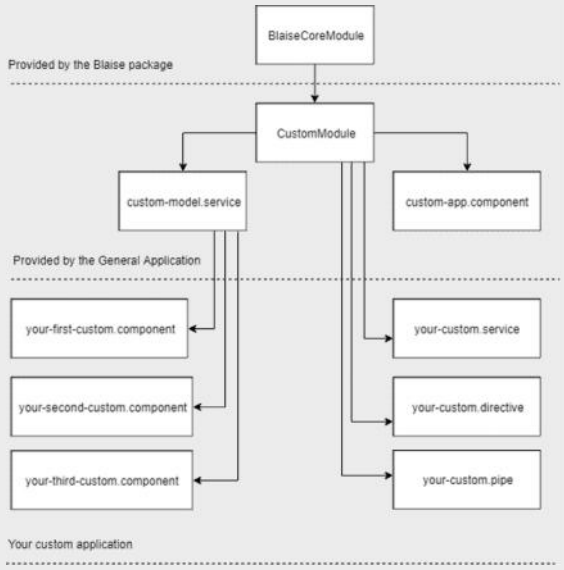
Anatomy of the General Application [3]

- src
  - Files
    - index.html
    - main.ts
    - polyfills.ts
    - vendor.ts
  - Directories
    - app
    - assets
    - typedefinitions

Anatomy of the General Application [4]

- app
  - Files
    - custom.module.ts
      - makes your components, providers etc. known to the system
  - Directories
    - components
      - custom-app.component.ts/.scss/.html
        - containing controller
      - your own component
    - constants
    - directives
    - pipes
    - services
      - custom-model.service.ts
        - instantiate your custom control(s) here

How does it all fit together



Demo

Summary

- As of Blaise 5.2.0 we use the Angular Framework for our web runtime
- As of Blaise 5.2.5 we use Angular 2+ Framework for our web runtime, Typescript, Testable, Extensible
- As of Blaise 5.5 we shift to Angular CLI, customization easier, ahead-of-time compiling
- We provide hooks to extend or customize this runtime
- With every new major release of Blaise we try to update the Angular runtime
- For every version the help is online available

Ready to start customizing?

We are ready!