JavaCro'16

QUALITY. PRODUCTIVITY. INNOVATION.

endava

Modularize your Angular application in two weeks

endava.com

Modularize your Angular application in two weeks

- Problems / Motivation
- Monolithic V.S. Modularized Apps
- How to implement modularization?
- Current situation with modularization
- New approaches
- Demo
- Tools that we must use

Modularization

endava

Restaurant approach

- Customer (Client) orders a meal
- Waiter (PM) takes the order
- Chef (Developer) prepares the meal
- Sauce chef decorator (Designer)
  - Make it look beautiful
- Tester tastes the meal
- At the end Waiter serves the meal

Modularization

Problems that we want to solve and what we want to achieve

- Reuse the code
- Ease of maintenance / testing
  - Prepare for continuous integration
- Extensibility / new features
- Increase development speed
- Speed up the build of similar applications
  - Reusable modules for other applications (regardless of domain)

Modularization

endava

Same goals but with different approach

Reuse the code	Other orders can have same starter?
Ease maintenance	Not all of us want chicken with curry
Testing	Ingredient can be tasted, but also the meal
Extensibility / new features	Client can order multiple meals
Increase development speed	Recipes how to prepare meals

Solution is simple: **Use Modules**! But how?

Monolith applications

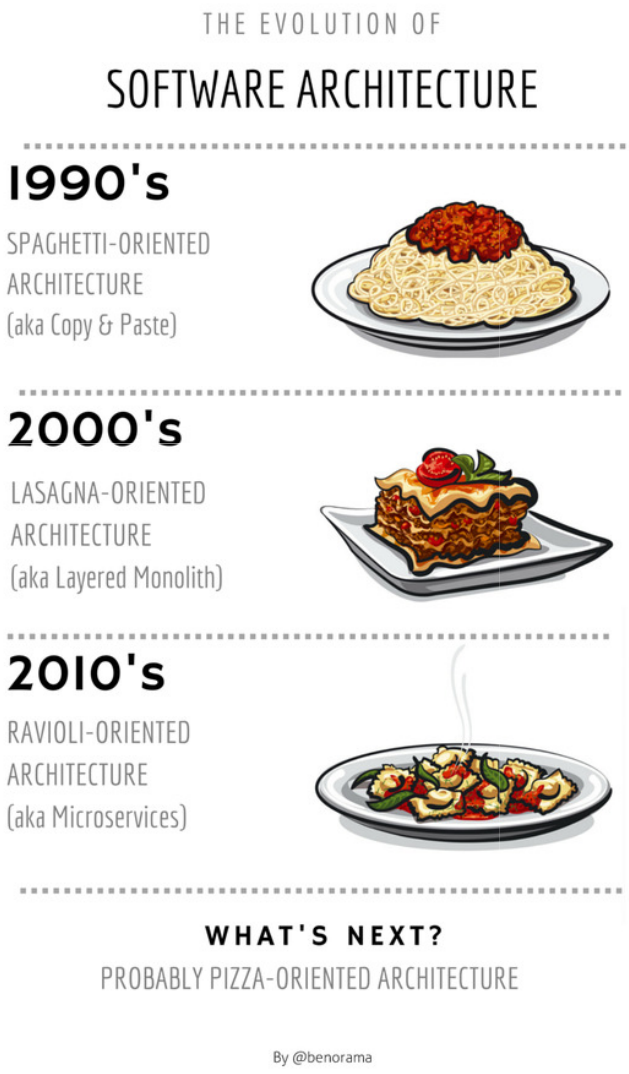
endava

Monolith Apps

- What is monolith app?
- Is angular monolith?
- What is module, or what should be?
- How to distribute a module?

DEMO

Evolution of software architecture

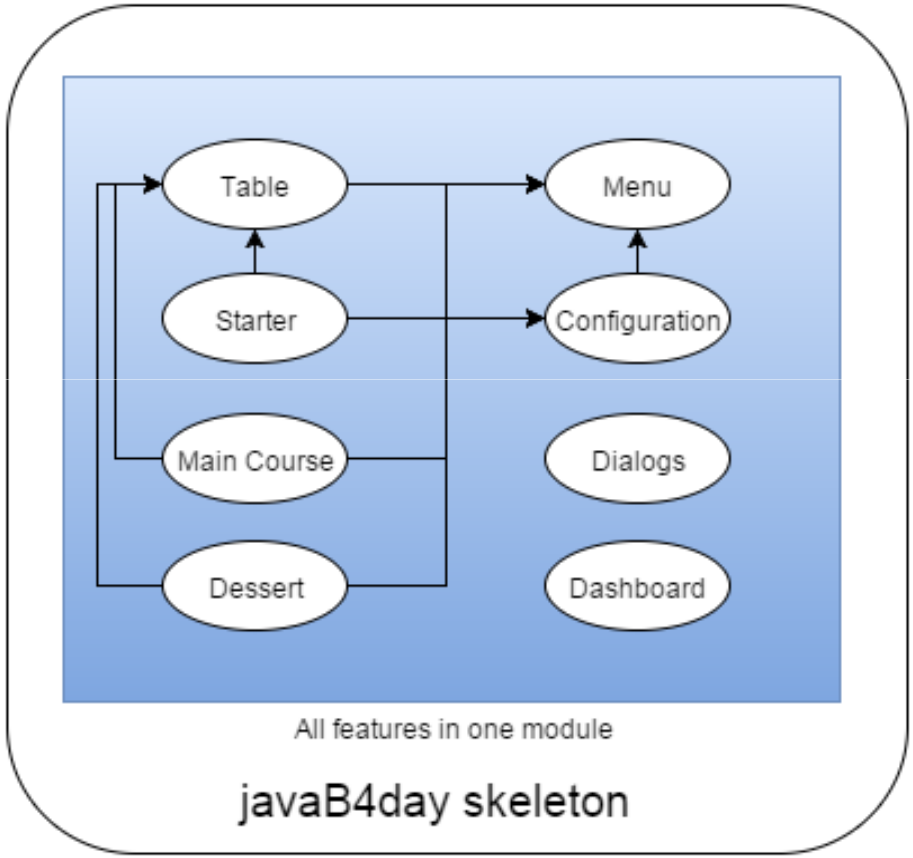


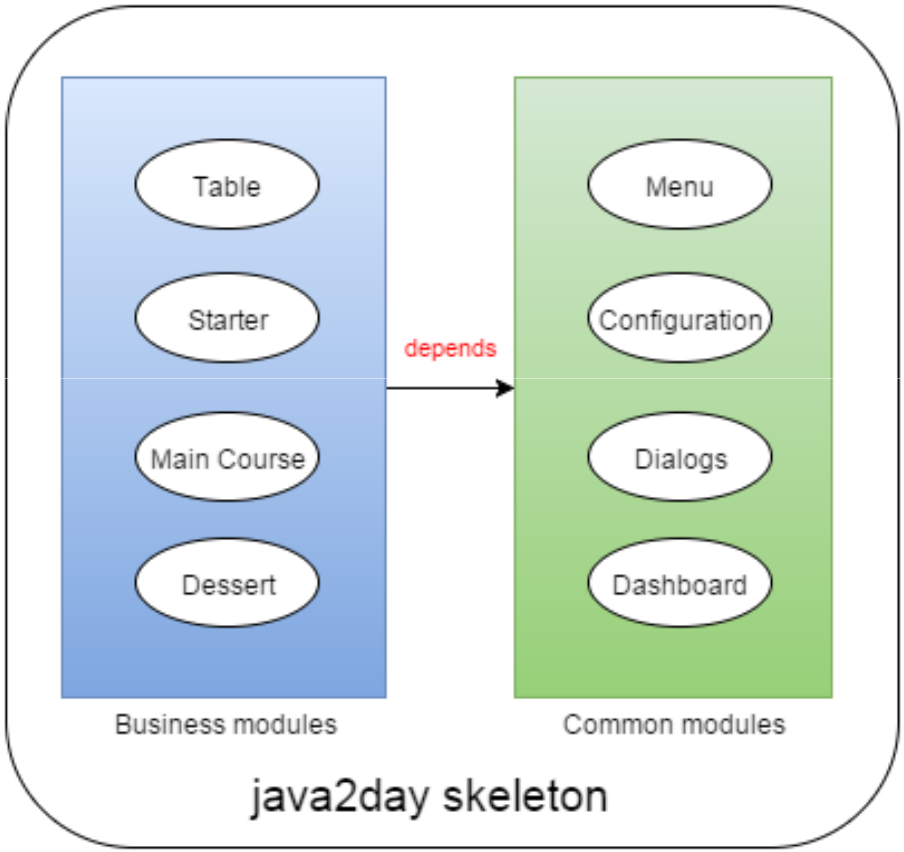
Modularization

Benefits of using modules

- Distributed / independent development
- Code reusability
- Program readability / quality
  - make some common module bower component?
- Encapsulation (module is a function)
- Manageable tasks
  - Design, implement and test

Actual vs. expected design





How to implement modularization

Think s lot!

2 approaches when you need to

- Starting from scratch
  - Think about every feature is a module to be reused in any kind of app
- Existing project
  - Total reorganization of code
  - REFACTOR
  - OR delete all code and start from scratch ☺

How to do it?

endava

Headline

Define meaning of module

- business vs. common module

Work in parallel between the teams:

- grouping / refactoring business code
  - (top down approach)
- grouping refactoring common code
  - (bottom up)
- While other team is making all compatible with build scripts, module distribution, CI

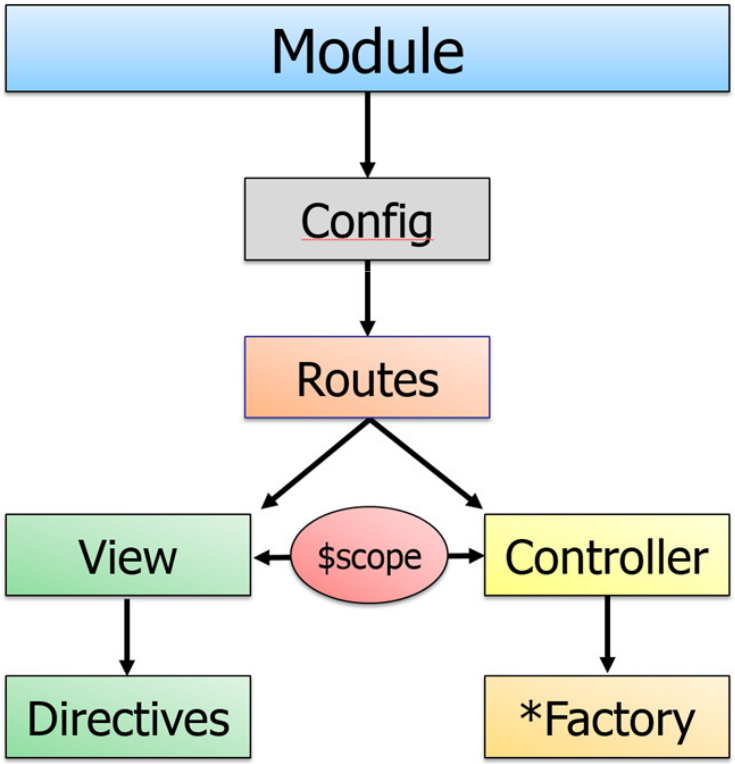
Business Modules

endava

Business Modules

- Structure
- Module definition: config / routes / dependencies
- Controllers: services, model
- View: html / directives
  - More directives, less views
- Other resources: css, images,
- Examples:
  - Starter meal
  - Main dish
  - Dessert

The Big Picture



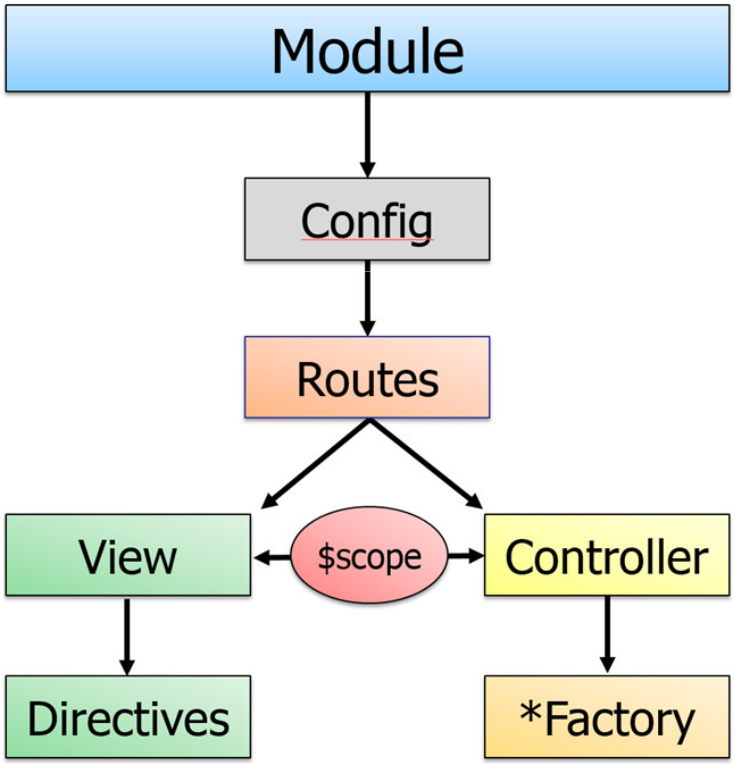
Common Modules

endava

Common Modules

- Same structure as business modules
- Mainly services and providers
- Html, directives for common UI components
- View: html / directives
  - More directives, less views
- Other resources: css, images,
- Examples:
  - Starter meal
  - Main dish
  - Dessert

The Big Picture



DEMO

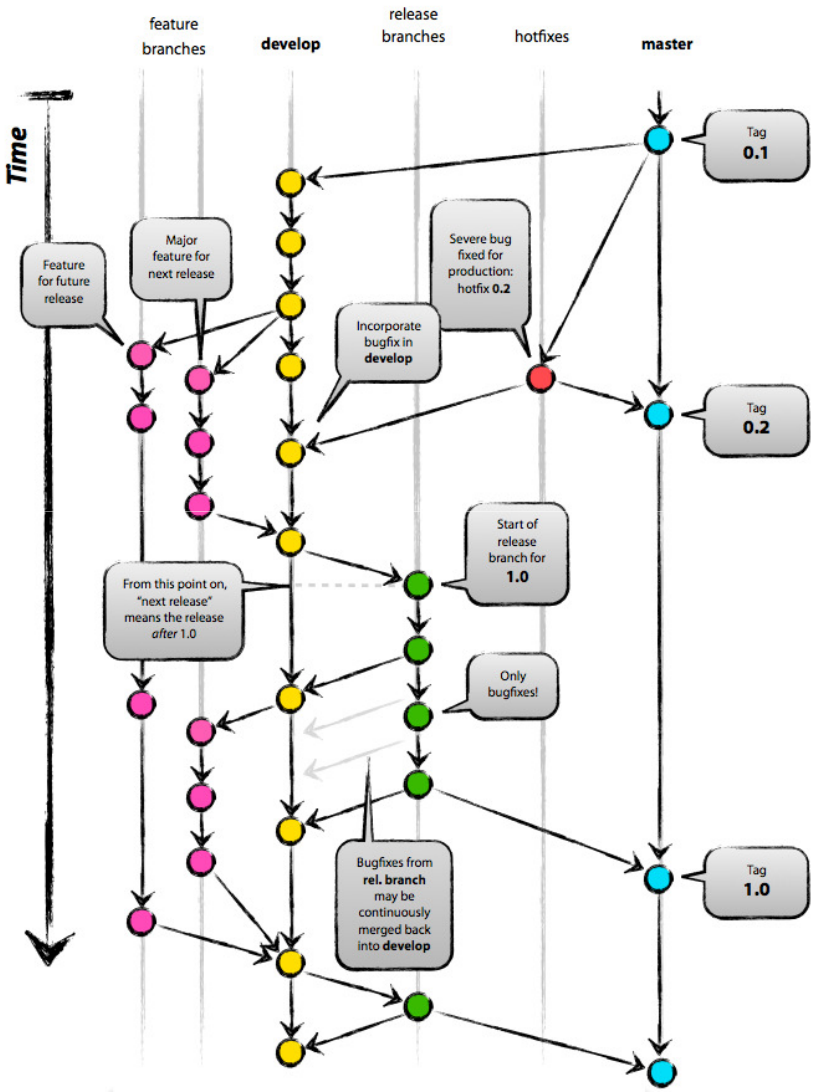
Going even further

Not only separate folders, but separate repos/versions

- bower
- Node modules
- Git submodules

Maven like modules/artifacts

Each module has its own definition



New version and the power of modularization

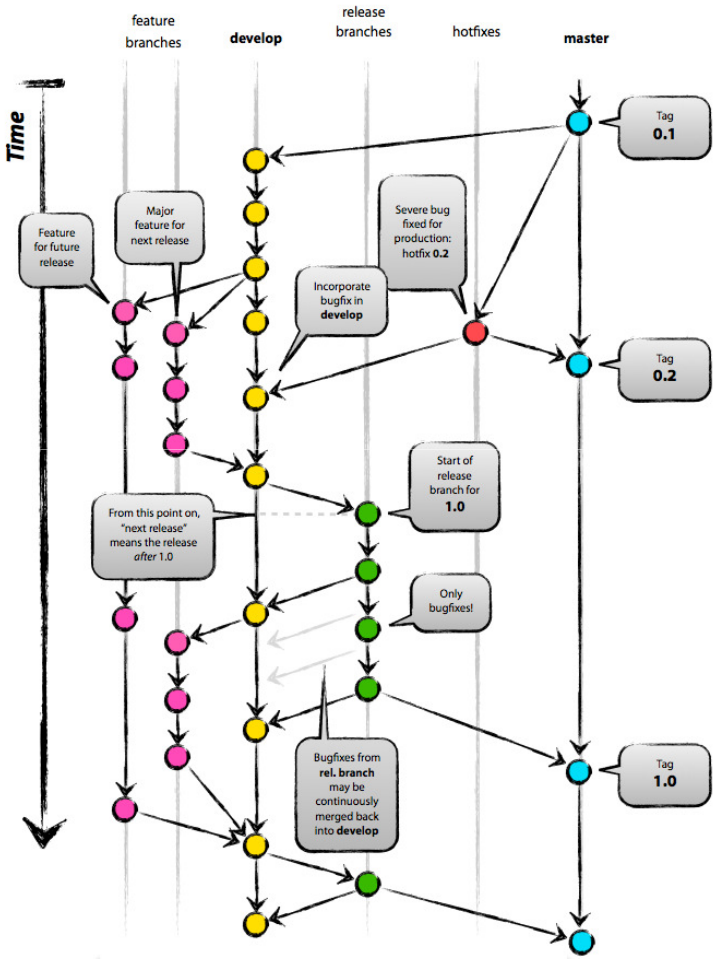
New Version

Java2day 1.0

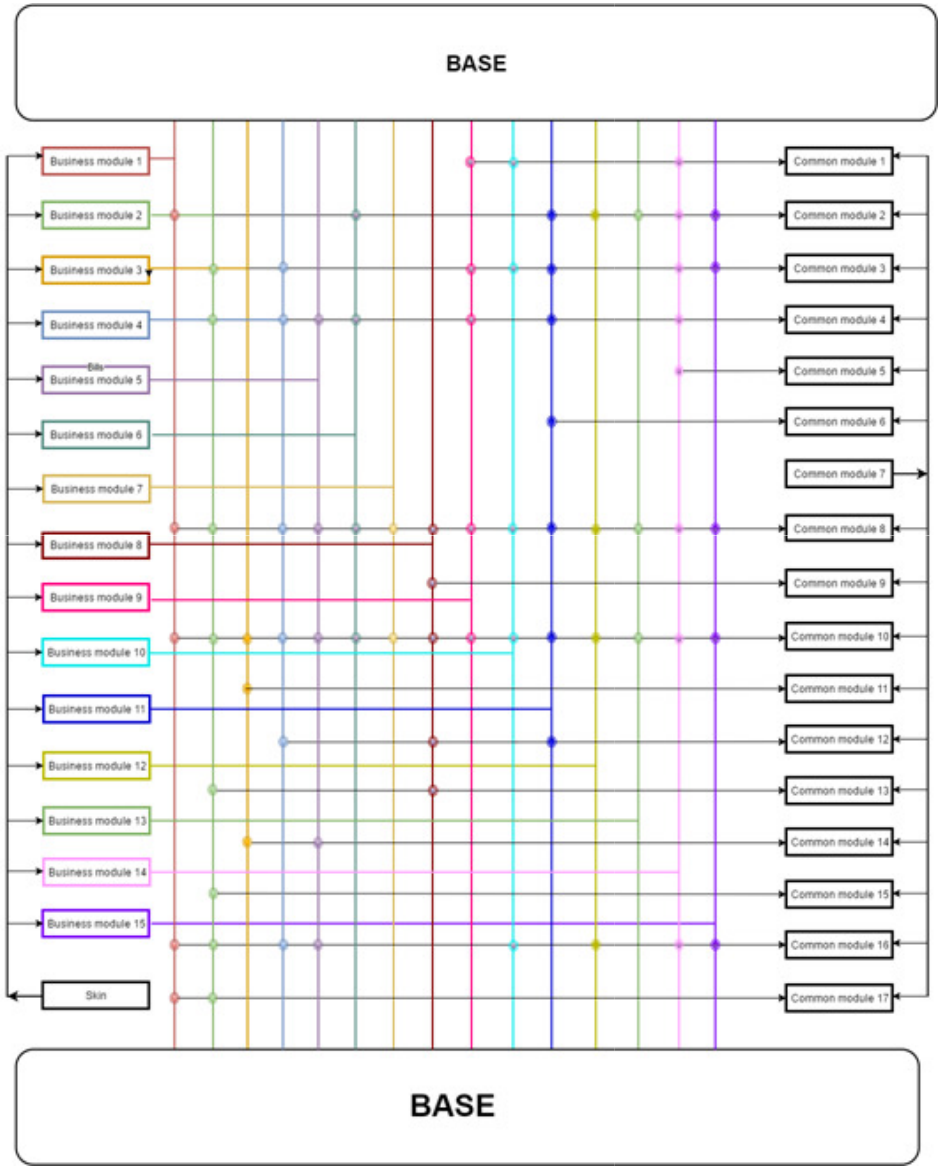
- Starter 1.3
- Main dish 1.1
- Dessert 1.5

Java2mmorow 1.0

- Main dish 1.4
- Dessert



How some app would look like



New approaches

ES 6

- A lot of cool features
  - Out of the box modules
  - Generators, arrow, object literals, string interpolation etc.
  - But do we really need Classes? Why it is bringing the OO design in already perfect functional language?

React JS

- Currently most trending and promising way of building applications that perfectly matches ES 6
- Angular 2 is basically “stealing” the good practices from React

Tools that we must use regardless framework that we use

endava

- Build tools
    - Grunt
    - Gulp
    - Webpack
    - Brunch and the list goes to infinity
    - Regardless which one is better we **must** use one!

- Do we really use the potential of our IDE?
    - Running projects
    - Debbuging?
    - Easy refactoring

How you will benefit from the modularized app

endava

Similar different client requirements

Backbase wants different apps and extensions

- Easy extension to current apps
- Different style
- Similar but still different set of features
  - Add new feature
  - Change old ones
- Make all of the components reusable

Modularization summary

Pros

- Increased speed of development
- Testable code, less regression bugs
- Faster, more secure application (sealed package)
- Improved the process of new app creation
- Ready for Continuous Integration

Cons

- Maintenance of module repositories is more time consuming
  - New repo, merge, versioning, etc.
- Needs a lot of attention when creating a business module
  - make it independent from other business modules

Obstacles

endava

Things that you should consider always

Code everywhere

- it’s hard to gain control with a monolithic design

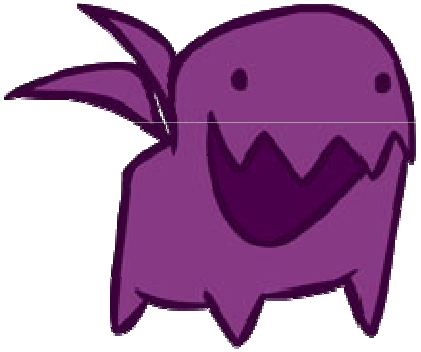
Plug-ability of modules

- make them aware of each other using module manager / registration

Generic common features

- menu, i18n bundles, configuration, caching, etc.

Questions?



Goran Kopevski

Senior Developer

goran.kopevski@endava.com

+389 70 949 363

en_gkopevski