



1. Aim

A case study into the benefits of configuration driven programming in web development.

Objectives

- To gain a systematic understanding of configuration driven programming and its effectiveness for an application.
- Critically evaluate the benefits of configuration driven programming by performing a case study of its implementation in an Angular application.
- Analyse the process and procedures created to determine the effectiveness of the development process.

2. Introduction

- CDD is the use of configuration files to influence the functionality of software.
- Everyone has used them in some sort of capacity, think configuring settings in phone applications you use
- In this case it's a faster development method that will be implemented into a development team in BT.
- They will only need to write a configuration file and 1 method in order to create order forms on their website.

3. Research Methodology

- Case study with results deduced from survey responses to gauge the project's success.
- Cross-section approach used. Interviews were conducted after developers had practical experience with the project. Compared with survey answers.

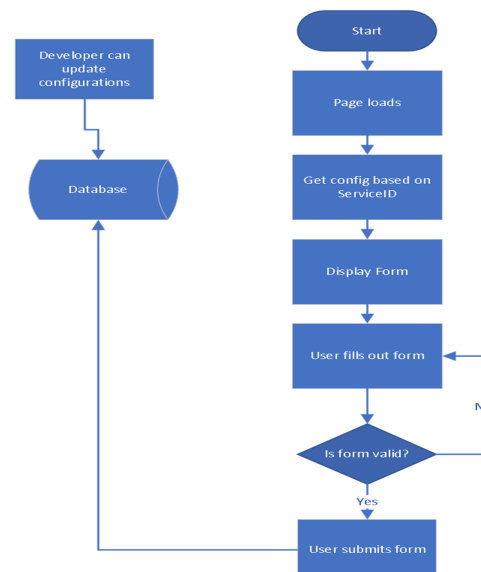
Development Methodology

- "Agile" methodology
- Iterative development process

4. Literature Review

- Configuration files used at run time to prevent application recompiling [1].
- Configuration files should be single-purpose, simple as possible and have a good naming convention that is self-descriptive [3].
- Configuration files improve scalability and provide a standard for writing order forms.
- Code to handle the configuration files is abstract to cope with varying configurations [4] [5].
- Code is more maintainable as there is less code. Common abstracted functionality that handles all configurations.
- JSON format used for configuration files for its good interoperability with Angular [2] [6].
- High initial cost to implement [6].
- Iterative development is recommended. Start with simple configurations and build it up [6].

5. Program Design

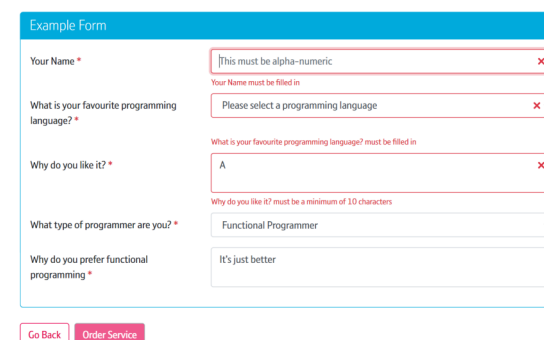


- Above is a high-level flow chart of the program.
- First iteration was to make a simple version of this.
- Each subsequent iteration built it into the final application.

6. Implementation

```
private static configuration: any = {}
yourName: {
  type: 'text',
  value: '',
  label: 'Your Name',
  placeholder: 'This must be alpha-numeric',
  rules: {
    required: true,
    minlength: 1,
  },
},
favouriteProgrammingLanguage: {
  type: 'select',
  label: 'What is your favourite programming language?',
  value: 'Please select a programming language',
  placeholder: 'Please select a programming language',
  options: [
    { value: 'A', label: 'A' },
  ],
  rules: {
    required: true,
  },
},
description: {
  type: 'textarea',
  value: '',
  label: 'Why do you like it?',
  placeholder: 'This can contain any character and line breaks',
  rules: {
    required: true,
    minlength: 10,
    maxlength: 4000,
  },
},
type: {
  type: 'select',
  label: 'What type of programmer are you?',
  value: 'Please select type',
  placeholder: 'Please select the type',
  options: [
    { value: 'Functional Programmer', label: 'Functional Programmer' },
  ],
  rules: {
    required: true,
  },
},
oopDescription: {
  type: 'textarea',
  value: '',
  label: 'Why do you prefer functional programming',
  placeholder: 'This can contain any character and line breaks',
  rules: {
    dependencies: [
      { dependentOn: 'type', dependentValue: 'Functional Programmer' },
    ],
  },
},
}
```

- This is the configuration file that generates the forms.
- It defines everything the form needs, like the type of form field, sets placeholders, gives values to dropdowns and validates the form with the set rules.



Example Form

Your Name * [This must be alpha-numeric]

What is your favourite programming language? * [Please select a programming language]

Why do you like it? * [A]

What type of programmer are you? * [Functional Programmer]

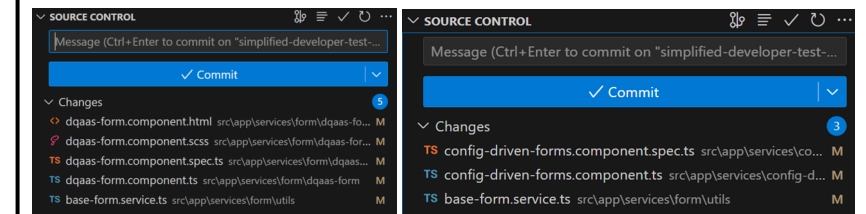
Why do you prefer functional programming? * [It's just better]

Go Back Order Service

- Configuration file looped over to display form.
- Placeholders describe the form fields for the users.
- Errors appear when form fields are invalid.

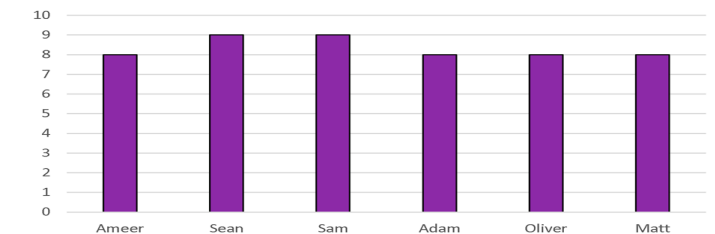
7. Results and Conclusions

- Was the project successful in speeding up Data Services frontend development time?



- The above images show the code to review before CDD and after its implementation. It has been reduced.

Rating of satisfaction with the projects implementation.



- The Data Services team at BT was happy with the overall idea and implementation of CDD.
- Overall responses to the survey and interviews were positive and in favour of using CDD with some critiques along the way.
- The team believes it would speed up their frontend development time.

8. Further Work

- To look into testing strategies for the configuration files.
- Configuration Management Tools to manage discrepancies between environments.

References

- J. G.-N. a. A. N. E. Alba, "On the Configuration of Optimization Algorithms by Using XML Files," 2003.
- G. L. a. D. Lemire, "Parsing Gigabytes of JSON per Second," *The VLDB Journal*, 13 08 2019.
- M. F. a. D. S. Tushar Sharma, "Does Your Configuration Code Smell?," 2016.
- A. D. e. al., "Extensible component-based architecture for FLASH, a massively parallel, multiphysics simulation code," *Parallel computing*, 2009.
- G. D. Bergland, A Guided Tour of Program Design Methodologies, 10 ed., vol. 14, 1981.
- B. N. a. m. DeMore, "Using XML configuration-driven development to create a customizable ground data system," 2009.