

アプリケーションプログラミングにおける
構造体の初期化に関する論考
～理論と実践～

市川恭佑

`papers@mail.ebi-yade.com`

目次

第 1 章	序論	1
1.1	研究背景	1
1.2	本論文の構成	2
第 2 章	基礎知識	3
第 3 章	提案手法	4
第 4 章	関連研究	5
第 5 章	実験	6
第 6 章	評価と考察	7
第 7 章	おわりに	8
参考文献		10

図目次

表目次

第 1 章

序論

1.1 研究背景

近年、Go や Rust などの強力な型システムを持つ言語の登場や、VSCode や IDEA などのコード解析機能が充実した IDE、そして AWS や GCP に代表されるパブリッククラウドの台頭により、アプリケーションプログラミングを取り巻く環境は大きく変化した。以前は、Web アプリケーションのみならず、コマンドラインツールなどを含む多様なアプリケーションの開発において、Ruby on Rails などの包括的なフレームワークに則ったプログラミングが一般的だった。これに対し、言語と IDE の発達はプログラマの認知負荷を効果的に減らした。つまり、一人ひとりが書き下し、把握できるコードの範囲は大きく広がった。また、パブリッククラウドは従量課金制を採用し、コンピューティングリソースを必要に応じて即座に用意できる機会を提供した。これにより、アプリケーションの実行環境は、垂直よりも水平にスケールすることが好まれる傾向が強まった。

上記の変化により、従来の“アプリケーションプログラマが把握するコード量を減らし、垂直なスケーリングを前提とした”フレームワークは徐々にシェアを減らしている。アプリケーションプログラマは、支配的なフレームワークの束縛から解放され、より自由度の高い設計でコードを書けるようになった。

しかし、このことは新たな課題を生み出した。それは、個々のアプリケーションを作るプログラマ自身が、ほとんど全ての期間における処理の流れとデータの扱いに責任を持つ必要に迫られたことである（以前、これはフレームワーク側の責務であった）。ゆえに、異なるアプリケーションのうち、似た振る舞いをする部分のデータの扱い方が、大きく異なることも少なくない。もちろん、一部は自由度の高さの代償として捉えることもできるが、すべてではない。記述的特徴に差異があったとしても、共通の骨法に沿ってコードが

設計されていることは、多角的な利益になる。作者以外がコードを読む際の認知負荷を減らし、また作者自身にとっても記述時の認知負荷を減らすだけでなく、意図しない挙動を防ぐ効果が期待できる。

1.2 本論文の構成

本論文の構成を以下に示す。

第2章では、本論文の基礎知識を述べる。

第3章では、本論文の提案手法を述べる。

第4章では、関連研究を紹介する。

第5章では、実験方法、実験結果を述べる。

第6章では、実験結果に対する評価と考察を述べる。

第7章では、本論文のまとめを述べる。

第 2 章

基礎知識

第 3 章

提案手法

第 4 章

関連研究

第 5 章

実験

第 6 章

評価と考察

第7章

おわりに

謝辭

thanks!

参考文献