MDC для реактивного приложения, или О скрещивании ужа с ежом

Лев Безбородов

Газпромбанк

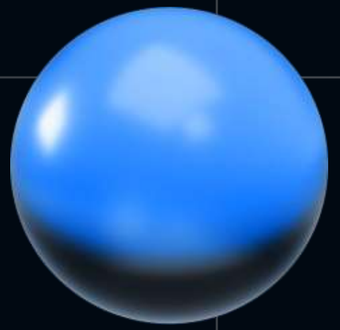
lev.bezborodov@phystech.edu

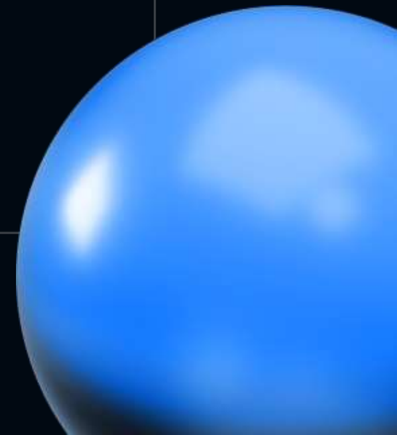
JPoint

ГАЗПРОМБАНК

Содержание доклада:

1. Введение в проблематику.
2. Поиск решения.
3. Структура итогового решения.
4. Защита служебных полей.
5. Нестроковые метаданные.
6. Итоги.



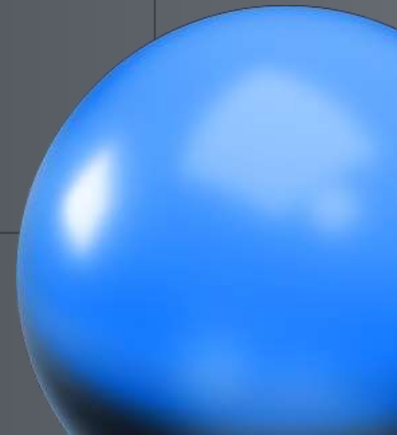


1. Введение в проблематику.

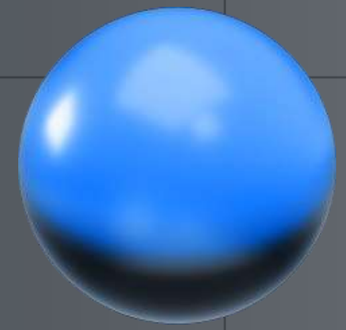
1.1. Два слова о реактивной парадигме.

1.2. Типовое использование классического MDC.

1.3. Неприменимость классического MDC в реактивном приложении.



2. Поиск решения.

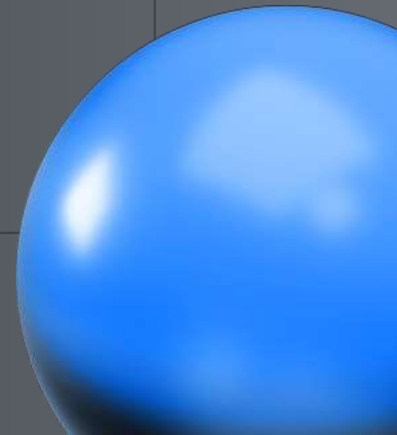


2.1. Официальное решение от команды Project Reactor.

2.2. Типовой сценарий, для которого официальное решение не подходит.

2.3. Официальное решение как исходная точка поиска решения.

2.4. Использование хуков реактора.



3. Структура итогового решения.

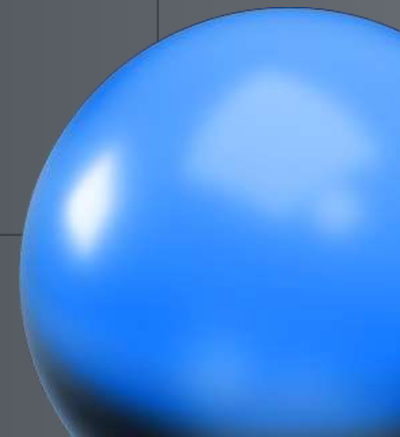
3.1. Таблица метаданных — в куче, идентификатор — в контексте.

3.2. Инициализация и утилизация таблицы метаданных.

3.3. Вычисление таблицы метаданных только при логировании.

3.4. Вспомогательный класс для работы с MDC-контекстом.

3.5. Замечание по поводу Mono.fromRunnable(...) и Mono.fromCallable(...).

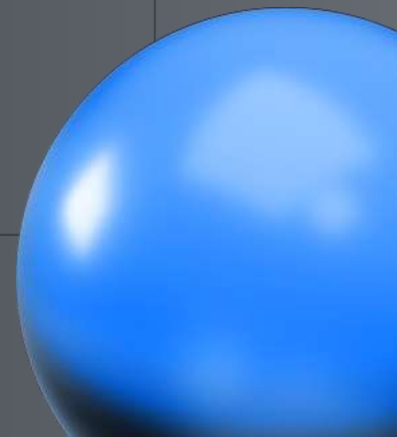


4. Защита служебных полей.

4.1. Возможность случайной перезаписи служебных полей.

4.2. Семантически служебные поля.

4.3. Безопасная работа со служебными полями.



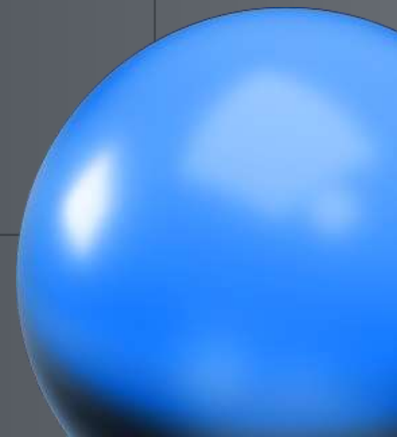
5. Нестроковые метаданные.

5.1. Ограничение на тип значений в классическом MDC.

5.2. Когда строкового типа данных недостаточно.

5.3. Костыльное решение проблемы с типом данных
и почему это плохо.

5.4. Целевое решение проблемы с нестроковым типом данных.



6. Итоги

- ✓ Типовой сценарий использования MDC работает в реактивном приложении «из коробки»;
- ✓ При этом механизм динамического заполнения метаданных безопасный;
- ✓ Имеется возможность при необходимости задать произвольный тип любого поля лог-объекта.