

Moshi Code Gen

Brian Plummer - Nike

Moshi Code Gen

A Love Story

Moshi Code Gen

=

Reflection Free Parsing

&

Immutable Data Model

ELI5 Definitions

Immutable

Not Mutable

Hard to change

Reflection Free

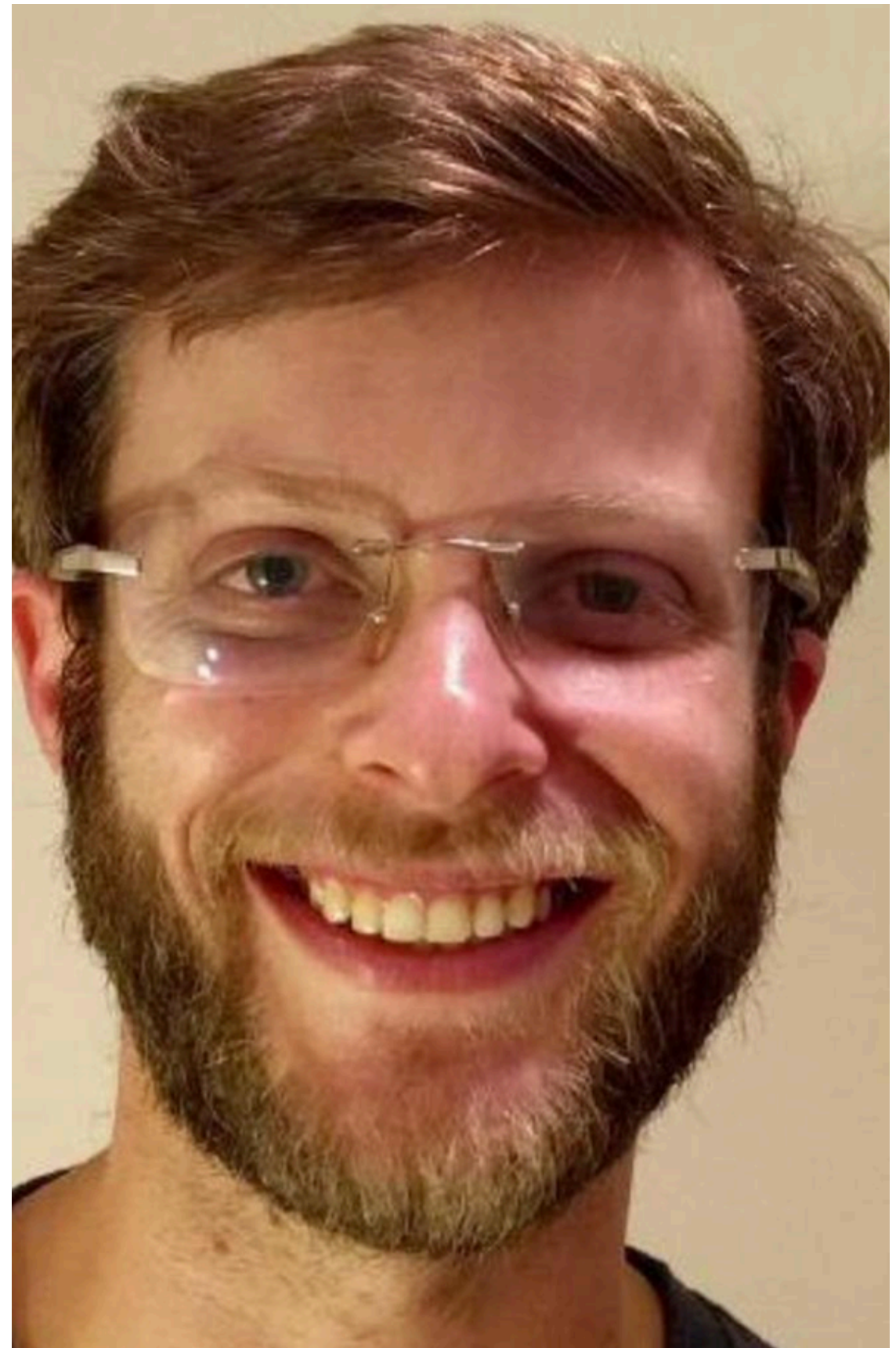
Not Dynamic

Compiled

**Why does
this matter?**

Mike's first month at the New York Times

Application startup time
was horrible. He started
digging and found some
issues.



**One big culprit was
parsing a configuration
file that we needed for
Application start up**

**One big culprit was
parsing a configuration
file that we needed for
Application start up**

Roughly 700ms wasted!

**What was the
issue? Why did it
take so long?**

What was the issue?

**Why did it take so
long?**

Reflection!

How shall we fix this?

How shall we fix this?

**We could write our own
type adapters**



```
@Override
    public Book read(final JsonReader in) throws IOException {

        final Book book = new Book();

        in.beginObject();
        while (in.hasNext()) {

            switch (in洗洗洗Name()) {
                case "isbn":
                    book.setIsbn(in洗洗洗String());
                    break;
                case "title":
                    book.setTitle(in洗洗洗String());
                    break;
                case "authors":
                    book.setAuthors(in洗洗洗String().split(";"));
                    break;
            }
        }
        in.endObject();
        return book;
    }
```

BACK IN MY DAY

**WE CODED UPHILL BOTH WAYS TO
WORK**

memegenerator.net

Annotation Processing

- AutoValue
- Immutables.org

Unfun stuff taken care of for us

- `toString()`
- `equals()`
- `hashCode()`

Copy methods for mutate

Immutable Data Model

- **Thread safety**
- **Reduced Complexity**
- **No more bugs caused by mutation**

Reflection free type adapters

All was well!

**Kotlin came and we
have data classes**



```
data class Book (  
    val title: String?,  
    val authors: List<Author>,  
    val isbn: String  
)
```


Data classes give us most of what we needed

- Unfun methods
- Copy methods
- Immutability
- ~~Reflection free parsing~~

Moshi to the rescue!

```
data class Book (  
    val title: String?,  
    val authors: List<Author>,  
    val isbn: String  
)
```

```
@JsonClass(generateAdapter = true)

data class Book (

    val title: String?,

    val authors: List<Author>,

    val isbn: String

)
```

That's it!

That's it!

**Really really it..besides
adding it as a dependency
in gradle of course!**

**Really really it..besides
adding it as a dependency
in gradle of course!**

```
implementation "com.squareup.moshi:moshi:$moshi_version"
```

```
kapt("com.squareup.moshi:moshi-kotlin-codegen:$moshi_version")
```

**That sounds too good to
be true.**

**That sounds too good to
be true.**

How does it work?

How does it work?

Annotation processing

**Type adapter resolves at
runtime so you don't
have to do anything else**

**Uses reflection once to
lookup, caches it for use.
Good across modules!**

Dig deeper

gson? It's time

**Moshi is the way
forward**

How Slow is Reflection in Android?

By Anton Krasov • Feb 23, 2016

[View site information](#)

**[http://blog.nimbleandroid.com/
2016/02/23/slow-Android-
reflection.html](http://blog.nimbleandroid.com/2016/02/23/slow-Android-reflection.html)**

Exploring Moshi's Kotlin Code Gen



Zac Sweers

Follow


May 16, 2018 · 9 min read

**[https://medium.com/
@ZacSweers/exploring-moshis-
kotlin-code-gen-dec09d72de5e](https://medium.com/@ZacSweers/exploring-moshis-kotlin-code-gen-dec09d72de5e)**

Improving Startup Time in the NYTimes Android App

BY MIKE NAKHIMOVICH FEBRUARY 11, 2016 1:56 PM 2

 Email

 Share

 Tweet

 Save

 More

Improving application startup and load time has been a priority for The New York Times Android development team, and we're not alone. As device manufacturers continue to offer faster and more fluid experiences, users expect their native apps to be faster still.

Our team recently rewrote our news app to take advantage of modern patterns such as dependency injection and reactive programming. The rewrite offered improvements in maintenance, code modernization and modularization benefits, but required some adjustments to optimize.

**[https://
open.blogs.nytimes.com/
author/mike-nakhimovich/](https://open.blogs.nytimes.com/author/mike-nakhimovich/)**