Moshi's Kotlin CodeGen

brian plummer

Or how we should all have
an Immutable Data Model
with Reflection Free Parsing

- Immutable -

- Immutable - It’s hard to mutate

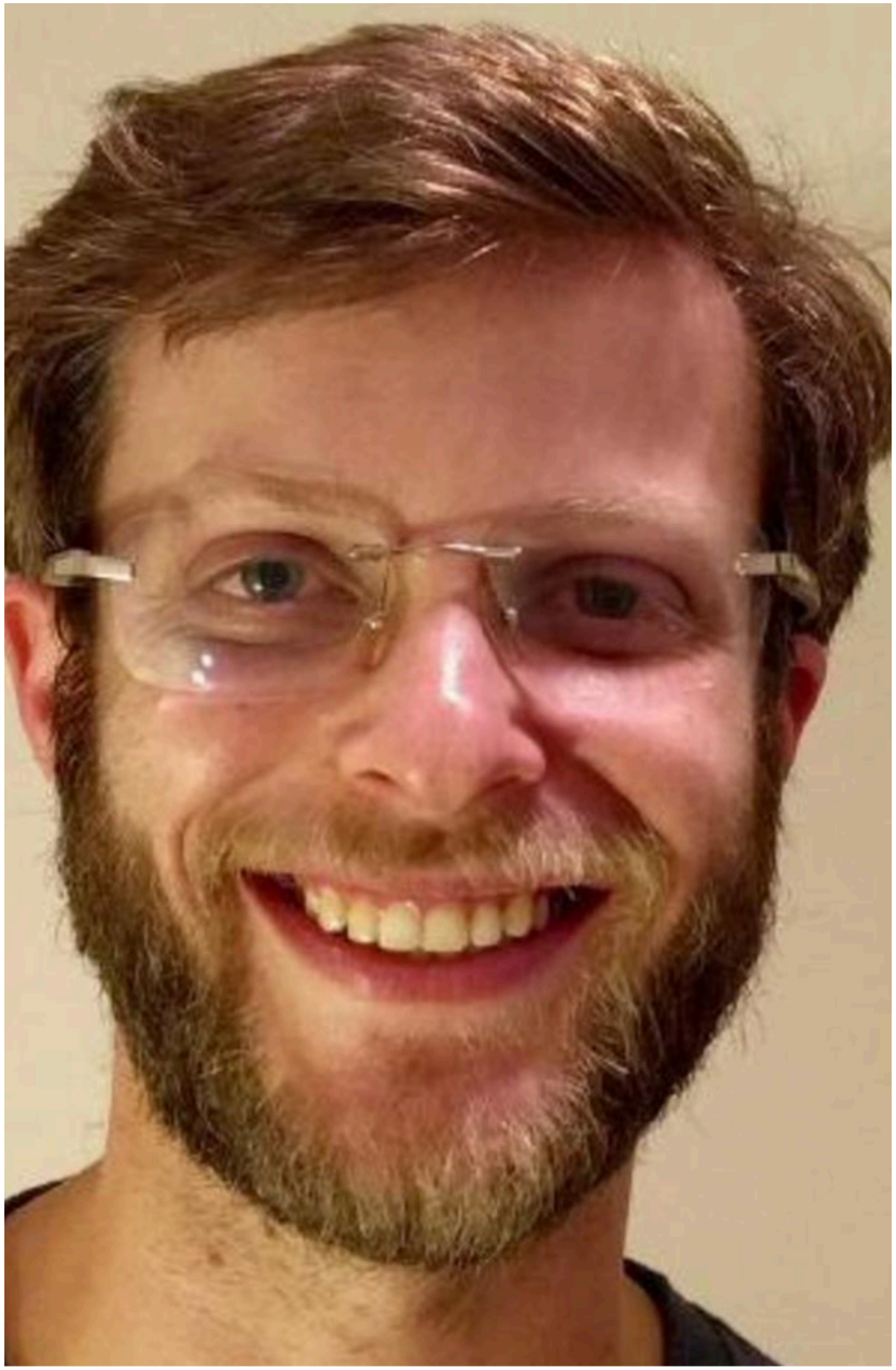
- Immutable - It’s hard to mutate
- Reflection -

- Immutable - It’s hard to mutate
- Reflection - It’s costly on Android

Why does this even 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

~700ms

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

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

Reflection!

How would we fix this?

We could write our own type adapters

```java
@Override

  public Book read(final JsonReader in) throws IOException {

    final Book book = new Book();

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

      switch (in.nextName()) {
      case "isbn":
        book.setIsbn(in.nextString());
        break;
      case "title":
        book.setTitle(in.nextString());
        break;
      case "authors":
        book.setAuthors(in.nextString().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()

Unfun stuff taken care of for us

- toString()
- equals()

Unfun stuff taken care of for us

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

Copy methods for mutate

Immutable Data Model

Immutable Data Model

- Thread safety

Immutable Data Model

- Thread safety
- Reduced Complexity

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!

```kotlin
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!

That’s it! 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.

How does it work?

Annotation processing

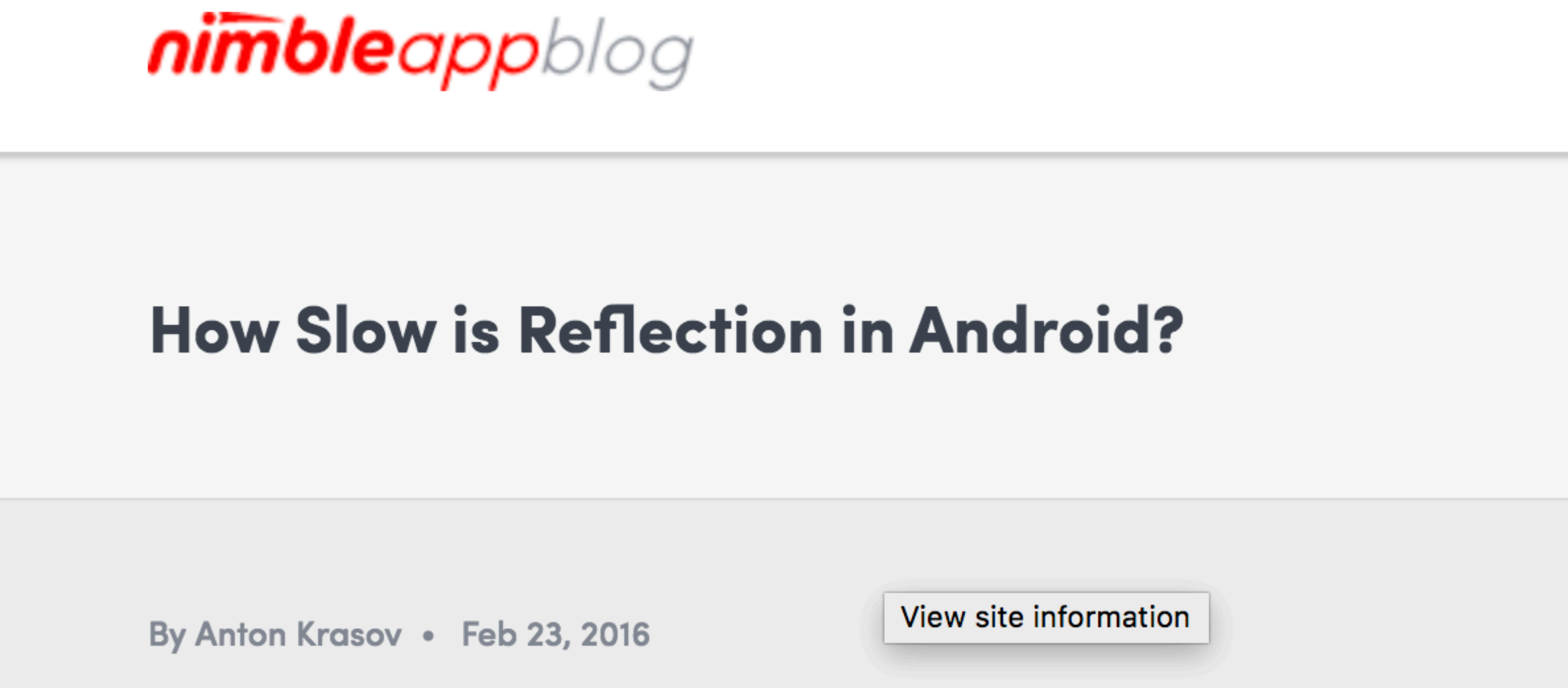
Type adapter resolves at runtime so you don’t have to do anything else

Dig deeper

Dig deeper

gson? It’s time

Moshi is the way forward

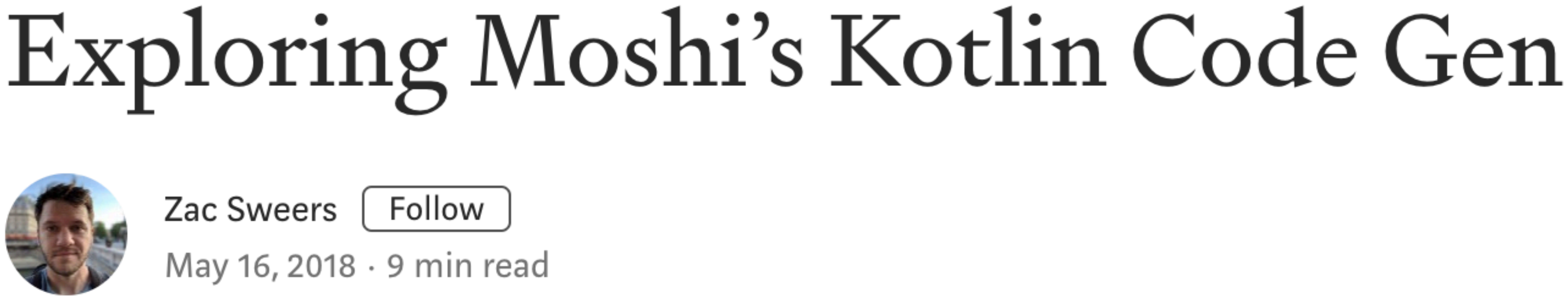


nimbleappblog

How Slow is Reflection in Android?

By Anton Krasov • Feb 23, 2016

http://blog.nimbledroid.com/2016/02/23/slow-Android-reflection.html

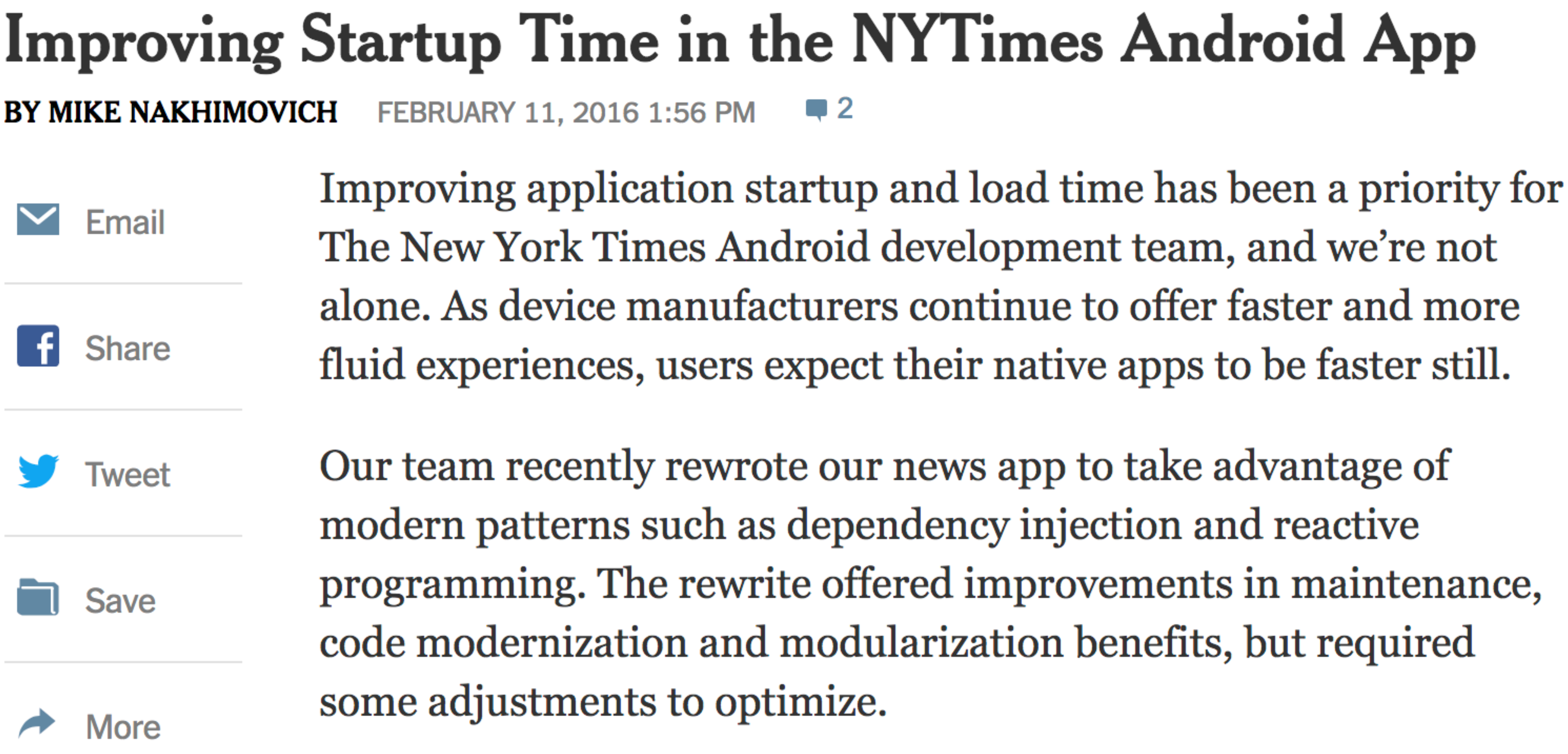


Exploring Moshi's Kotlin Code Gen

Zac Sweers

May 16, 2018 · 9 min read

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

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/