



AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

ProtoDoc - odpowiednik JavaDoc dla Google Protocol Buffers

Konrad Malawski

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki
Katedra Automatyki

6-12-2011

✚ Wprowadzenie

- ▶ Przedstawienie problemu dokumentacji Protocol Buffers
- ▶ Cel projektu

✦ Wprowadzenie

- ▶ Przedstawienie problemu dokumentacji Protocol Buffers
- ▶ Cel projektu

✦ Opis implementacji

- ▶ Architektura aplikacji
- ▶ Porównanie narzędzi i uzasadnienie dokonanych wyborów
- ▶ Scala Parser Combinators

✦ Wprowadzenie

- ▶ Przedstawienie problemu dokumentacji Protocol Buffers
- ▶ Cel projektu

✦ Opis implementacji

- ▶ Architektura aplikacji
- ▶ Porównanie narzędzi i uzasadnienie dokonanych wyborów
- ▶ Scala Parser Combinators

✦ Wyniki pracy

- ▶ Integracja z Apache Maven
- ▶ Przykładowy efekt działania aplikacji

- ✚ Wprowadzenie
 - ▶ Przedstawienie problemu dokumentacji Protocol Buffers
 - ▶ Cel projektu
- ✚ Opis implementacji
 - ▶ Architektura aplikacji
 - ▶ Porównanie narzędzi i uzasadnienie dokonanych wyborów
 - ▶ Scala Parser Combinators
- ✚ Wyniki pracy
 - ▶ Integracja z Apache Maven
 - ▶ Przykładowy efekt działania aplikacji
- ✚ Podsumowanie i przyszłość projektu

Nie istnieje zunifikowany sposób dokumenowania oraz publikowania wiadomości Protocol Buffers.

-

Typowymi problemami są:

Nie istnieje zunifikowany sposób dokumenowania oraz publikowania wiadomości Protocol Buffers.

-
Typowymi problemami są:

- ✚ Praca z wygenerowanymi z ProtoBuf klasami, nie zawierającymi dokumentacji.

Nie istnieje zunifikowany sposób dokumenowania oraz publikowania wiadomości Protocol Buffers.

-

Typowymi problemami są:

- ✦ Praca z wygenerowanymi z ProtoBuf klasami, nie zawierającymi dokumentacji.
- ✦ Trudności w rozpoznawaniu pól „legacy”, których nie powinno się już używać.
(Raz dodane do wiadomości pole, zostaje w niej „na zawsze”).

Nie istnieje zunifikowany sposób dokumenowania oraz publikowania wiadomości Protocol Buffers.

-

Typowymi problemami są:

- ✦ Praca z wygenerowanymi z ProtoBuf klasami, nie zawierającymi dokumentacji.
- ✦ Trudności w rozpoznawaniu pól „legacy”, których nie powinno się już używać.
(Raz dodane do wiadomości pole, zostaje w niej „na zawsze”).
- ✦ Brak możliwości przeglądnięcia dokumentacji / listy wiadomości online (przed pobraniem ich).

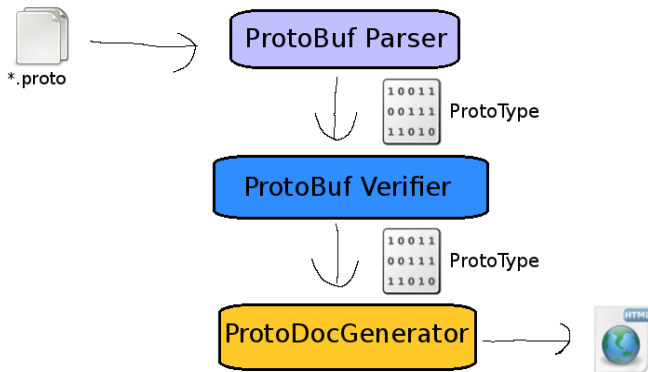
ProtoDoc stawia sobie za cel rozwiązanie problemów dokumentowania wiadomości Protocol Buffers poprzez:

ProtoDoc stawia sobie za cel rozwiązanie problemów dokumentowania wiadomości Protocol Buffers poprzez:

- ✚ Parsera języka definicji interfejsów Protocol Buffers (pliki *.proto)
 - ▶ Nie ignorując komentarzy.

ProtoDoc stawia sobie za cel rozwiązanie problemów dokumentowania wiadomości Protocol Buffers poprzez:

- ✚ Parsera języka definicji interfejsów Protocol Buffers (pliki *.proto)
 - ▶ Nie ignorując komentarzy.
- ✚ Stworzenie narzędzia generującego z uzyskanych informacji stronę www (ala JavaDoc), zawierającą wszystkie wiadomości jak i ich komentarze, pola oraz definiowane typy.





AGH

Architektura aplikacji

✦ ProtoBufParser

- ▶ Parsuje po jednym pliku (który może zawierać wiele wiadomości)
- ▶ Tworzy obiektu typu ProtoType, zawierające pełną informację o typie / wiadomości
- ▶ Przeprowadza analizę syntaktyczną



AGH

Architektura aplikacji

✦ ProtoBufParser

- ▶ Parsuje po jednym pliku (który może zawierać wiele wiadomości)
- ▶ Tworzy obiektu typu ProtoType, zawierające pełną informację o typie / wiadomości
- ▶ Przeprowadza analizę syntaktyczną

✦ ProtoBufVerifier

- ▶ Uruchamiany jest po zakończeniu parsingu wszystkich plików
- ▶ Przeprowadza analizę semantyczną uzyskanych obiektów ProtoType
- ▶ Między innymi, sprawdza widoczność oraz poprawność używanych typów



AGH

Architektura aplikacji

✦ ProtoBufParser

- ▶ Parsuje po jednym pliku (który może zawierać wiele wiadomości)
- ▶ Tworzy obiektu typu ProtoType, zawierające pełną informację o typie / wiadomości
- ▶ Przeprowadza analizę syntaktyczną

✦ ProtoBufVerifier

- ▶ Uruchamiany jest po zakończeniu parsingu wszystkich plików
- ▶ Przeprowadza analizę semantyczną uzyskanych obiektów ProtoType
- ▶ Między innymi, sprawdza widoczność oraz poprawność używanych typów

✦ ProtoDocGenerator

- ▶ Generuje strony www zawierające informacje o sparsowanych wiadomościach (ProtoType'ach)
- ▶ Wykorzystuje również informacje o potencjalnych błędach które mógł wykryć ProtoBufVerifier

Podczas wyboru technologii (generatora) parsera przeglądano popularne narzędzia takie jak JBison / JFlex, jednak ostateczny wybór padł na Scala Parser Combinators - z powodu elegancji tego rozwiązania.

Podczas wyboru technologii (generatora) parsera przeglądano popularne narzędzia takie jak JBison / JFlex, jednak ostateczny wybór padł na Scala Parser Combinators - z powodu elegancji tego rozwiązania.

Kombinator Parserów

Podejście to pozwala na czytelną i elegancką implementację Rekursywnie Zstępującego Parsera.

Kombinatory parserów (konkretna implementacja) są częścią języka Scala, zastosowanego do implementacji parsera w tym projekcie.

–
Zalety zastosowanego rozwiązania:

Kombinatory parserów (konkretna implementacja) są częścią języka Scala, zastosowanego do implementacji parsera w tym projekcie.

-

Zalety zastosowanego rozwiązania:

- ✚ Kod jest czytelny oraz łatwy w utrzymaniu (przypomina notację BNF)

Kombinatory parserów (konkretna implementacja) są częścią języka Scala, zastosowanego do implementacji parsera w tym projekcie.

-

Zalety zastosowanego rozwiązania:

- ✦ Kod jest czytelny oraz łatwy w utrzymaniu (przypomina notację BNF)
- ✦ Nie jest konieczne generowanie plików „implementacji” parsera, definicje które piszemy same z siebie są wykonywalne

Kombinatory parserów (konkretna implementacja) są częścią języka Scala, zastosowanego do implementacji parsera w tym projekcie.

-

Zalety zastosowanego rozwiązania:

- ✦ Kod jest czytelny oraz łatwy w utrzymaniu (przypomina notację BNF)
- ✦ Nie jest konieczne generowanie plików „implementacji” parsera, definicje które piszemy same z siebie są wykonywalne
- ✦ Umożliwia to podejście do projektu w sposób prowadzony testami (ponad 40!).

Analogicznie jak w przypadku znanego pluginu Javadoc:

```
mvn javadoc:javadoc
```

ProtoDoc:

```
mvn protodoc:protodoc
```



ProtoDoc

by Konrad Malawski konrad.malawski@poczta.pw.edu.pl


Table of Contents

Messages:

[Search...](#)

- [pl.project13.AmazingMessage](#)
- [pl.project13.AmazingMessage.EnumType](#)
- [pl.project13.AmazingMessage.InnerMessage](#)
- [pl.project13.AmazingMessage.SecondEnumType](#)
- [pl.project13.MessageWithInner.InnerMessage](#)
- [pl.project13.MessageWithInner.InnerMessage](#)
- [pl.project13.TopLevel](#)
- [pl.project13.TopLevel.MiddleLevel](#)
- [pl.project13.TopLevel.MiddleLevel.InnermostLevel](#)
- [pl.project13.WithEnum](#)
- [pl.project13.WithEnum.MessageType](#)

ProtoDoc (by Konrad Malawski) is Free Software.
Licensed under the Apache2 License. Visit the sources? [Visit the sources? \[protodoc.ghub\]\(https://github.com/kmalawski/protodoc\)](https://github.com/kmalawski/protodoc)



This message has an enum

defined in package: pl.project13

Fields:

message_medium

Modifier: optional
Tag: 3
Defined as: MessageType
Mapped to: MessageType

name

Default value: default
Modifier: required
Tag: 2
Defined as: string
Mapped to: java.lang.String

Inner Enums:

MessageType

This is a documented enumeration On protodoc saved my life

Defines values:

Value	TAG	Comment
EMAIL	1	Sent messages via email
SMS	2	Sent messages via sms
NORMAL	3	Sent messages via normal mail

ProtoDoc

by Konrad Malawski konradmalawski@gmail.pl

Table of Contents

Messages:

Search

- [pl.project13.AmazingMessage](#)
- [pl.project13.AmazingMessage.EnumType](#)
- [pl.project13.AmazingMessage.InnerMessage](#)
- [pl.project13.AmazingMessage.SecondEnumType](#)
- [pl.project13.MessageWithHint](#)
- [pl.project13.MessageWithHint.InnerMessage](#)
- [pl.project13.TopLevel](#)
- [pl.project13.TopLevel.MiddleLevel](#)
- [pl.project13.TopLevel.MiddleLevel.InnerLevel](#)
- [pl.project13.WithEnum](#)
- [pl.project13.WithEnum.MessageType](#)

ProtoDoc (by Konrad Malawski) is Free Software, licensed under the Apache2 License. Want the sources? [Get it on GitHub](#).

E MessageType

This is a documented enumeration. Oh protoDoc saved my life

defined in message: [pl.project13.WithEnum](#)

Values:

EMAIL
Send messages via email
Tag 1
SMS
Send messages via sms
Tag 2
MAIL
Send messages via normal mail
Tag 3

Spełniono założenia projektu.

Powstał:

- ✦ Parser podstawowej części Protocol Buffers Interface Description Language
- ✦ Generator dokumentacji
- ✦ ... jako element dodatkowy element - Weryfikator poprawności (widoczności typów) w sparsowanych wiadomościach

- ✦ Plugin Mavenowy zostanie opublikowany w Maven Central Umożliwiając dowolne wykorzystywanie go „out of the box” w rzeczywistych projektach.
- ✦ Plugin jak i parser zostaną umieszczone na ogólnodostępnej stronie www, pod licencją GPLv3, celem ułatwienia wprowadzania poprawek przez społeczność.

Dziękuję za uwagę.