package com.twitter.unified\_user\_actions.enricher.graphql

import com.twitter.dynmap.DynMap

import com.twitter.inject.Test

import com.twitter.util.Return

import com.twitter.util.Throw

import com.twitter.util.Try

import org.scalatest.matchers.should.Matchers

class GraphqlSpecs extends Test with Matchers {

trait Fixtures {

val sampleError = """

|{

| "errors": [

| {

| "message": "Some err msg!",

| "code": 366,

| "kind": "Validation",

| "name": "QueryViolationError",

| "source": "Client",

| "tracing": {

| "trace\_id": "1234567890"

| }

| }

| ]

|}""".stripMargin

val sampleValidRsp =

"""

|{

| "data": {

| "tweet\_result\_by\_rest\_id": {

| "result": {

| "core": {

| "user": {

| "legacy": {

| "id\_str": "12"

| }

| }

| }

| }

| }

| }

|}

|""".stripMargin

val sampleValidRspExpected = Return(

Set(("data.tweet\_result\_by\_rest\_id.result.core.user.legacy.id\_str", "12")))

val sampleErrorExpected = Throw(

GraphqlRspErrors(

DynMap.from(

"errors" -> List(

Map(

"message" -> "Some err msg!",

"code" -> 366,

"kind" -> "Validation",

"name" -> "QueryViolationError",

"source" -> "Client",

"tracing" -> Map("trace\_id" -> "1234567890")

)))))

def toFlattened(testStr: String): Try[Set[(String, Any)]] =

GraphqlRspParser.toDynMap(testStr).map { dm => dm.valuesFlattened.toSet }

}

test("Graphql Response Parser") {

new Fixtures {

toFlattened(sampleValidRsp) shouldBe sampleValidRspExpected

toFlattened(sampleError) shouldBe sampleErrorExpected

}

}

}