Table of Contents

Feature Recipes

The Feature classes are in the **geoscript.feature** package.

The major classes in this package include Field, Schema, and Feature.

A Field has a name and a type and describes a column of data.

A Schema is a collection of Fields together with a name. Schemas are used to create new Layers.

A Feature contains a geometry and a collection of attributes. A collection of Features is called a Layer.

Creating Fields

Create a Field with a name and a type

```
Field field = new Field("name", "String")
println field
```

```
name: String
```

Create a Geometry Field with a name and a geometry type and an optional projection

```
Field field = new Field("geom", "Point", "EPSG:4326")
println field
```

```
geom: Point(EPSG:4326)
```

Create a Field with a List of Strings (name, type, projection)

```
Field field = new Field(["geom", "Polygon", "EPSG:4326"])
println field
```

```
geom: Polygon(EPSG:4326)
```

Create a Field from a Map where keys are name, type, proj

```
Field field = new Field([
        "name": "geom",
        "type": "LineString",
        "proj": new Projection("EPSG:4326")
])
println field
```

```
geom: LineString(EPSG:4326)
```

Access a Field's properties

```
Field field = new Field("geom", "Point", "EPSG:4326")
println "Name = ${field.name}"
println "Type = ${field.typ}"
println "Projection = ${field.proj}"
println "Is Geometry = ${field.geometry}"
```

```
Name = geom
Type = Point
Projection = "EPSG:4326
Is Geometry = true
```

Creating Schemas

Create a Schema from a list of Fields

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
println schema
```

```
cities geom: Point(EPSG:4326), id: Integer, name: String
```

Create a Schema from a list of Lists

```
Schema schema = new Schema("cities", [
        ["geom", "Point", "EPSG:4326"],
        ["id", "Integer"],
        ["name", "String"]
])
println schema
```

```
cities geom: Point(EPSG:4326), id: Integer, name: String
```

Create a Schema from a list of Maps

```
Schema schema = new Schema("cities", [
        [name: "geom", type: "Point", proj: "EPSG:4326"],
        [name: "id",   type: "Integer"],
        [name: "name", type: "String"]
])
println schema
```

```
cities geom: Point(EPSG:4326), id: Integer, name: String
```

Create a Schema from a string

```
Schema schema = new Schema("cities", "geom:Point:srid=4326,id:Integer,name:String")
println schema
```

```
cities geom: Point(EPSG:4326), id: Integer, name: String
```

Getting Schema Properties

Get the Schema's name

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
], "https://github.com/jericks/geoscript-groovy-cookbook")
String name = schema.name
println name
```

```
cities
```

Get the Schema's geometry Field

```
Field geomField = schema.geom
println geomField
```

```
geom: Point(EPSG:4326)
```

Get the Schema's Projection

```
Projection proj = schema.proj
println proj
```

```
EPSG:4326
```

Get the Schema's URI

```
String uri = schema.uri
println uri
```

```
https://github.com/jericks/geoscript-groovy-cookbook
```

Get the Schema's specification string

```
String spec = schema.spec
println spec
```

```
geom:Point:srid=4326,id:Integer,name:String
```

Getting Schema Fields

Get the Schema's Fields

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
List<Field> fields = schema.fields
fields.each { Field field ->
    println field
}
```

```
geom: Point(EPSG:4326)
id: Integer
name: String
```

Get a Field

```
Field nameField = schema.field("name")
println nameField
```

```
name: String
```

Get a Field

```
Field idField = schema.get("id")
println idField
```

```
id: Integer
```

Check if a Schema has a Field

```
boolean hasArea = schema.has("area")
println "Has area Field? ${hasArea}"

boolean hasGeom = schema.has("geom")
println "Has geom Field? ${hasGeom}"
```

```
false
true
```

Modifying Schemas

Change the projection of a Schema

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Schema reprojectedSchema = schema.reproject("EPSG:2927", "cities_spws")
```

```
cities_spws geom: Point(EPSG:2927), id: Integer, name: String
```

Change the geometry type of a Schema

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Schema polyognSchema = schema.changeGeometryType("Polygon", "cities_buffer")
```

```
cities_buffer geom: Polygon(EPSG:4326), id: Integer, name: String
```

Change a Field definition of a Schema

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Schema guidSchema = schema.changeField(schema.field('id'), new Field('guid','String'),
'cities_guid')
```

```
cities_guid geom: Point(EPSG:4326), guid: String, name: String
```

Change Field definitions of a Schema

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Schema updatedSchema = schema.changeFields(
        [
            (schema.field('id'))   : new Field('guid','String'),
            (schema.field('name')) : new Field('description','String')
        ], 'cities_updated')
```

```
cities_updated geom: Point(EPSG:4326), guid: String, description: String
```

Add a Field to a Schema

```
Schema schema = new Schema("countries", [
        new Field("geom", "Polygon", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Schema updatedSchema = schema.addField(new Field("area", "Double"), "countries_area")
```

```
countries_area geom: Polygon(EPSG:4326), id: Integer, name: String, area: Double
```

Add a List of Fields to a Schema

```
Schema schema = new Schema("countries", [
        new Field("geom", "Polygon", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Schema updatedSchema = schema.addFields([
        new Field("area", "Double"),
        new Field("perimeter", "Double"),
], "countries_areaperimeter")
```

```
countries_areaperimeter geom: Polygon(EPSG:4326), id: Integer, name: String, area:
Double, perimeter: Double
```

Remove a Field from a Schema

```
Schema schema = new Schema("countries", [
        new Field("geom", "Polygon", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String"),
        new Field("area", "Double")
])
Schema updatedSchema = schema.removeField(schema.field("area"), "countries_updated")
```

```
countries_updated geom: Polygon(EPSG:4326), id: Integer, name: String
```

Remove a List of Fields from a Schema

```
Schema schema = new Schema("countries", [
        new Field("geom", "Polygon", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String"),
        new Field("area", "Double")
])
Schema updatedSchema = schema.removeFields([
        schema.field("area"),
        schema.field("name")
], "countries_updated")
```

```
countries_updated geom: Polygon(EPSG:4326), id: Integer
```

Create a new Schema from an existing Schema but only including a subset of Fields

```
Schema schema = new Schema("countries", [
        new Field("geom", "Polygon", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String"),
        new Field("area", "Double")
])
Schema updatedSchema = schema.removeFields([
        schema.field("area"),
        schema.field("name")
], "countries_updated")
```

```
countries_updated geom: Polygon(EPSG:4326), name: String
```

Combining Schemas

Combining two Schemas results in a Map with two values: schema and fields. The schema property contains the new Schema. The fields property is List of two Maps which both contain a mapping between the fields of the original Schema and the newly created Schema.

Optional arguments to the Schema.addSchema method are:

- postfixAll: Whether to postfix all field names (true) or not (false). If true, all Fields from the this current Schema will have '1' at the end of their name while the other Schema's Fields will have '2'. Defaults to false.
- includeDuplicates: Whether or not to include duplicate fields names. Defaults to false. If a duplicate is found a '2' will be added.
- maxFieldNameLength: The maximum new Field name length (mostly to support shapefiles where Field names can't be longer than 10 characters
- firstPostfix: The postfix string (default is '1') for Fields from the current Schema. Only applicable when postfixAll or includeDuplicates is true.
- secondPostfix: The postfix string (default is '2') for Fields from the other Schema. Only applicable when postfixAll or includeDuplicates is true.

Combine two Schemas with no duplicate fields and no postfixes to field names

```
Schema shopSchema = new Schema("shops", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])

Schema cafeSchema = new Schema("cafes", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String"),
        new Field("address", "String")
])

Map result = shopSchema.addSchema(cafeSchema, "business")

Schema combinedSchema = result.schema
println combinedSchema
```

```
business geom: Point(EPSG:4326), id: Integer, name: String, address: String
```

```
Map<String,String> shopSchemaFieldMapping = result.fields[0]
println shopSchemaFieldMapping
```

```
[geom:geom, id:id, name:name]
```

```
Map<String,String> cafeSchemaSchemaFieldMapping = result.fields[1]
println cafeSchemaSchemaFieldMapping
```

```
[address:address]
```

Combine two Schemas with no duplicate fields and postfixes

```
Schema shopSchema = new Schema("shops", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])

Schema cafeSchema = new Schema("cafes", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String"),
        new Field("address", "String")
])

Map result = shopSchema.addSchema(cafeSchema, "business", postfixAll: true,
includeDuplicates: false)

Schema combinedSchema = result.schema
println combinedSchema
```

```
business geom: Point(EPSG:4326), id1: Integer, name1: String, id2: Integer, name2:
String, address2: String
```

```
Map<String,String> shopSchemaFieldMapping = result.fields[0]
println shopSchemaFieldMapping
```

```
[geom:geom, id:id1, name:name1]
```

```
Map<String,String> cafeSchemaSchemaFieldMapping = result.fields[1]
println cafeSchemaSchemaFieldMapping
```

```
[id:id2, name:name2, address:address2]
```

Creating Features from a Schema

Create a Feature from a Schema with a Map of values

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = schema.feature([
        id: 1,
        name: 'Seattle',
        geom: new Point( -122.3204, 47.6024)
], "city.1")
println feature
```

```
cities.city.1 geom: POINT (-122.3204 47.6024), id: 1, name: Seattle
```

Create a Feature from a Schema with a List of values. The order of the values must match the order of the Fields.

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = schema.feature([
        new Point( -122.3204, 47.6024),
        1,
        'Seattle'
], "city.1")
println feature
```

```
cities.city.1 geom: POINT (-122.3204 47.6024), id: 1, name: Seattle
```

Create a Feature from a Schema with another Feature.

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature1 = new Feature([
        id: 1,
        name: 'Seattle',
        geom: new Point( -122.3204, 47.6024)
], "city.1", schema)
println feature1
Feature feature2 = schema.feature(feature1)
println feature2
```

```
cities.city.1 geom: POINT (-122.3204 47.6024), id: 1, name: Seattle
cities.city.1 geom: POINT (-122.3204 47.6024), id: 1, name: Seattle
```

Create an empty Feature from a Schema.

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = schema.feature()
println feature
```

```
cities.fid--60aba1d4_178616fae06_-58d1 geom: null, id: null, name: null
```

Reading and Writing Schemas

The Schema IO classes are in the **geoscript.feature.io** package.

Finding Schema Writer and Readers

List all Schema Writers

```
List<SchemaWriter> writers = SchemaWriters.list()
writers.each { SchemaWriter writer ->
    println writer.class.simpleName
}
```

```
JsonSchemaWriter
StringSchemaWriter
XmlSchemaWriter
```

Find a Schema Writer

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])

SchemaWriter writer = SchemaWriters.find("string")
String schemaStr = writer.write(schema)
println schemaStr
```

```
geom:Point:srid=4326,id:Integer,name:String
```

List all Schema Readers

```
List<SchemaReader> readers = SchemaReaders.list()
readers.each { SchemaReader reader ->
    println reader.class.simpleName
}
```

```
JsonSchemaReader
StringSchemaReader
XmlSchemaReader
```

Find a Schema Reader

```
SchemaReader reader = SchemaReaders.find("string")
Schema schema = reader.read("geom:Point:srid=4326,id:Integer,name:String")
println schema
```

```
layer geom: Point(EPSG:4326), id: Integer, name: String
```

String

Read a Schema from a String

```
StringSchemaReader reader = new StringSchemaReader()
Schema schema = reader.read("geom:Point:srid=4326,id:Integer,name:String", name:
"points")
println schema
```

```
points geom: Point(EPSG:4326), id: Integer, name: String
```

Write a Schema to a String

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])

StringSchemaWriter writer = new StringSchemaWriter()
String schemaStr = writer.write(schema)
println schemaStr
```

```
geom:Point:srid=4326,id:Integer,name:String
```

JSON

Read a Schema from a JSON

```
        JsonSchemaReader reader = new JsonSchemaReader()
        Schema schema = reader.read("""{
    "name": "cities",
    "projection": "EPSG:4326",
    "geometry": "geom",
    "fields": [
        {
            "name": "geom",
            "type": "Point",
            "geometry": true,
            "projection": "EPSG:4326"
        },
        {
            "name": "id",
            "type": "Integer"
        },
        {
            "name": "name",
            "type": "String"
        }
    ]
}""")
        println schema
```

```
cities geom: Point(EPSG:4326), id: Integer, name: String
```

Write a Schema to a JSON

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])

JsonSchemaWriter writer = new JsonSchemaWriter()
String schemaStr = writer.write(schema)
println schemaStr
```

```
{
    "name": "cities",
    "projection": "EPSG:4326",
    "geometry": "geom",
    "fields": [
        {
            "name": "geom",
            "type": "Point",
            "geometry": true,
            "projection": "EPSG:4326"
        },
        {
            "name": "id",
            "type": "Integer"
        },
        {
            "name": "name",
            "type": "String"
        }
    ]
}
```

XML

Read a Schema from a XML

```
        XmlSchemaReader reader = new XmlSchemaReader()
        Schema schema = reader.read("""<schema>
  <name>cities</name>
  <projection>EPSG:4326</projection>
  <geometry>geom</geometry>
  <fields>
    <field>
      <name>geom</name>
      <type>Point</type>
      <projection>EPSG:4326</projection>
    </field>
    <field>
      <name>id</name>
      <type>Integer</type>
    </field>
    <field>
      <name>name</name>
      <type>String</type>
    </field>
  </fields>
</schema>""")
        println schema
```

```
cities geom: Point(EPSG:4326), id: Integer, name: String
```

Write a Schema to a XML

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])

XmlSchemaWriter writer = new XmlSchemaWriter()
String schemaStr = writer.write(schema)
println schemaStr
```

```
<schema>
  <name>cities</name>
  <projection>EPSG:4326</projection>
  <geometry>geom</geometry>
  <fields>
    <field>
      <name>geom</name>
      <type>Point</type>
      <projection>EPSG:4326</projection>
    </field>
    <field>
      <name>id</name>
      <type>Integer</type>
    </field>
    <field>
      <name>name</name>
      <type>String</type>
    </field>
  </fields>
</schema>
```

Creating Features

Create an empty Feature from a Map of values and a Schema.

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        id: 1,
        name: "Seattle",
        geom: new Point(-122.3204, 47.6024)
], "city.1", schema)
println feature
```

```
cities.city.1 geom: POINT (-122.3204 47.6024), id: 1, name: Seattle
```

Create an empty Feature from a List of values and a Schema.

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        new Point(-122.3204, 47.6024),
        1,
        "Seattle"
], "city.1", schema)
println feature
```

```
cities.city.1 geom: POINT (-122.3204 47.6024), id: 1, name: Seattle
```

Create an empty Feature from a Map of values. The Schema is inferred from the values.

```
Feature feature = new Feature([
  id: 1,
  name: "Seattle",
  geom: new Point(-122.3204, 47.6024)
], "city.1")
println feature
```

```
feature.city.1 id: 1, name: Seattle, geom: POINT (-122.3204 47.6024)
```

Getting Feature Properties

Get a Feature's ID

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        new Point(-122.3204, 47.6024),
        1,
        "Seattle"
], "city.1", schema)

String id = feature.id
println id
```

```
city.1
```

Get a Feature's Geometry

```
Geometry geometry = feature.geom
println geometry
```

```
POINT (-122.3204 47.6024)
```

Get a Feature's Bounds

```
Bounds bounds = feature.bounds
println bounds
```

```
(-122.3204,47.6024,-122.3204,47.6024,EPSG:4326)
```

Get a Feature's attributes

```
Map attributes = feature.attributes
println attributes
```

```
[geom:POINT (-122.3204 47.6024), id:1, name:Seattle]
```

Getting Feature Attributes

Get an attribute from a Feature using a Field name

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        new Point(-122.3204, 47.6024),
        1,
        "Seattle"
], "city.1", schema)

int id = feature.get("id")
println id
```

```
1
```

Get an attribute from a Feature using a Field

```
String name = feature.get(schema.field("name"))
println name
```

```
Seattle
```

Set an attribute of a Feature using a Field name and a new value

```
feature.set("name", "Tacoma")
println feature["name"]
```

```
Tacoma
```

Set an attribute of a Feature using a Field and a new value

```
feature.set(schema.field("name"), "Mercer Island")
println feature["name"]
```

```
Mercer Island
```

Set attributes of a Feature using a Map of new values

```
feature.set([id: 2])
println feature["id"]
```

```
2
```

Set a new Geometry value

```
feature.geom = new Point(-122.2220, 47.5673)
println feature.geom
```

```
POINT (-122.222 47.5673)
```

Reading and Writing Features

The Feature IO classes are in the **geoscript.feature.io** package.

Finding Feature Writer and Readers

List all Feature Writers

```
List<Writer> writers = Writers.list()
writers.each { Writer writer ->
    println writer.class.simpleName
}
```

```
GeobufWriter
GeoJSONWriter
GeoRSSWriter
GmlWriter
GpxWriter
KmlWriter
```

Find a Feature Writer

```
Writer writer = Writers.find("geojson")
println writer.class.simpleName
```

```
GeoJSONWriter
```

List all Feature Readers

```
List<Reader> readers = Readers.list()
readers.each { Reader reader ->
    println reader.class.simpleName
}
```

```
GeobufReader
GeoJSONReader
GeoRSSReader
GmlReader
GpxReader
KmlReader
```

Find a Feature Reader

```
Reader reader = Readers.find("geojson")
println reader.class.simpleName
```

```
GeoJSONReader
```

GeoJSON

Get a GeoJSON String from a Feature

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        new Point(-122.3204, 47.6024),
        1,
        "Seattle"
], "city.1", schema)

String geojson = feature.geoJSON
println geojson
```

```
{"type":"Feature","geometry":{"type":"Point","coordinates":[-
122.3204,47.6024]},"properties":{"id":1,"name":"Seattle"},"id":"city.1"}
```

Write a Feature to GeoJSON

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        new Point(-122.3204, 47.6024),
        1,
        "Seattle"
], "city.1", schema)

GeoJSONWriter writer = new GeoJSONWriter()
String geojson = writer.write(feature)
println geojson
```

```
{"type":"Feature","geometry":{"type":"Point","coordinates":[-
122.3204,47.6024]},"properties":{"id":1,"name":"Seattle"},"id":"city.1"}
```

Get a Feature from GeoJSON

```
String geojson = '{"type":"Feature","geometry":{"type":"Point","coordinates":[-
122.3204,47.6024]},"properties":{"id":1,"name":"Seattle"},"id":"city.1"}'
Feature feature = Feature.fromGeoJSON(geojson)
println feature
```

```
feature.city.1 id: 1, name: Seattle, geometry: POINT (-122.3204 47.6024)
```

Read a Feature from GeoJSON

```
GeoJSONReader reader = new GeoJSONReader()
String geojson = '{"type":"Feature","geometry":{"type":"Point","coordinates":[-
122.3204,47.6024]},"properties":{"id":1,"name":"Seattle"},"id":"city.1"}'
Feature feature = reader.read(geojson)
println feature
```

```
feature.city.1 id: 1, name: Seattle, geometry: POINT (-122.3204 47.6024)
```

GeoBuf

Get a GeoBuf String from a Feature

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        new Point(-122.3204, 47.6024),
        1,
        "Seattle"
], "city.1", schema)

String geobuf = feature.geobuf
println geobuf
```

```
0a0269640a046e616d65100218062a1d0a0c08001a089fd8d374c0ebb22d6a0218016a090a075365617474
6c65
```

Get a Feature from a GeoBuf String

```
String geobuf =
'0a0269640a046e616d65100218062a1d0a0c08001a089fd8d374c0ebb22d6a0218016a090a07536561747
46c65'
Feature feature = Feature.fromGeobuf(geobuf)
println feature
```

```
features.0 geom: POINT (-122.3204 47.6024), id: 1, name: Seattle
```

Write a Feature to a GeoBuf String

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        new Point(-122.3204, 47.6024),
        1,
        "Seattle"
], "city.1", schema)

GeobufWriter writer = new GeobufWriter()
String geobuf = writer.write(feature)
println geobuf
```

```
0a0269640a046e616d65100218062a1d0a0c08001a089fd8d374c0ebb22d6a0218016a090a075365617474
6c65
```

Read a Feature from a GeoBuf String

```
GeobufReader reader = new GeobufReader()
String geobuf =
'0a0269640a046e616d65100218062a1d0a0c08001a089fd8d374c0ebb22d6a0218016a090a07536561747
46c65'
Feature feature = reader.read(geobuf)
println feature
```

```
features.0 geom: POINT (-122.3204 47.6024), id: 1, name: Seattle
```

GeoRSS

Get a GeoRSS String from a Feature

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        new Point(-122.3204, 47.6024),
        1,
        "Seattle"
], "city.1", schema)

String georss = feature.geoRSS
println georss
```

```
<entry xmlns:georss='http://www.georss.org/georss'
xmlns='http://www.w3.org/2005/Atom'><title>city.1</title><summary
xmlns:georss='http://www.georss.org/georss'
xmlns='http://www.w3.org/2005/Atom'>[geom:POINT (-122.3204 47.6024), id:1,
name:Seattle]</summary><updated>Tue Mar 23 16:34:05 PDT 2021</updated><georss:point
xmlns:georss='http://www.georss.org/georss'
xmlns='http://www.w3.org/2005/Atom'>47.6024 -122.3204</georss:point></entry>
```

Get a Feature from a GeoRSS String

```
        String georss = """<entry xmlns:georss='http://www.georss.org/georss'
xmlns='http://www.w3.org/2005/Atom'>
    <title>city.1</title>
    <summary>[geom:POINT (-122.3204 47.6024), id:1, name:Seattle]</summary>
    <updated>Sat Jan 28 15:51:47 PST 2017</updated>
    <georss:point>47.6024 -122.3204</georss:point>
</entry>
"""
        Feature feature = Feature.fromGeoRSS(georss)
        println feature
```

```
georss.fid--60aba1d4_178616fae06_-58d5 title: city.1, summary: [geom:POINT (-122.3204
47.6024), id:1, name:Seattle], updated: Sat Jan 28 15:51:47 PST 2017, geom: POINT (-
122.3204 47.6024)
```

Write a Feature to a GeoRSS String

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        new Point(-122.3204, 47.6024),
        1,
        "Seattle"
], "city.1", schema)

GeoRSSWriter writer = new GeoRSSWriter()
String georss = writer.write(feature)
println georss
```

```
<entry xmlns:georss='http://www.georss.org/georss'
xmlns='http://www.w3.org/2005/Atom'><title>city.1</title><summary
xmlns:georss='http://www.georss.org/georss'
xmlns='http://www.w3.org/2005/Atom'>[geom:POINT (-122.3204 47.6024), id:1,
name:Seattle]</summary><updated>Tue Mar 23 16:34:05 PDT 2021</updated><georss:point
xmlns:georss='http://www.georss.org/georss'
xmlns='http://www.w3.org/2005/Atom'>47.6024 -122.3204</georss:point></entry>
```

Read a Feature from a GeoRSS String

```
        GeoRSSReader reader = new GeoRSSReader()
        String georss = """<entry xmlns:georss='http://www.georss.org/georss'
xmlns='http://www.w3.org/2005/Atom'>
    <title>city.1</title>
    <summary>[geom:POINT (-122.3204 47.6024), id:1, name:Seattle]</summary>
    <updated>Sat Jan 28 15:51:47 PST 2017</updated>
    <georss:point>47.6024 -122.3204</georss:point>
</entry>
"""
        Feature feature = reader.read(georss)
        println feature
```

```
georss.fid--60aba1d4_178616fae06_-58d3 title: city.1, summary: [geom:POINT (-122.3204
47.6024), id:1, name:Seattle], updated: Sat Jan 28 15:51:47 PST 2017, geom: POINT (-
122.3204 47.6024)
```

GML

Get a GML String from a Feature

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        new Point(-122.3204, 47.6024),
        1,
        "Seattle"
], "city.1", schema)

String gml = feature.gml
println gml
```

```
<gsf:cities xmlns:gsf="http://geoscript.org/feature"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" fid="city.1">
<gml:name>Seattle</gml:name>
<gsf:geom>
<gml:Point>
<gml:coord>
<gml:X>-122.3204</gml:X>
<gml:Y>47.6024</gml:Y>
</gml:coord>
</gml:Point>
</gsf:geom>
<gsf:id>1</gsf:id>
</gsf:cities>
```

Get a Feature from a GML String

```
        String gml = """<gsf:cities xmlns:gsf="http://geoscript.org/feature"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" fid="city.1">
    <gml:name>Seattle</gml:name>
    <gsf:geom>
        <gml:Point>
            <gml:coord>
                <gml:X>-122.3204</gml:X>
                <gml:Y>47.6024</gml:Y>
            </gml:coord>
        </gml:Point>
    </gsf:geom>
    <gsf:id>1</gsf:id>
</gsf:cities>
"""
        Feature feature = Feature.fromGml(gml)
        println feature
```

```
feature.city.1 name: Seattle, id: 1, geom: POINT (-122.3204 47.6024)
```

Write a Feature to a GML String

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        new Point(-122.3204, 47.6024),
        1,
        "Seattle"
], "city.1", schema)

GmlWriter writer = new GmlWriter()
String gml = writer.write(feature)
println gml
```

```
<gsf:cities xmlns:gsf="http://geoscript.org/feature"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" fid="city.1">
<gml:name>Seattle</gml:name>
<gsf:geom>
<gml:Point>
<gml:coord>
<gml:X>-122.3204</gml:X>
<gml:Y>47.6024</gml:Y>
</gml:coord>
</gml:Point>
</gsf:geom>
<gsf:id>1</gsf:id>
</gsf:cities>
```

Read a Feature from a GML String

```
        GmlReader reader = new GmlReader()
        String gml = """<gsf:cities xmlns:gsf="http://geoscript.org/feature"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" fid="city.1">
    <gml:name>Seattle</gml:name>
    <gsf:geom>
        <gml:Point>
            <gml:coord>
                <gml:X>-122.3204</gml:X>
                <gml:Y>47.6024</gml:Y>
            </gml:coord>
        </gml:Point>
    </gsf:geom>
    <gsf:id>1</gsf:id>
</gsf:cities>
"""
        Feature feature = reader.read(gml)
        println feature
```

```
feature.city.1 name: Seattle, id: 1, geom: POINT (-122.3204 47.6024)
```

GPX

Get a GPX String from a Feature

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        new Point(-122.3204, 47.6024),
        1,
        "Seattle"
], "city.1", schema)

String gpx = feature.gpx
println gpx
```

```
<wpt lat='47.6024' lon='-122.3204'
xmlns='http://www.topografix.com/GPX/1/1'><name>city.1</name></wpt>
```

Get a Feature from a GPX String

```
String gpx = "<wpt lat='47.6024' lon='-122.3204'
xmlns='http://www.topografix.com/GPX/1/1'><name>city.1</name></wpt>"
Feature feature = Feature.fromGpx(gpx)
println feature
```

```
gpx.fid--60aba1d4_178616fae06_-58d2 geom: POINT (-122.3204 47.6024), name: city.1
```

Write a Feature to a GPX String

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        new Point(-122.3204, 47.6024),
        1,
        "Seattle"
], "city.1", schema)

GpxWriter writer = new GpxWriter()
String gpx = writer.write(feature)
println gpx
```

```
<wpt lat='47.6024' lon='-122.3204'
xmlns='http://www.topografix.com/GPX/1/1'><name>city.1</name></wpt>
```

Read a Feature from a GPX String

```
GpxReader reader = new GpxReader()
String gpx = "<wpt lat='47.6024' lon='-122.3204'
xmlns='http://www.topografix.com/GPX/1/1'><name>city.1</name></wpt>"
Feature feature = reader.read(gpx)
println feature
```

```
gpx.fid--60aba1d4_178616fae06_-58d4 geom: POINT (-122.3204 47.6024), name: city.1
```

KML

Get a KML String from a Feature

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        new Point(-122.3204, 47.6024),
        1,
        "Seattle"
], "city.1", schema)

String kml = feature.kml
println kml
```

```
<kml:Placemark xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:kml="http://earth.google.com/kml/2.1"
id="city.1"><kml:name>Seattle</kml:name><kml:Point><kml:coordinates>-
122.3204,47.6024</kml:coordinates></kml:Point></kml:Placemark>
```

Get a Feature from a KML String

```
        String kml = """<kml:Placemark xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:kml="http://earth.google.com/kml/2.1" id="city.1">
    <kml:name>Seattle</kml:name>
    <kml:Point>
        <kml:coordinates>-122.3204,47.6024</kml:coordinates>
    </kml:Point>
</kml:Placemark>"""
        Feature feature = Feature.fromKml(kml)
        println feature
```

```
placemark.city.1 name: Seattle, description: null, Geometry: POINT (-122.3204 47.6024)
```

Write a Feature to a KML String

```
Schema schema = new Schema("cities", [
        new Field("geom", "Point", "EPSG:4326"),
        new Field("id", "Integer"),
        new Field("name", "String")
])
Feature feature = new Feature([
        new Point(-122.3204, 47.6024),
        1,
        "Seattle"
], "city.1", schema)

KmlWriter writer = new KmlWriter()
String kml = writer.write(feature)
println kml
```

```
<kml:Placemark xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:kml="http://earth.google.com/kml/2.1"
id="city.1"><kml:name>Seattle</kml:name><kml:Point><kml:coordinates>-
122.3204,47.6024</kml:coordinates></kml:Point></kml:Placemark>
```

Read a Feature from a KML String

```
        KmlReader reader = new KmlReader()
        String kml = """<kml:Placemark xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:kml="http://earth.google.com/kml/2.1" id="city.1">
    <kml:name>Seattle</kml:name>
    <kml:Point>
        <kml:coordinates>-122.3204,47.6024</kml:coordinates>
    </kml:Point>
</kml:Placemark>"""
        Feature feature = reader.read(kml)
        println feature
```

```
placemark.city.1 name: Seattle, description: null, Geometry: POINT (-122.3204 47.6024)
```