

# Table of Contents

Feature Recipes .....	1
Creating Fields .....	1
Creating Schemas .....	2
Getting Schema Properties .....	3
Getting Schema Fields .....	4
Modifying Schemas .....	5
Creating Features from a Schema .....	8

# Feature Recipes

## 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 polygonSchema = 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
```

## 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-7b0ac272_159c35a7cb1_-8000 geom: null, id: null, name: null
```