Going Live with Enterprise Solutions

TopQuadrant™

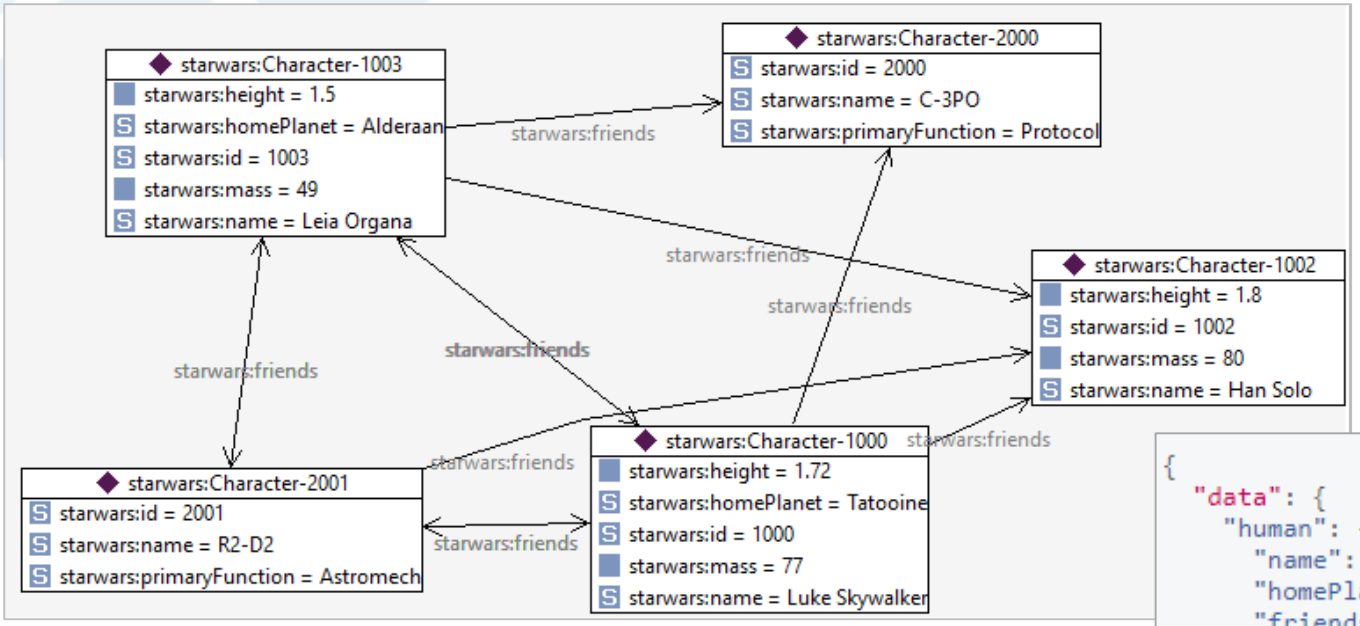
TopBraid GraphQL

Bringing the Worlds of JSON and RDF Together

Webinar, June 2018

Holger Knublauch

RDF Graphs to JSON Trees




```
{
  "data": {
    "human": {
      "name": "Luke Skywalker",
      "homePlanet": "Tatooine",
      "friends": [
        {
          "name": "Han Solo"
        },
        {
          "name": "Leia Organa"
        },
        {
          "name": "C-3PO"
        },
        {
          "name": "R2-D2"
        }
      ]
    }
  }
}
```

Overview

- What is GraphQL?
- How to bring together RDF technologies and GraphQL?
- GraphQL queries over RDF graphs
- GraphQL mutations (updates)
- JSON to RDF

- Questions

GraphQL (General Demo)

GraphiQL ▶ Prettify History

```
1 {
2   human(id: "1000") {
3     name
4     homePlanet
5     friends {
6       name
7     }
8   }
9 }
```

QUERY VARIABLES

```
{
  "data": {
    "human": {
      "name": "Luke Skywalker",
      "homePlanet": "Tatooine",
      "friends": [
        {
          "name": "Han Solo"
        },
        {
          "name": "Leia Organa"
        },
        {
          "name": "C-3PO"
        },
        {
          "name": "R2-D2"
        }
      ]
    }
  }
}
```

Schema **Query**

Search Query...

The query type, represents all of the entry points into our object graph

FIELDS

hero(episode: Episode): Character

reviews(episode: Episode!): [Review]

search(text: String): [SearchResult]

character(id: ID!): Character

droid(id: ID!): Droid

human(id: ID!): Human

starship(id: ID!): Starship

GraphQL Queries over RDF Data

TopBraid GraphiQL

Prettify

```
1 {
2   humans(id: "1000") {
3     name
4     homePlanet
5     friends {
6       name
7     }
8   }
9 }
```

QUERY VARIABLES

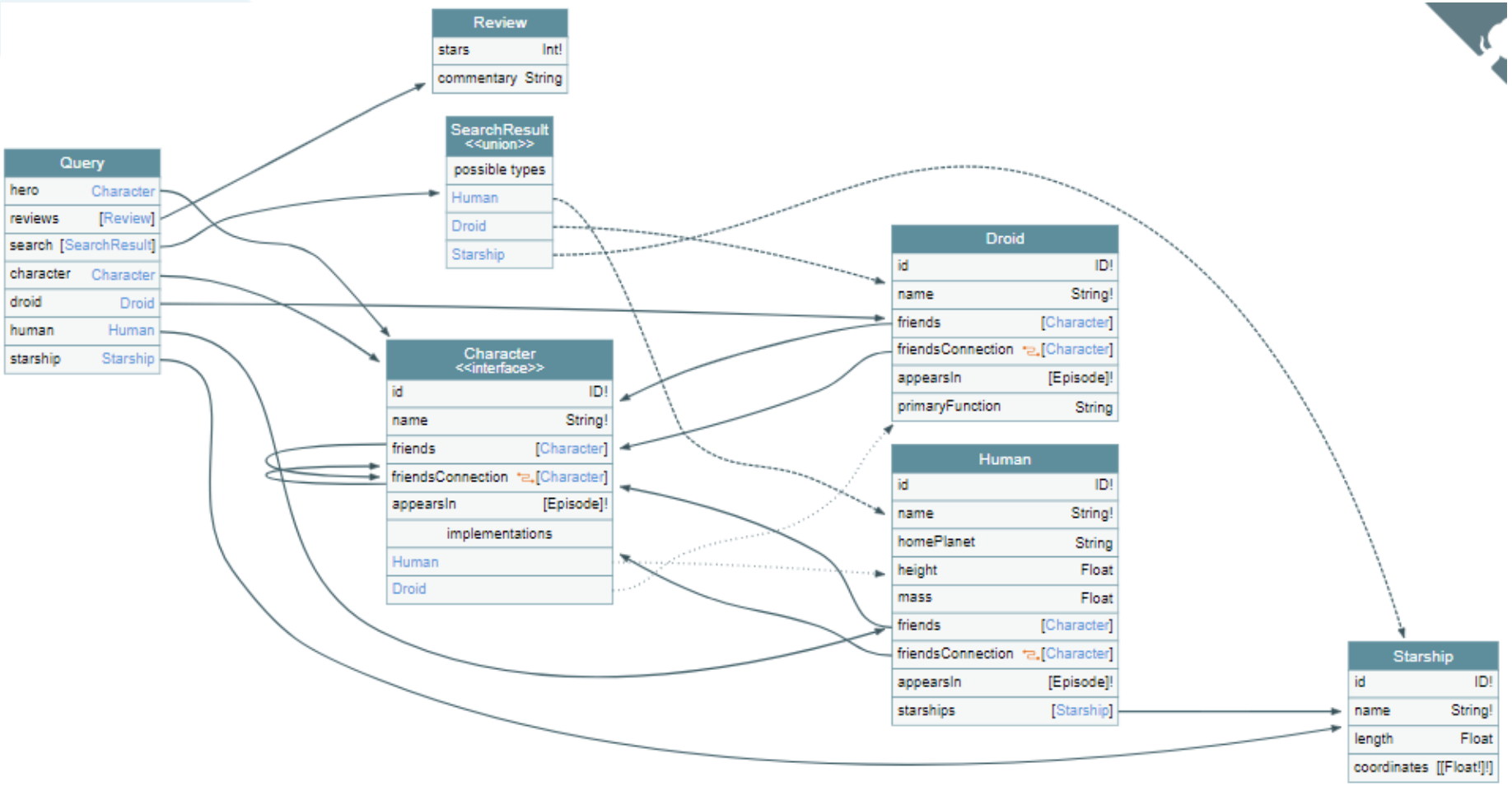
```
{
  "data": {
    "humans": [
      {
        "name": "Luke Skywalker",
        "homePlanet": "Tatooine",
        "friends": [
          {
            "name": "Leia Organa"
          },
          {
            "name": "C-3PO"
          },
          {
            "name": "R2-D2"
          },
          {
            "name": "Han Solo"
          }
        ]
      }
    ]
  }
}
```

Schema **RootRDFQuery**

```
humans(
  queryText: String
  filter: String
  uri: ID
  orderBy: Human_FieldsEnum
  orderByExpr: String
  orderByDesc: Boolean
  appearsIn: String
  friends: ID
  height: Float
  homePlanet: String
  id: String
  mass: Float
  name: String
  starships: ID
  where: Human_where
  first: Int
  skip: Int
): [Human]
```

A humanoid creature from the Star Wars

GraphQL Schemas



Star Wars Example Schema

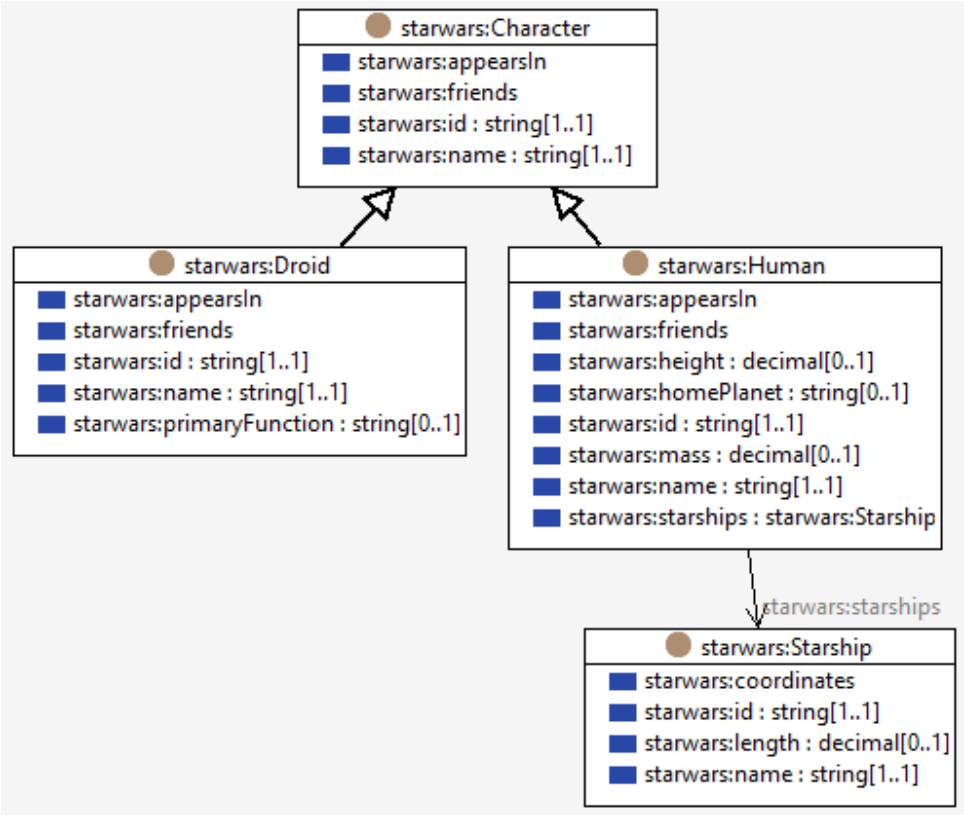
```
interface Character {
  appearsIn: [Episode]!
  friends: [Character]
  id: ID!
  name: String!
}

type Human implements Character {
  appearsIn: [Episode]!
  friends: [Character]
  height(unit: LengthUnit = METER): Float
  homePlanet: String
  id: ID!
  mass: Float
  name: String!
  starships: [Starship]
}
```

Motivations and Design Goals for Providing GraphQL Access to RDF Data

- Benefits:
  - JSON output
  - Simple to use (especially for UI developers)
  - Widely accepted
- Challenges:
  - While the “why” is clear and compelling, there are also some short comings (expressivity, etc.) and differences
- Design Goals
  - Integration should offer the best of both worlds

Shapes Constraint Language (SHACL)

- W3C Standard since 2017
- Schema Language for RDF
- Rich Semantic Constraints
- Shapes > Classes
- Views
- Paths



Alignment of GraphQL and SHACL – 1

Types vs RDF Datatypes

GraphQL Scalar Type	RDF Data Type
Boolean	xsd:boolean
Float	xsd:decimal
ID	xsd:string
Int	xsd:integer
String	xsd:string

Alignment of GraphQL and SHACL – 2

Object Types vs Shapes

Example GraphQL Schema

```
type Person {
   name: String
   friends: [Person]
}
```

Example RDF/SHACL

```
ex:Person
   a sh:NodeShape ;
   sh:property [
      sh:path ex:friends ;
      sh:node ex:Person ;
   ] ;
   sh:property [
      sh:path ex:name ;
      sh:datatype xsd:string ;
      sh:maxCount 1 ;
    ] .
```

Alignment of GraphQL and SHACL – 3

Interfaces & Simple Inheritance

Example GraphQL Schema

```
interface Character {
   id: ID!
}

type Human implements Character {
   id: ID!
   friends: [Character]
}
```

Example RDF/SHACL

```
ex:Character
   a sh:NodeShape ;
   graphql:isInterface true ;
   sh:property [
      sh:path ex:id ;
      sh:datatype xsd:string ;
      sh:maxCount 1 ;
      sh:minCount 1 ;
   ] .

ex:Human
   a sh:NodeShape ;
   sh:node ex:Character ;
   sh:property [
      sh:path ex:friends ;
      sh:node ex:Character ;
    ] ;
   sh:property [
      sh:path ex:id ;
      sh:datatype xsd:string ;
      sh:maxCount 1 ;
      sh:minCount 1 ;
   ] .
```

Alignment of GraphQL and SHACL – 4

Union Types

Example GraphQL Schema

```
type Human {
   name: String
}

type Starship {
   length: Int
}

union SearchResult = Human | Starship
```

Example RDF/SHACL

```
ex:Human
   a sh:NodeShape ;
   sh:property [
      sh:path ex:name ;
      sh:datatype xsd:string ;
      sh:maxCount 1 ;
   ] .

ex:Starship
   a sh:NodeShape ;
   sh:property [
      sh:path ex:length ;
      sh:datatype xsd:integer ;
      sh:maxCount 1 ;
   ] .

ex:SearchResult
   a sh:NodeShape ;
   sh:or (
      ex:Human
      ex:Starship
   ) .
```

Alignment of GraphQL and SHACL – 5

Enumerations

Example GraphQL Schema

```
type Unicorn {
   colors: Color
}

enum Color {
   # Yellow is our least favorite color
   YELLOW
   RED
   PINK
}
```

Example RDF/SHACL

```
ex:Unicorn
   a sh:NodeShape ;
   sh:property [
      sh:path ex:colors ;
      sh:node Color ;
      sh:maxCount 1 ;
   ] .

ex:Color
   a sh:NodeShape ;
   sh:in (
      "YELLOW"
      "RED"
      "PINK"
   ) .
```

Alignment of GraphQL and SHACL – 6

Semantic Constraints

Example GraphQL Schema

```
type Adult {
   age: Int  @shape(minInclusive: 18)
}
```

GraphQL Directives
(Extension Mechanism)

Example RDF/SHACL

```
ex:Adult
   a sh:NodeShape ;
   sh:property [
      sh:path ex:age ;
      sh:datatype xsd:integer ;
      sh:maxCount 1 ;
      sh:minInclusive 18 ;
   ] .
```

Alignment of GraphQL and SHACL – 7

Names and URIs

Example GraphQL Schema

```
schema
   @prefixes(
      human: "http://example.org/human/",
      starwars: "http://starwars.com/data/ (default)"
   )
{
   query: Query
}

type Human @uri(template: "human:{$id}") {
   id: ID!
...
```

Alignment of GraphQL and SHACL – 8

Linked Schemas

Example GraphQL Schema

```
schema
   @graph(
      uri: "http://starwars.com/data/",
      imports: [ "http://movies.org/data/" ]
   )
   @prefixes(
      starwars: "http://starwars.com/data/ (default)",
      movies: "http://movies.org/data/"
   )
...
type Actor {
   appearedIn: movies_Movie
   ...
}
```

Alignment of GraphQL and SHACL – 9

Display Metadata (SHACL)

Example RDF/SHACL

```
ex:NamesGroup
   a sh:PropertyGroup ;
   rdfs:label "Names" ;
   sh:order "0"^^xsd:decimal .

ex:AddressGroup
   a sh:PropertyGroup ;
   rdfs:label "Address" ;
   rdfs:label "Addresse"@de ;
   sh:order "1"^^xsd:decimal .

ex:Customer
   a sh:NodeShape ;
   sh:property [
      sh:path ex:firstName ;
      sh:datatype xsd:string ;
      sh:maxCount 1 ;
      sh:order "0"^^xsd:decimal ;
      sh:group ex:NamesGroup ;
      sh:name "given name" .
...
```

```
Names:
   given name:     Rolf-Michel
   family name:    Massin

Address:
   street:         9100 Oak Street
   zip code:       91823
   country:        USA
```

Alignment of GraphQL and SHACL – 9

Display Metadata (GraphQL)

```
Names:
   given name:     Rolf-Michel
   family name:    Massin

Address:
   street:         9100 Oak Street
   zip code:       91823
   country:        USA
```

Example GraphQL Schema

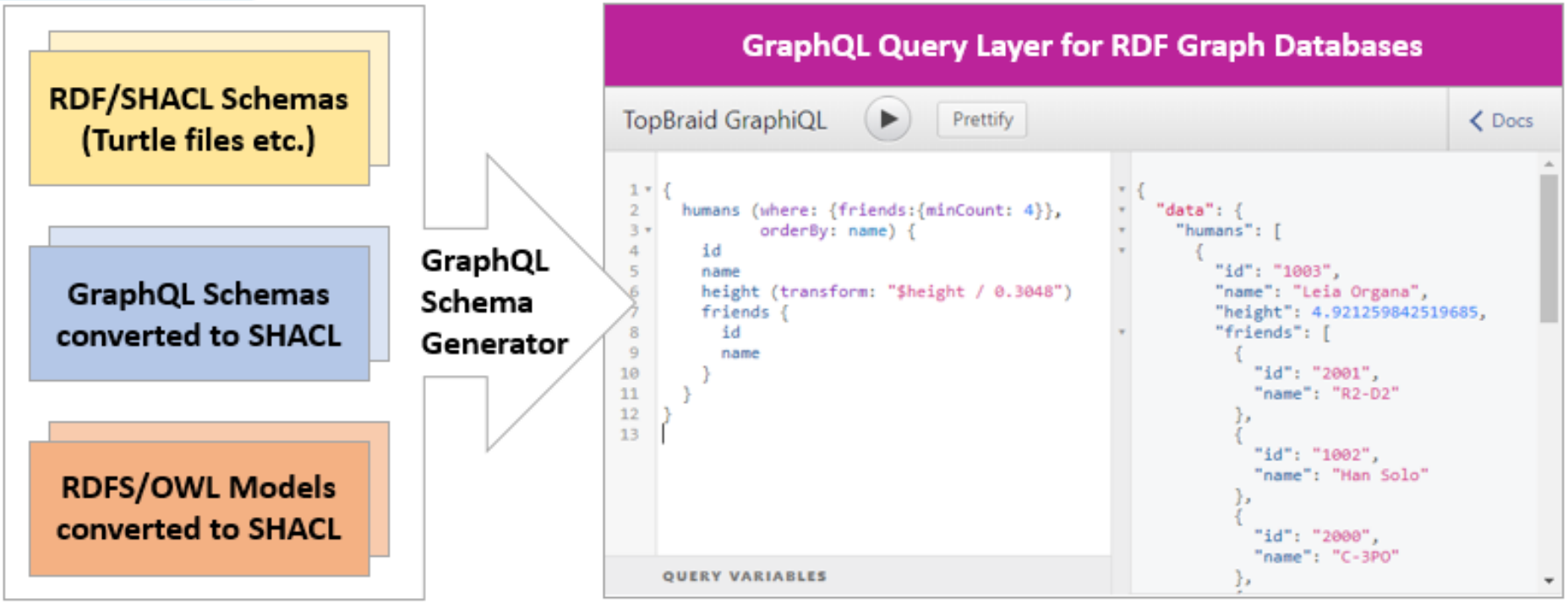
```
schema
   @groups(
      NamesGroup: {
         label: "Names"
      },
      AddressGroup: {
         label: "Address"
         label_de: "Addresse"
      }
   )
...

type Customer {
   firstName: String       @display(group: NamesGroup,    label: "given name")
   lastName: String        @display(group: NamesGroup,    label: "family name")
   street: String          @display(group: AddressGroup)
   postalCode: String      @display(group: AddressGroup, label: "zip code", label_de: "Postleitzahl")
   country: String         @display(group: AddressGroup, defaultValue: "USA")
}
```

GraphQL Schema + SHACL = Better Together

- Lots of similarities (types, fields, etc)
- GraphQL has a huge user base & tool support
- GraphQL has user-friendly syntaxes
- RDF is a flexible model for knowledge graphs
- RDF offers URIs and subclasses
- SHACL offers rich constraints & UI metadata

- All RDF knowledge graphs can be turned into GraphQL query endpoints with SHACL

GraphQL in TopBraid



Loading GraphQL Files in TopBraid Composer

TBCME-workflow2 - TopBraid - TopBraid/Examples/starwars/starwars.graphql - Eclipse Platform

File Edit Navigate Project Model Scripts System Inference TopBraid Applications Resource Window Help

starwars:Character | Quick Access

Classes | Shapes | Propert...

- starwars:Character
  - starwars:Droid
  - starwars:Human
- starwars:Episode (3)
- starwars:LengthUnit (2)
- starwars:Mutation
- starwars:Query
- starwars:Review
- starwars:Starship

Project Explorer | Basket

- starwars
  - fetch.js
  - starwars-all.ttl [http://topbrai
  - starwars-fetch.ui.ttlx [http://t
  - starwars-instances.ttl [http://t
  - starwars-original.graphql [htt
  - starwars.graphql [http://topb
- CreateReportSpreadsheets.ttl [htt
- geo-ontology.ttl [http://topbraid

starwars.graphql

Node Shape Form

Name: starwars:Character

Annotations

rdfs:comment

A character from the Star Wars universe

Constraints

sh:property

- starwars:Character-appearsIn
- starwars:Character-friends
- starwars:Character-id
- starwars:Character-name

Targets

Other Properties

dash:abstract

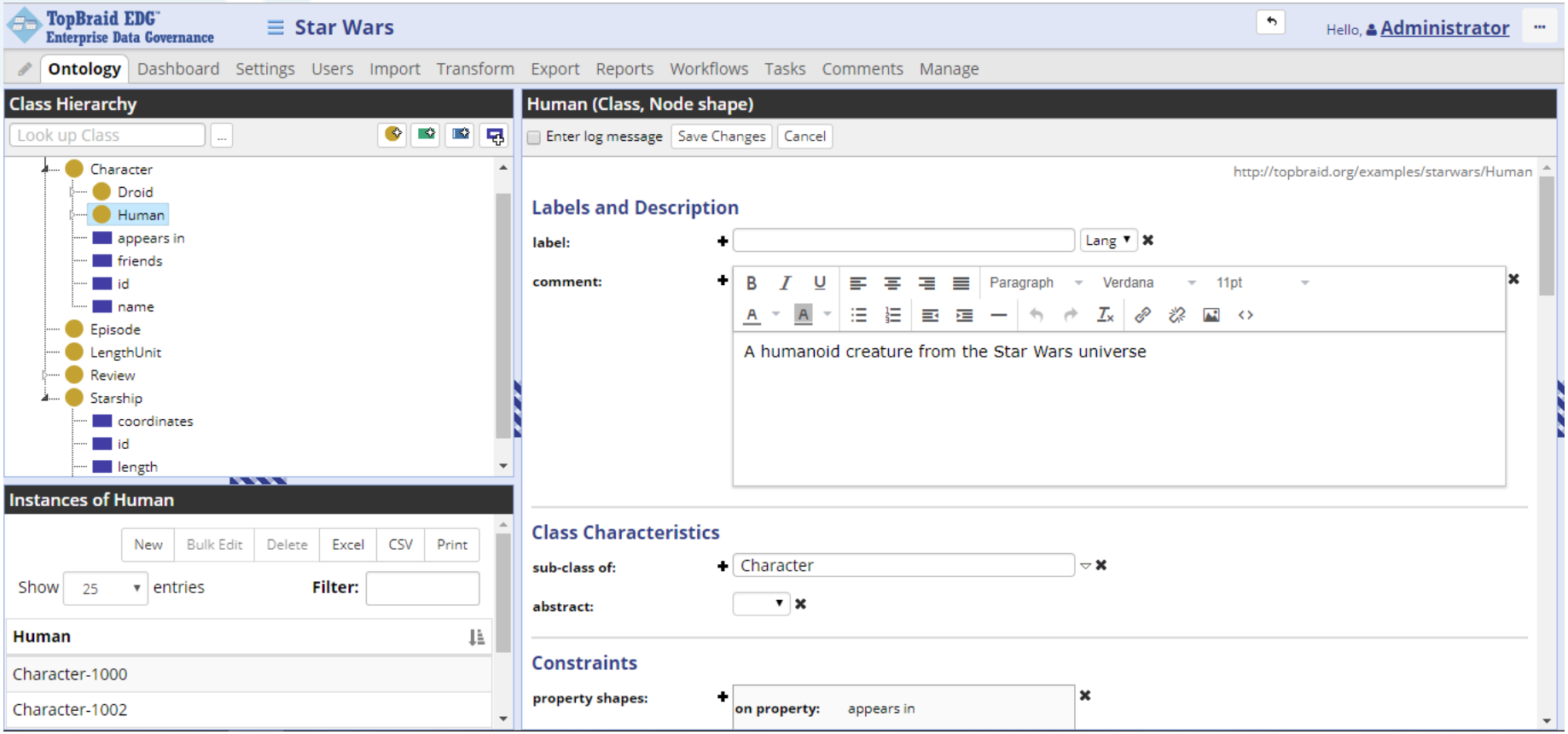
Form | Browser | Diagram | Graph | Form Layout | Source Code

Imports | Domain | Instances | Error Log | SPARQL | Text Search | Targets | SHACL Valid...

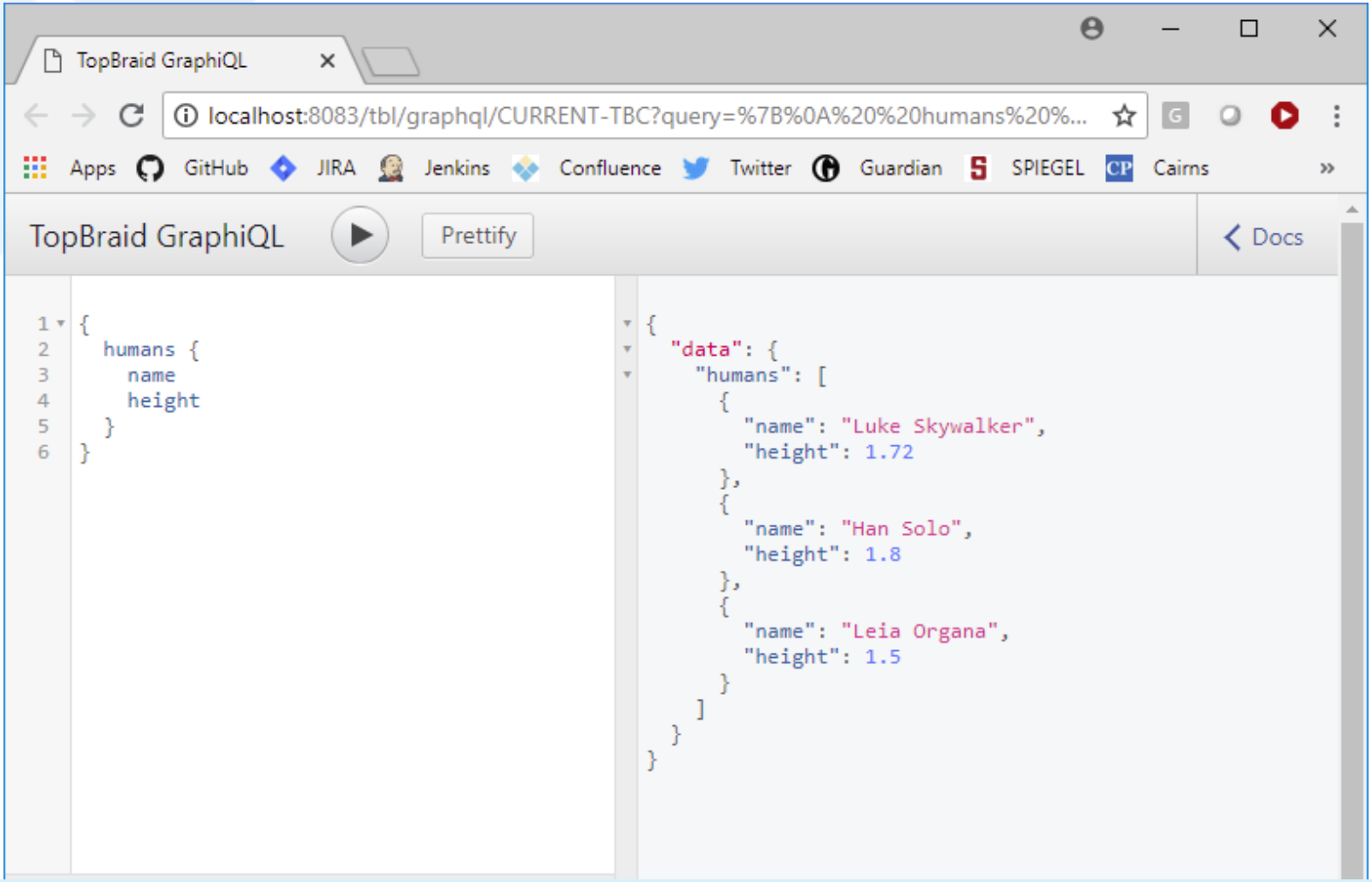
- http://datashapes.org/graphql (owl:imports from /TopBraid/SHACL/graphql.shapes.ttl)
  - http://datashapes.org/dash (owl:imports from /TopBraid/SHACL/dash.ttl)

TopBraid/Examples/starwars/starwars.graphql

SHACL/Ontology Editor in TopBraid EDG



GraphQL Queries in TopBraid



Available in all server products (e.g., TopBraid EDG) and in TBC-ME

Ordering

TopBraid GraphiQL

localhost:8083/tbl/graphql/CURRENT-TBC?query=%7B%0A%20%20%20%20huma...

Apps GitHub JIRA Jenkins Confluence Twitter Guardian SPIEGEL Cairns

TopBraid GraphiQL Prettify Docs

```
1 {
2     humans (orderBy: height, orderByDesc: true) {
3         name
4         height
5     }
6 }
```

QUERY VARIABLES

```
{
  "data": {
    "humans": [
      {
        "name": "Han Solo",
        "height": 1.8
      },
      {
        "name": "Luke Skywalker",
        "height": 1.72
      },
      {
        "name": "Leia Organa",
        "height": 1.5
      }
    ]
  }
}
```

Matching

TopBraid GraphiQL

localhost:8083/tbl/graphql/CURRENT-TBC?query=%7B%0A%20%20humans(id%3A%20...

Apps GitHub JIRA Jenkins Confluence Twitter Guardian SPIEGEL Cairns ABC

TopBraid GraphiQL Prettify Docs

```
1 {
2   humans(id: "1000") {
3     name
4     id
5   }
6 }
```

```
{
  "data": {
    "humans": [
      {
        "name": "Luke Skywalker",
        "id": "1000"
      }
    ]
  }
}
```

QUERY VARIABLES

Filtering

TopBraid GraphiQL

localhost:8083/tbl/graphql/CURRENT-TBC?query=%7B%0A%20%20humans%20(w...

Apps GitHub JIRA Jenkins Confluence Twitter Guardian SPIEGEL Cairns

TopBraid GraphiQL Prettify Docs

```
1  {
2    humans (where: {
3              height: {
4                minInclusive: 1.6
5              }
6            })
7    {
8      name
9      height
10   }
11 }
```

```
{
  "data": {
    "humans": [
      {
        "name": "Luke Skywalker",
        "height": 1.72
      },
      {
        "name": "Han Solo",
        "height": 1.8
      }
    ]
  }
}
```

QUERY VARIABLES

Filtering (nested)

TopBraid GraphiQL

localhost:8083/tbl/graphql/CURRENT-TBC?query=%7B%0A%20%20%20%20huma...

Apps GitHub JIRA Jenkins Confluence Twitter Guardian SPIEGEL Cairns

TopBraid GraphiQL Prettify Docs

```
{
    humans(where: {
            starships: {
                exists: {
                    length: {
                        minInclusive: 30
                    }
                }
            }
        }) {
        name
        height
        homePlanet
        starships {
            name
            length
        }
    }
}
```

QUERY VARIABLES

```
{
  "data": {
    "humans": [
      {
        "name": "Han Solo",
        "height": 1.8,
        "homePlanet": null,
        "starships": [
          {
            "name": "Millenium Falcon",
            "length": 34.37
          },
          {
            "name": "Imperial shuttle",
            "length": 20
          }
        ]
      }
    ]
  }
}
```

Filtering with SPARQL

TopBraid GraphiQL

localhost:8083/tbl/graphql/CURRENT-TBC?query=%7B%0A%20%20%20%20huma...

Apps GitHub JIRA Jenkins Confluence Twitter Guardian SPIEGEL Cairns

TopBraid GraphiQL Prettify Docs

```
{
    humans (filter: "$height > 1.6") {
        name
    }
}
```

```
{
  "data": {
    "humans": [
      {
        "name": "Luke Skywalker"
      },
      {
        "name": "Han Solo"
      }
    ]
  }
}
```

QUERY VARIABLES

Counting

TopBraid GraphiQL

localhost:8083/tbl/graphql/CURRENT-TBC?query=%7B%0A%20%20%20%20huma...

Apps GitHub JIRA Jenkins Confluence Twitter Guardian SPIEGEL Cairns

TopBraid GraphiQL Prettify Docs

```
1 {
2     humans_COUNT (where: {height: {minInclusive: 1.6}})
3 }
```

```
{
  "data": {
    "humans_COUNT": 2
  }
}
```

QUERY VARIABLES

Transforming

TopBraid GraphiQL

localhost:8083/tbl/graphql/CURRENT-TBC?query=%7B%0A%20%20%20%20huma...

Apps GitHub JIRA Jenkins Confluence Twitter Guardian SPIEGEL Cairns

TopBraid GraphiQL Prettify Docs

```
1 {
2     humans {
3         name (transform: "UCASE($name)")
4         height (transform: "$height * 3.28084")
5     }
6 }
```

QUERY VARIABLES

```
{
  "data": {
    "humans": [
      {
        "name": "LUKE SKYWALKER",
        "height": 5.6430448
      },
      {
        "name": "HAN SOLO",
        "height": 5.905512
      },
      {
        "name": "LEIA ORGANA",
        "height": 4.92126
      }
    ]
  }
}
```

Deriving Values

TopBraid GraphiQL

localhost:8083/tbl/graphql/CURRENT-TBC?query=%0A%7B%0A%20%20%20%20h...

Apps GitHub JIRA Jenkins Confluence Twitter Guardian SPIEGEL Cairns

TopBraid GraphiQL Prettify Docs

```
{
    humans {
        name
        nameLength: deriveInt(expr: "STRLEN($name)")
    }
}
```

QUERY VARIABLES

```
{
  "data": {
    "humans": [
      {
        "name": "Luke Skywalker",
        "nameLength": 14
      },
      {
        "name": "Han Solo",
        "nameLength": 8
      },
      {
        "name": "Leia Organa",
        "nameLength": 11
      }
    ]
  }
}
```

Query User Interface

TopBraid EDG Enterprise Data Governance

Northwind

Global Lookup

Hello, Administrator

Assets | Dashboard | Settings | Users | Import | Transform | Export | Reports | Workflows | Tasks | Comments | Manage

DASHBOARD

- Home
- Metrics Dashboard
- My Workflows
- Find Asset Collection
- Find Code
- Find Tasks

ASSET COLLECTIONS

- Glossaries
- Requirements Assets
- Big Data Assets
- Data Assets
- Datatypes
- Enumerations

Type: Database Column | Free Text | Advanced | Filters | Columns

is primary key: any... | any value

New | Details | Bulk Edit | Delete | Clone | Export

Show 25 entries

Filter:

Database Column	type	total number of values	is primary key
CATEGORYID (NORTHWIND.DBO.CATEGORIES)	Database Column	8	true
CUSTOMERID (NORTHWIND.DBO.CUSTOMERCUSTOMERDEMO)	Database Column	0	true
CUSTOMERID (NORTHWIND.DBO.CUSTOMERS)	Database Column	91	true
CUSTOMERTYPEID (NORTHWIND.DBO.CUSTOM	Database Column	0	true
CUSTOMERTYPEID (NORTHWIND.DBO.CUSTOM			
EMPLOYEEID (NORTHWIND.DBO.EMPLOYEES)			
EMPLOYEEID (NORTHWIND.DBO.EMPLOYEETER			
ORDERID (NORTHWIND.DBO.ORDER DETAILS)			

```
{
  results: databaseColumns(where: {isPrimaryKey: {minCount: 1}}) {
    uri
    label
    type {
      uri
      label
    }
    nonNullValuesCount
    isPrimaryKey
  }
}
```

GraphQL Mutations (Updates)

Example GraphQL Query

```
mutation {
   createHuman (input: {
      uri: "http://example.org/Humans/123",
      id: "123",
      name: "Darth Vader"
   })
   updateHuman (input: {
      uri: "http://example.org/Humans/456",
      father: {
         uri: "http://example.org/Humans/123"
      }
   })
   results {
      addedCount
      deletedCount
   }
   commit (message: "Added Luke's dad")
}
```

Example JSON Result

```
{
   "data": {
      "createHuman": true,
      "updateHuman": true,
      "results": {
         "addedCount": 4,
         "deletedCount": 0
      },
      "commit": "Added Luke's dad"
   }
}
```

JSON to RDF Conversion

```
{
   "human": {
      "id": "HAN",
      "name": "Han Solo",
      "friends": [
         { "id": "LEIA" },
         { "id": "LUKE" }
      ]
   }
}
```

```
{
   "human": {
      "id": "LUKE",
      "name": "Luke Skywalker",
      "friends": [
         { "id": "HAN" }
      ]
   }
}
```

```
{
   "human": {
      "id": "LEIA",
      "name": "Leia Organa",
      "friends": null
   }
}
```

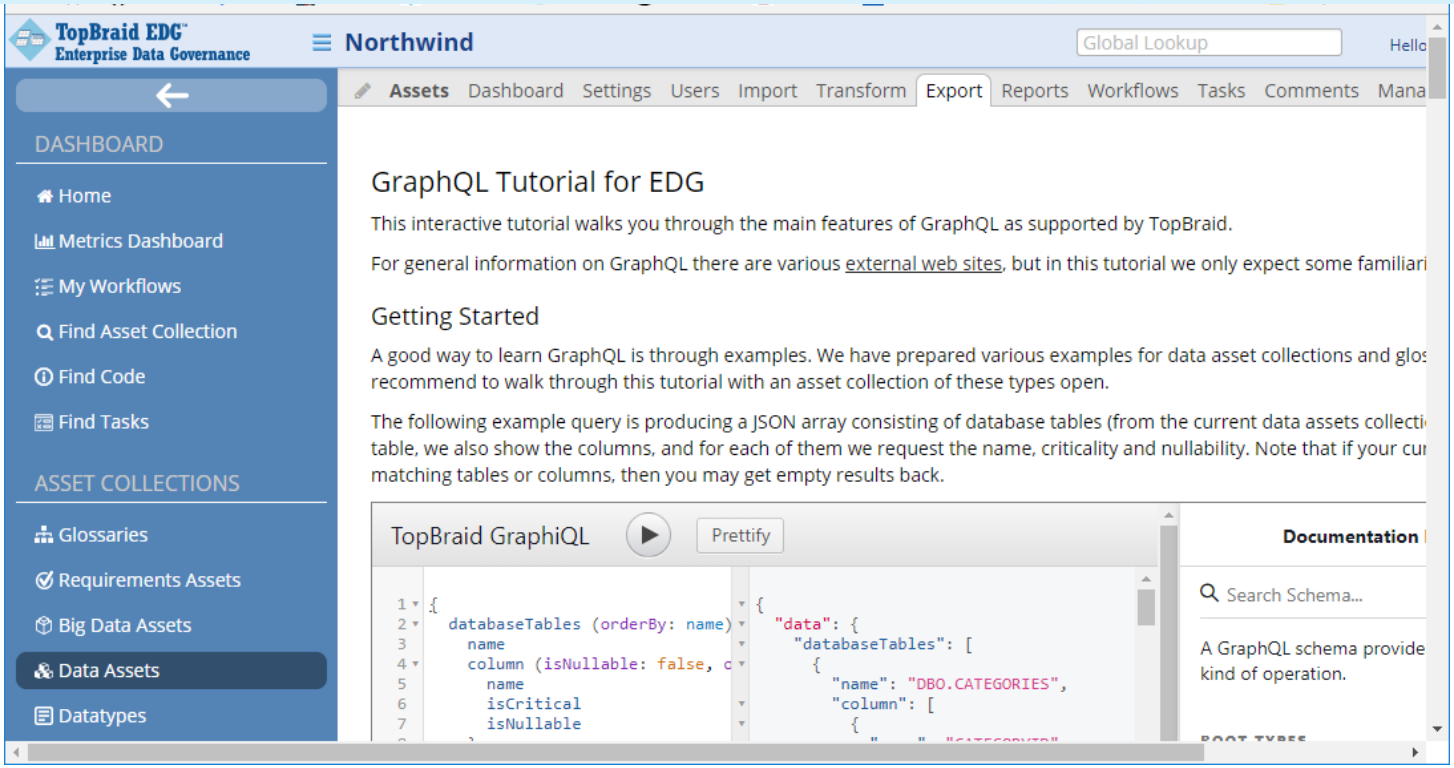
Example GraphQL Schema

```
schema {
   query: Query
}

type Query {
   human (id: ID!): [Human]
}

type Human @uri(template: "http://example.org/human/{$id}") {
   id: ID!
   name: String!
   friends: [Human]
}
```

URI from JSON Object	Field	Value
http://example.org/human/HAN	id	"HAN"
http://example.org/human/HAN	name	"Han Solo"
http://example.org/human/HAN	friends	http://example.org/human/LEIA
http://example.org/human/HAN	friends	http://example.org/human/LUKE
http://example.org/human/LEIA	id	"LEIA"
http://example.org/human/LEIA	name	"Leia Organa"
http://example.org/human/LUKE	id	"LUKE"
http://example.org/human/LUKE	name	"Luke Skywalker"
http://example.org/human/LUKE	friends	http://example.org/human/HAN

Next Steps

To try this out, download TBC-ME 6.0 or request a server evaluation account for TopBraid EDG at https://www.topquadrant.com/products/topbraid-enterprise-data-governance/request-edg-evaluation-account/
GraphQL tutorials and samples are built in. To access, go to an Export tab of any asset collection.



also read **https://www.topquadrant.com/technology/graphql/**