

## Configuration

# Génération d'une application GraphQL

## La mise en œuvre de l'application

Le projet doit avoir la structure **finale** suivante :

```

v WorkshopGraphQL ~/SOA 2526/WorkshopGraphQL
  > .idea
  v src
    v main
      v java
        v entities
          Module
          UniteEnseignement
        v graphql
          GraphQLEndPoint
          Mutations
          Query
        v repository
          ModuleBusiness
          UniteEnseignementBusiness
      v resources
        schema.graphql
    > test
  m pom.xml
```

1. Ajouter les dépendances suivantes au fichier POM :

<dependencies>

```

<dependency>
<groupId>com.graphql-java</groupId>
<artifactId>graphql-java</artifactId>
<version>3.0.0</version>
</dependency>
<dependency>
<groupId>com.graphql-java</groupId>
<artifactId>graphql-java-tools</artifactId>
<version>3.2.0</version>
</dependency>
<dependency>
<groupId>com.graphql-java</groupId>
<artifactId>graphql-java-servlet</artifactId>
<version>4.0.0</version>
</dependency>
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>javax.servlet-api</artifactId>
<version>3.0.1</version>
<scope>provided</scope> </dependency> </dependencies>

```

2. Créer la classe « **Query.java** » dans le package « **graphql** »

```

import com.coxautodev.graphql.tools.GraphQLRootResolver;

public class Query implements GraphQLRootResolver {

}

```

3. Créer le fichier « **schema.graphqls** » dans **src/main/ressources** :

```

type UniteEnseignement {
  code: Int!
  domaine: String!
}

```

```

    responsable: String!
    credits: Int!
    semestre: Int!
  }
  enum TypeModule {
    TRANSVERSAL
    PROFESSIONNEL
    RECHERCHE
  }
  type Module {
    matricule: String!
    nom: String!
    coefficient: Int!
    volumeHoraire: Int!
    type: TypeModule!
    uniteEnseignement: UniteEnseignement
  }
  type Query {
    getallmodules: [Module!]!
  }
  schema {
    query: Query
  }

```

4. Créer la classe « **GraphQLEndpoint.java** » dans le package « **graphql** ».

```

package graphql;
import com.coxautodev.graphql.tools.SchemaParser;
import graphql.schema.GraphQLSchema;
import graphql.servlet.SimpleGraphQLServlet;
import repository.ModuleBusiness;

import javax.servlet.annotation.WebServlet;

@WebServlet(urlPatterns = "/graphql")
public class GraphQLEndPoint extends SimpleGraphQLServlet {

    public GraphQLEndPoint() {
        super(buildSchema());
    }

    private static GraphQLSchema buildSchema() {

```

```

        return SchemaParser.newParser()
            .file("schema.graphqls")
            .resolvers(new Query())
            .build()
            .makeExecutableSchema();
    }
}

```

NB :

**@WebServlet** est utilisée pour définir un composant Servlet dans une application Web. Cette annotation est spécifiée sur une classe et contient des métadonnées sur le servlet déclaré.

**urlPatterns** est le paramètre permettant de définir le chemin d'accès à la ressource.

## 5. Déployer le projet et tester l'url via **Postman** :

<http://localhost:port/nomProjet/graphql>

The screenshot shows the Postman interface. At the top, the URL is set to `http://localhost:8083/WorkshopGraphQL_war_exploded/graphql`. The 'Query' tab is selected, and the query editor contains the following GraphQL query:

```

1 query Getallmodules {
2   getallmodules {
3     matricule
4     nom
5     coefficient
6     volumeHoraire
7     type
8   }
9 }
10

```

The 'Body' tab is selected, showing the JSON response:

```

1 {
2   "data": {
3     "getallmodules": [
4       {
5         "matricule": "M101",
6         "nom": "Algorithmique",
7         "coefficient": 3,
8         "volumeHoraire": 30,
9         "type": "PROFESSIONNEL"
10      }
11    ]
12  }
13 }

```

At the bottom right, the status is `200 OK`, the time is `13.66 ms`, and the size is `479 B`.