MICRONAUT MULTITENANCY

SERGIO DEL AMO
SERGIO DEL AMO

• MICRONAUT / GRAILS OCI TEAM
• GUADALAJARA, SPAIN
• CURATOR OF GROOVYCALAMARI.COM
• PODCAST HOST OF PODCAST.GROOVYCALAMARI.COM
• GREACH Conference organizer
• @SDELAMO
• HTTP://SERGIODELAMO.ES
Multi-Tenancy, as it relates to software development, is when a single instance of an application is used to service multiple clients (tenants) in a way that each tenants’ data is isolated from the other.
INSTALLATION
build.gradle
dependencies {
    ...
    ..
    ...'
    compile: 'io.micronaut:micronaut-multitenancy'
}
TENANT RESOLUTION
# Built-in Tenant Resolvers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cookie Tenant Resolver</td>
<td>Resolves the current tenant from an HTTP Cookie</td>
</tr>
<tr>
<td>HttpHeadersTenantResolver</td>
<td>Resolves the current tenant from the request HTTP Header</td>
</tr>
<tr>
<td>PrincipalTenantResolver</td>
<td>Resolves the current tenant from the authenticated username.</td>
</tr>
<tr>
<td>Subdomain Tenant Resolver</td>
<td>Resolves the tenant id from the sub domain</td>
</tr>
<tr>
<td>SessionTenantResolver</td>
<td>Resolves the current tenant from the HTTP Session.</td>
</tr>
</tbody>
</table>
SUBDOMAIN TENANT RESOLUTION
SUBDOMAIN TENANT RESOLUTION
# BUILT-IN TENANT RESOLVER

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemPropertyTenantResolver</td>
<td>Resolves the tenant id from a system property</td>
</tr>
<tr>
<td>FixedTenantResolver</td>
<td>Resolves against a fixed tenant id.</td>
</tr>
</tbody>
</table>
TENANT PROPAGATION
MULTI-TENANCY PROPAGATION

catalogue/src/main/resources/application.yml

micronaut:
  multitenancy:
    tenantresolver:
      httpheader:
        enabled: true
MULTI-TENANCY PROPAGATION

gateway/src/main/resources/application.yml

micronaut:
  multitenancy:
    propagation:
      enabled: true
      service-id-regex: 'catalogue'
    tenantresolver:
      subdomain:
        enabled: true
    tenantwriter:
      httpheader:
        enabled: true
GORM MULTITENANCY

GORM Multi-tenancy

- Configure Multitenancy Mode
- Configure Tenant Resolver
- Configure Domain classes which you want to be regarded as multi tenant.
- **User Tenants.* or Multi-tenancy Transformations.**
## Multi-tenancy modes

<table>
<thead>
<tr>
<th>Modes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE</td>
<td>Separate database with a separate connection pool is used to store each tenants data.</td>
</tr>
<tr>
<td>SCHEMA</td>
<td>The same database, but different schemas are used to store each tenants’ data.</td>
</tr>
<tr>
<td>DISCRIMINATOR</td>
<td>The same database is used with a discriminator used to partition and isolate data.</td>
</tr>
</tbody>
</table>
## Multi-tenancy modes - Data isolation

<table>
<thead>
<tr>
<th>Modes</th>
<th>Isolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATABASE</td>
<td>🔒🔒🔒</td>
</tr>
<tr>
<td>SCHEMA</td>
<td>🔒🔒</td>
</tr>
<tr>
<td>DISCRIMINATOR</td>
<td>🔒</td>
</tr>
</tbody>
</table>
MULTI-TENANCY PROPAGATION

src/main/resources/application.yml

grails:
gorm:
  multiTenancy:
    mode: DATABASE
    tenantResolverClass: 'io.micronaut.multitenancy.gorm.HttpHeaderTenantResolver'
@Entity
class Book implements MultiTenant<Book> {
    String title
}

@Entity
class Book implements MultiTenant<Book> {
    String title
    String tenantId
}
@Entity
class Book implements MultiTenant<Book> {
    String title
    String publisher
    static mapping = {
        tenantId name: 'publisher'
    }
}
# Multi-tenancy transformations

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@CurrentTenant</td>
<td>Resolve the current tenant for the context of a class or method</td>
</tr>
<tr>
<td>@Tenant</td>
<td>Use a specific tenant for the context of a class or method</td>
</tr>
<tr>
<td>@WithoutTenant</td>
<td>Execute logic without a specific tenant (using the default connection)</td>
</tr>
</tbody>
</table>
import grails.gorm.multitenancy.*

// resolve the current tenant for every method
@CurrentTenant
@ReadOnly
class TeamService {

  // execute the countPlayers method without a tenant id
  @WithoutTenant
  int countPlayers() {
    Player.count()
  }

  // use the tenant id "another" for all GORM logic within the method
  @Tenant("another")
  List<Team> allTwoTeams() {
    Team.list()
  }

  List<Team> listTeams() {
    Team.list(max:10)
  }

  @Transactional
  void addTeam(String name) {
    new Team(name:name).save(flush:true)
  }
}

© 2019, Object Computing, Inc. (OCI). All rights reserved.
GORM DEMO
LEARN MORE ABOUT OCI EVENTS & TRAINING

Events:

- objectcomputing.com/events

Training:

- objectcomputing.com/training
- grailstraining.com
- micronauttraining.com

Or email info@ocitraining.com to schedule a custom training program for your team online, on site, or in our state-of-the-art, Midwest training lab.