Exercise (Instructions): Loopback Relations

Objectives and Outcomes

In this exercise you will explore the use of model relations in Loopback and how we can link various models by defining relations among them. In additionn you will explore the use of a timestamp mixin. At the end of this exercise, you will be able to:

- Define model relations among various Loopback models
- Make use of a mixin within the Loopback server

Add a Comments Model

- Create a new model called Comments by typing the following at the prompt:

```
slc loopback:model
```

- Use the following options:

```
Model Name: Comments
Data Source: MongoDB
Model base: Persisted Model
Expose REST API: Yes
Model folder: common
Properties:
  Name: Rating
  Type: number
  Required: Yes
  Default: 5
  Name: comment
  Type: String
  Required: Yes
  Default: (empty)
```

Setting up Model Relations

- To define relationships, type the following at the command prompt:

```
slc loopback:relation
```

- First the relation between dishes and Comments, use the following options:

```
Model: dishes
Relation type: has many
Relationship with: Comments
Name: comments
Foreign key: none
Through model: no
```

- Now define a relation between dishes and customers, use the following options:

```
Model: dishes
Relation type: has many
Relationship with: Customer
Name: customers
Foreign key: none
Through model: no
```

- Between Comments and Dishes, use the following options:

```
Model: Comments
Relation type: belongs to
Relationship with: dishes
Name: dishes
Foreign key: none
```

- Between Comments and Customer, use the following options:

```
Model: Comments
Relation type: belongs to
Relationship with: Customer
Name: customer
Foreign key: customerId
```

- Between Customer and Comments, use the following options:

```
Model: Customer
Relation type: has many
Relationship with: Comment
Name: comments
Foreign key: customerId
Require through model: no
```

Define and Use a Mixin

- Install the loopback-ds-timestamp-mixin as follows:

```
npm install loopback-ds-timestamp-mixin --save
```

- Open model-config.json in the server folder, edit the mixins as follows:

```
"mixins": [
  "loopback/common/mixins",
  "loopback/server/mixins",
  "../node_modules/loopback-ds-timestamp-mixin",
  "../common/mixins",
  "./mixins"
]
```

- To use the mixin, add the following code to both comments.json and dishes.json in the common folder, after the properties:

```
"mixins": {
  "TimeStamp": true
},
```

Configuring Access Control

- You will now set access control for both dishes and Comments by typing the following at the prompt:

```
slc loopback:acl
```

- For the dishes model, use the following settings:

```
Model: dishes
Scope: All methods and properties
access type: Write
role: other
role name: admin
Permission: Explicitly grant access
```

- For the Comments model, use the following options:

```
Model: Comments
Scope: All methods and properties
access type: All
role: All users
Permission: Explicitly deny access
```

- Now to allow customers to read comments, use the following options:

```
Model: Comments
Scope: All methods and properties
access type: Read
role: Any authenticated user
Permission: Explicitly grant access
```

- To allow customers to post comments, use the following options:

```
Model: Comments
Scope: A single method
method name: create
role: Any authenticated user
Permission: Explicitly grant access
```

- To allow a customer that posted a comment to edit or delete the comment, use the following options:

```
Model: Comments
Scope: All methods and properties
access type: Write
role: The user owning the object
Permission: Explicitly grant access
```

- Start the server and explore the REST API using the API explorer.
- In particular if you get comments with the following filter: {"include":["dishes","customer"]}, the system will include the dish information and customer information into the comments.

Conclusions

In this exercise you explored the use of model relations in Loopback and used a timestamp mixin.

Mark as completed