バックエンドと結合 解説

RESTful APIサーバー設定

nuxt.config.jsの変更

`nuxt.config.js`の`axios{}`にbaseURLを設定する

設定前

```
  axios: {},
```

設定後

```
  axios: {
    baseURL: process.env.API_URL || 'http://localhost:2000'
  },
```

検索機能の実装

tableData初期値削除

`tableData`の`[ ~ ]`内データを削除し初期値を空にする

```
  data() {
    return {
      tableData: [],
      :
    }
  },
```

1件取得・全件取得の実装

`fetchKey(id) { .. }`の内容をRESTAPIコールするように書き換える
`fetchAll() { .. }`全件取得を実装する

変更前

```
    fetchKey(id) {
      // ToDo: REST APIのkey(row.id)検索し結果をitemにセットする
      // Dummy select
      const items = this.tableData.filter((data) => data.id === id)
      this.item = { ...items[0] }
    },
```

変更後

```
    async fetchAll() {
      await this.$axios.$get('/api/items/').then((res) => {
        this.tableData = [...res]
      })
    },
    async fetchKey(id) {
      await this.$axios.$get(`/api/items/${id}`).then((res) => {
        this.form = { ...res }
        this.tableData[this.rowNumber] = { ...res }
      })
    },
```

解説

> `async ~ await:` 通常は非同期処理となるが、async～awaitを使うことで、時間がかかるAPIコールの結果を待って処理するようになる

> `this.$axios.$get(path)`: baseURLに設定したサーバーに対してpathをGETする
> `.then((res) => { .. }`: 実行結果を待ち、その結果を{}内で処理する

初期表示で全件取得し結果を表示する

初期表示の際に全件取得し、結果を一覧表示するよう実装する

`conputed: { .. }` と `methods: { .. }` の間に `mounted() { .. }` を追加し、画面のレンダリングが完了したら一覧表示するように変更する。

変更前

```
  computed: {
    dialogTitle() {
      return this.isUpdate ? 'ToDo 編集' : 'ToDo 新規登録'
    }
  },
  methods: {
```

変更後

```
  computed: {
    dialogTitle() {
      return this.isUpdate ? 'ToDo 編集' : 'ToDo 新規登録'
    }
  },
  mounted() {
    this.fetchAll()
  },
  methods: {
```

動作確認

まず、REST APIをポート2000番で起動しておくこと。

```
cd [RESTAPIディレクトリ]/build/libs
java -jar webapi-0.0.1-SNAPSHOT.jar
```

その後
`yarn dev` し起動した後、`http://localhost:3000/items/` にアクセスする。

一覧表示でDBの内容が表示されます。

更新機能の実装

更新(`updateItem`)の修正

updateItemをRESTAPIをコールし、結果により表示を変えるよう修正します。

変更前

```
    updateItem(param) {
      // ToDo: REST APIのアップデートを起動する
      // Dummy update
      this.tableData[this.rowNumber] = { ...param }

      // 更新後の情報を取得しtableDataにセットする
      this.fetchKey(param.id)
      this.tableData[this.rowNumber] = { ...this.item }

      const target = `${param.id}:${param.title}`
      this.$message({
        type: 'success',
        message: `${target} : 更新が成功しました。`,
        showClose: true,
        duration: 5000
      })
    },
```

変更後

```
    updateItem(param) {
      const target = `${param.id}:${param.title}`
      this.$axios({
        url: `/api/items/${param.id}`,
        method: 'PUT',
        data: param
      })
        .then((res) => {
          this.fetchKey(param.id).then((response) => {
            this.$message({
              type: 'success',
              message: `${target} : 更新が成功しました。`,
              showClose: true,
              duration: 5000
            })
          })
        })
        .catch((err) => {
          this.$message({
            type: 'danger',
            message: `${target} : 更新に失敗しました。: ${err}`,
            showClose: true,
            duration: 0
          })
        })
    },
```

解説

> `this.$axios({..})`: PUTやPOSTでリクエストボディにデータセットが必要な場合は{}内にパラメータを記載する
> `.catch((err) => {..})`: HTTPレスポンスで200番以外が返却された場合、このブロックが実行される

更新をコールする部分(`handleEdit`)の修正

不具合の修正とformへのデータセットをfetchKey内で行うように変更する

変更前

```
    handleEdit(index, row) {
      this.isUpdate = true
      this.rowNumber = index
      this.fetchKey(row.id)
      this.form = { ...this.item }
      this.dialogFormVisible = true
    },
```

変更後

```
    handleEdit(index, row) {
      this.isUpdate = true
      this.rowNumber = this.tableData.indexOf(row)
      this.fetchKey(row.id).then((res) => {
        this.dialogFormVisible = true
      })
    },
```

動作確認

一覧から編集ボタンをクリックし、内容変更後確定ボタンクリックすることで変更内容が反映されます。

削除機能の実装

削除(`deleteItem`)の修正

`deleteItem`をRESTAPIをコールし、結果により表示を変えるよう修正します。
削除成功メッセージの位置も削除結果により表示するように変更します。

変更前

```
    deleteItem(id) {
      // ToDo: REST API削除処理呼び出し
```

```
    // Dummy delete
    this.tableData = this.tableData.filter((data) => data.id !== id)
  },
```

変更後

```
  deleteItem(id) {
    const target = `${id}`
    this.$axios
      .$delete(`/api/items/${id}`)
      .then((res) => {
        this.tableData = this.tableData.filter((data) => data.id !== id)
        this.$message({
          type: 'success',
          message: `${target} : 削除が成功しました。`,
          showClose: true,
          duration: 5000
        })
      })
      .catch((err) => {
        this.$message({
          type: 'danger',
          message: `${target} : 削除に失敗しました。: ${err}`,
          showClose: true,
          duration: 0
        })
      })
  },
```

メッセージ位置変更

変更前

```
  confirmDelete() {
    const row = this.tableData[this.rowNumber]
    const target = `${row.id}:${row.title}`
    const msg = `${target} を削除します。削除後は元に戻せませんが、実行してよろしいですか?`
    this.$confirm(msg, 'Warning', {
      confirmButtonText: '削除する',
      cancelButtonText: 'やめる',
      type: 'warning'
    })
      .then(() => {
        this.deleteItem(row.id)
        this.$message({
          type: 'success',
          message: `${target} : 削除が成功しました。`,
          showClose: true,
          duration: 5000
```

```
        })
      })
      .catch(() => {
        this.$message({
          type: 'info',
          message: '削除を中止しました。'
        })
      })
    }
```

変更後

```
    confirmDelete() {
      const row = this.tableData[this.rowNumber]
      const target = `${row.id}:${row.title}`
      const msg = `${target} を削除します。削除後は元に戻せませんが、実行してよろしいですか?`
      this.$confirm(msg, 'Warning', {
        confirmButtonText: '削除する',
        cancelButtonText: 'やめる',
        type: 'warning'
      })
        .then(() => {
          this.deleteItem(row.id)
        })
        .catch(() => {
          this.$message({
            type: 'info',
            message: '削除を中止しました。'
          })
        })
    }
```

削除をコールする部分(`handleDelete`)の修正

不具合の修正

変更前

```
    handleDelete(index, row) {
      this.rowNumber = index
      this.confirmDelete()
    },
```

変更後

```
    handleDelete(index, row) {
      this.rowNumber = this.tableData.indexOf(row)
```

```
      this.confirmDelete()
    },
```

動作確認

一覧から削除ボタンをクリックし、削除確定ボタンクリックすることで削除内容が反映されます。

新規登録機能の実装

新規登録(`registerItem`)の修正

`registerItem`をRESTAPIをコールし、結果により表示を変えるよう修正します。

変更前

```
    registerItem(param) {
      // ToDo: REST API登録処理呼び出し
      // Dummiy insert
      const item = { ...param }
      this.tableData.push(item)
      this.nextId++

      const target = `${param.id}:${param.title}`
      this.$message({
        type: 'success',
        message: `${target} : 登録が成功しました。`,
        showClose: true,
        duration: 5000
      })
    },
```

変更後

```
    registerItem(param) {
      this.tableData.push(param)
      this.rowNumber = this.tableData.length - 1
      this.$axios({
        url: `/api/items/`,
        method: 'POST',
        data: param
      })
        .then((res) => {
          this.fetchKey(res.data).then((response) => {
            const target = `${res.data}:${param.title}`
            this.$message({
              type: 'success',
              message: `${target} : 登録が成功しました。`,
              showClose: true,
```

```
            duration: 5000
          })
        })
      })
      .catch((err) => {
        const target = `${param.title}`
        this.$message({
          type: 'danger',
          message: `${target} : 登録に失敗しました。: ${err}`,
          showClose: true,
          duration: 0
        })
      })
  },
```

解説

> `this.tableData.push(param)`: tableDataに1行追加
> `this.rowNumber = this.tableData.length-1`: fetchKeyの結果を移送するIndexをセット
> `this.fetchKey(res.data)`: 登録の返却値(id) `res.data` を使い再度データを検索し先ほど追加したレコードに移送する

新規登録をコールする部分(`handleNew`)の修正

ダミー処理の削除

変更前

```
  handleNew() {
    this.isUpdate = false
    this.form.id = null
    this.form.title = ''
    this.form.content = ''
    this.form.status = 'TODO'
    this.dialogFormVisible = true
    // 以下はREST APIと接続するまでのダミー処理
    // idはREST APIでは自動採番の予定
    this.form.id = this.nextId
  },
```

変更後

```
  handleNew() {
    this.isUpdate = false
    this.form.id = null
    this.form.title = ''
    this.form.content = ''
    this.form.status = 'TODO'
```

```
        this.dialogFormVisible = true
      },
```

動作確認

新規登録ボタンをクリックし、登録ダイアログを開き、新規データを入力しDBに登録されることを確認する。

リロードボタン追加と不要な変数削除

新規登録ボタンの右にリロードボタンを設置

変更前

```
        <!-- 検索入力 -->
        <el-row>
          <el-col :span="20">
            <el-input v-model="search" size="small" placeholder="検索文字列を入力">
              <template slot="prepend">内容で絞り込む</template>
            </el-input>
          </el-col>
          <el-col :span="4" style="text-align: center">
            <el-button type="success" icon="el-icon-circle-plus-outline"
size="small" @click="handleNew()">新規登録</el-button>
          </el-col>
        </el-row>
        <!-- 表部分 -->
```

変更後

```
        <!-- 検索入力 -->
        <el-row>
          <el-col :span="18">
            <el-input v-model="search" size="small" placeholder="検索文字列を入力">
              <template slot="prepend">内容で絞り込む</template>
            </el-input>
          </el-col>
          <el-col :span="6" style="text-align: right">
            <el-button type="success" icon="el-icon-circle-plus-outline"
size="small" @click="handleNew()">新規登録</el-button>
            <el-tooltip class="item" content="再読み込み" placement="top">
              <el-button type="info" icon="el-icon-refresh-right" size="small"
circle @click="fetchAll()"></el-button>
            </el-tooltip>
          </el-col>
        </el-row>
        <!-- 表部分 -->
```

変数削除

`data() { ~ }` 内にある `item: { }` と `nextId: 8` は不要なので削除する。