Using Arrow Functions in Controller Methods

When using controller methods as callback functions in Express routes, prefer **arrow functions** to preserve the context of `this`.

```
class UserController { getUser = (req, res) => {
// 'this' correctly refers to the controller instance
}; }
```

Alternatively, you can use `bind()` in the constructor to explicitly bind methods:

```
class UserController {
constructor() {
this.getUser = this.getUser.bind(this);
} getUser(req, res) { // 'this' is correctly bound } }
```

Both methods prevent `undefined` when `this` is used inside callbacks.

Dependency Injection in Controllers

Inject dependencies through instance properties (this.service`) instead of shared references or static imports.

Benefits:

- Easier unit testing
- Clear separation of concerns
- Supports SOLID principles
- Better flexibility and scalability

Error Handling with Try-Catch

Wrap service calls inside `try-catch` blocks to properly handle errors and avoid crashes.

```
async getUser(req, res) {
try {
const user = await this.userService.findById(req.params.id);
```

```
res.json(user);
} catch (error) {
res.status(500).json({ message: error.message });
} }
```

Model and Repository Pattern

Define a simple model and a separate repository layer.

Advantages:

- Clear separation of responsibilities
- Better code organization
- Easier to scale and maintain

Using layout for ejs template

```
npm install express-ejs-layout
```

in the index.js

```
const expressLayout = require("express-ejs-layouts");
```

```
app.set("view engine", "ejs");
app.set("views", "views");
```

```
app.use(expressLayout);
app.set("layout", "layouts/base") //define default layout;
app.set("layout extractScripts", true) // extract vues script;
app.set("layout extractStyles", true) // extract style script;
```

Using methodOverride for methods not supported by HTML

```
npm install method-override
```

In the template, use a POST form with a hidden input field for method-override to transform it into a DELETE request.

```
<form action="/tasks/delete/<%= task.id%>" method="POST">
<input type="hidden" name="_method" value="DELETE" />
<input type="submit" value="Delete" onclick="return confirm('Êtes-vous sûr de vouloir supprimer')/> `
```

then, in the server/index.js

```
const methodOverride = require("method-override");

app.use(methodOverride("_method"));
```

finally, u can just let in your Router the delete method

```
router.delete("/delete/:id", controller.delete);
```
