Урок N5

Создание модулей и работа с MySQL

У нашего приложения уже есть controller и view, но еще нет model. Чтобы оно было настоящим MVC-приложением, в этом уроке мы создадим модель для работы с данными и разместим ее в отдельном подключаемом модулей. Также наша модель будет работать с БД MySQL с помощью модуля **mysql**.

Содержание урока

- Модульная структура приложения
- Работа с БД через модуль **mysql**
- Пример: TODO-лист с HTML-интерфейсом

Создание модулей

Чтобы наша модель была универсальным компонентом, сделаем ее в виде отдельного файла, который можно будет легко подключить через уже знакомую нам функцию require.

models/tasks.js:

```
var Tasks = {
	list: function(callback) {
		// TODO
	},

	add: function(task, callback) {
		// TODO
	},

	change: function(id, text, callback) {
		// TODO
	},

	complete: function(id, callback) {
		// TODO
	},

	delete: function(id, callback) {
		// TODO
	}
};

module.exports = Tasks;
```

Через объект module.exports мы показываем, какие переменные из нашего модуля будут доступны снаружи. В данном случае это объект Tasks, который содержит функции для работы с моделью. При подключении модуля функция require вернет нам содержимое этого объекта и таким образом мы получим доступ к нужному функционалу:

```
var tasks = require('./models/tasks');
```

Модуль mysql

Устанавливается стандартно через npm под названием mysql.

Инициализация подключения:

```
var mysql = require('mysql');

var connection = mysql.createConnection({
	host		: 'localhost',
	database	: 'todo',
	user		: 'root',
	password	: 'mysql'
});

connection.connect(function(err) {
	if (err)
		console.error(err);
});
```

Для постоянно работающих программ соединения с базой лучше создавать с помощью connectionPool. Пул соединений будет автоматически следить за статусом созданных соединений и создавать новые при необходимости. Для создания пула вместо mysql.createConnection используйте mysql.createPool с теми же параметрами подключения к БД. Для получения действующего соединения используется метод getConnection:

```
pool.getConnection(function(err, connection) {
	// используем полученное соединение
	connection.query( 'SELECT * FROM table', function(err, rows) {
		// возвращаем соединение в пул
		connection.release();
	});
});
```

Подготовка SQL запросов

Для экранирования спецсимволов используйте escape:

```
var userId = 'some user provided value';
var sql    = 'SELECT * FROM users WHERE id = ';
sql = sql + mysql.escape(userId);
```

Для экранирования идентификаторов используйте escapeId:

```
var sorter = 'date';
var query = 'SELECT * FROM posts ORDER BY ';
query = query + mysql.escapeId(sorter);
console.log(query); // SELECT * FROM posts ORDER BY `date`
```

Чтобы подготовить запрос с автоматическим экранированием спецсимволов и удобной подстановкой значений, используйте format. Обратите внимание, что идентификаторы должны обозначаться как ??, а непосредственно значения как ?:

```
var sql = "SELECT * FROM ?? WHERE ?? = ?";
var inserts = ['users', 'id', 1];
sql = mysql.format(sql, inserts);
console.log(sql);
// SELECT * FROM `users` WHERE `id` = 1
```

Выполнение SQL запросов

Для выполнения произвольного запроса используйте query:

connection.query(sql, function(err, results) { ... });

При успешном запросе объект results будет содержать поле affectedRows, которое покажет количество строк, к которым был применен запрос. В changedRows будет указано количество строк, изменивших свое содержание (актуально для UPDATE). В insertId будет содержаться идентификатор последней вставленной строки (актуально для INSERT).

Для автоматического экранирования значений и их удобной подстановки в запрос используйте placeholders и дополнительный аргумент с массивом значений при вызове query:

```
connection.query(
        'SELECT * FROM users WHERE id = ? AND password = ?',
        [userId, userPassword],
        function(err, results) { ... }
);
```

Если вместо массива использовать объект, он будет преобразован в пары ключ-значений, что удобно использовать совместно с INSERT:

```
var post  = {id: 1, title: 'Hello MySQL'};
var query = connection.query(
        'INSERT INTO posts SET ?',
        post,
        function(err, result) { ... }
);
console.log(query.sql);
// INSERT INTO posts SET `id` = 1, `title` = 'Hello MySQL'
```

Самоконтроль

✓Создание собственный модулей
✓Использование пула соединений mysql
✓Подготовка и выполнение безопасных SQL запросов

Домашнее задание

Вариант А, доработка примера из урока:

Параметры подключения к БД вынести в модуль config.js
Использовать для работы с БД connectionPool
Реализовать методы редактирования и удаления записей.
Добавить опцию для указания приоритета задачи.

Вариант В, своя модель CRUD (create-read-update-delete):

Реализовать пункты задания из варианта А применительно к любой таблице в БД (например, для статей в блоге и т.п.). В дальнейшем вы сможете использовать этот модуль в качестве компонента для работы с данными в своем приложении.