

# Order ETag Demo (Node.js + Express + MongoDB)

---

*Base caching + concurrency demo using HTTP ETag and conditional requests*

## Index

1. Project Overview
  2. Learning Outcomes
  3. Tech Stack
  4. Folder Structure
  5. High-Level Architecture
  6. ETag Concepts Used in This App
  7. API Endpoints
  8. How to Run + Quick Tests
  9. Common Mistakes
  10. Debugging Techniques
- Appendix A: Full Source Code

## 1. Project Overview

This application demonstrates HTTP ETag and conditional requests to reduce unnecessary downloads and to avoid lost updates when multiple clients update the same resource.

## 2. Learning Outcomes

- Explain ETag and conditional headers (If-None-Match, If-Match).
- Return 304 Not Modified when a cached client copy is still fresh.
- Reject stale updates using If-Match (412 Precondition Failed).
- Debug common ETag and caching issues.

## 3. Tech Stack

package.json:

```
{
  "name": "order-etag-demo",
  "version": "1.0.0",
  "type": "commonjs",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "^4.19.2",
    "mongoose": "^8.5.2"
  }
}
```

## 4. Folder Structure

```
- package.json
- server.js
- routes/orders.js
- models/Order.js
```

## 5. High-Level Architecture

Read flow: Client → GET /orders/:id → server computes ETag → returns ETag + body

Cache validation: Client sends If-None-Match → server returns 304 if unchanged

Safe write flow: Client sends If-Match on PUT/PATCH → server applies only if matches current version; else 412

## 6. ETag Concepts Used in This App

ETag is a server-generated identifier for a specific version of a resource. The server returns it on GET. Clients can revalidate with If-None-Match, and can protect updates with If-Match.

### Headers:

- ETag (server response)
- If-None-Match (GET revalidation)
- If-Match (update precondition)

### Implementation snippets:

#### package.json

```
{
  "name": "order-etag-demo",
  "version": "1.0.0",
  "type": "commonjs",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "^4.19.2",
    "mongoose": "^8.5.2"
  }
}
```

#### server.js

```
app.use("/orders", orderRoutes);

app.listen(3000, () =>
  console.log("Order ETag Demo running on http://localhost:3000")
);
```

#### routes/orders.js

```
const order = await Order.findById(req.params.id);
if (!order) return res.status(404).json({ error: "Order not found" });

const etag = generateETag(order);
res.set("ETag", etag);

const clientETag = req.headers["if-none-match"];
```

## 7. API Endpoints

Base URL: <http://localhost:3000> (or as configured)

| Method | Path        | Source File      |
|--------|-------------|------------------|
| POST   | /orders/    | routes/orders.js |
| GET    | /orders/:id | routes/orders.js |

|     |             |                  |
|-----|-------------|------------------|
| PUT | /orders/:id | routes/orders.js |
|-----|-------------|------------------|

## 8. How to Run + Quick Tests

### Start MongoDB:

```
docker compose up -d
```

### Run server:

```
npm install
npm run dev
```

### ETag GET validation:

```
curl -i http://localhost:3000/orders/<ORDER_ID>

curl -i http://localhost:3000/orders/<ORDER_ID> \
  -H 'If-None-Match: <ETAG_FROM_PREVIOUS_RESPONSE>'
```

### Safe update with If-Match:

```
curl -i -X PATCH http://localhost:3000/orders/<ORDER_ID> \
  -H 'Content-Type: application/json' \
  -H 'If-Match: <ETAG_FROM_LATEST_GET>' \
  -d '{"status": "CONFIRMED"}'
```

## 9. Common Mistakes

### ETag missing on GET

Client cannot revalidate, so every GET downloads the full payload.

### ETag changes even when data is unchanged

Usually caused by hashing unstable fields or non-deterministic serialization.

### Not handling 304 correctly

304 must not include a response body; the client should use cached content.

### Ignoring If-Match on updates

Leads to lost updates when concurrent clients modify the same order.

### Client uses stale If-Match

Expect 412; client must refetch latest state and retry.

## 10. Debugging Techniques

- Use curl -i to inspect headers and confirm ETag is present.
- Call the same GET twice and verify ETag stability if the resource does not change.
- When you get 304, confirm the server returns no body and your client reads cached data.
- When you get 412, compare If-Match with the latest GET ETag; refetch and retry.

## Appendix B: Full Source Code

### package.json

```
{
  "name": "order-etag-demo",
  "version": "1.0.0",
  "type": "commonjs",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "express": "^4.19.2",
    "mongoose": "^8.5.2"
  }
}
```

### server.js

```
const express = require("express");
const mongoose = require("mongoose");
const orderRoutes = require("./routes/orders");

const app = express();
app.use(express.json());

mongoose.connect("mongodb://localhost:27017/order_etag_demo")
  .then(() => console.log("MongoDB connected"))
  .catch(err => console.error(err));

app.use("/orders", orderRoutes);

app.listen(3000, () =>
  console.log("Order ETag Demo running on http://localhost:3000")
);
```

### models/Order.js

```
const mongoose = require("mongoose");

const OrderSchema = new mongoose.Schema(
  {
    product: String,
    amount: Number,
    status: String
  },
  { timestamps: true }
);

module.exports = mongoose.model("Order", OrderSchema);
```

## routes/orders.js

```
const express = require("express");
const Order = require("../models/Order");

const router = express.Router();

function generateETag(order) {
  return `order-${order._id}-${order.updatedAt.getTime()}`;
}

router.post("/", async (req, res) => {
  const order = await Order.create({
    product: req.body.product,
    amount: req.body.amount,
    status: "CREATED"
  });
  res.status(201).json(order);
});

router.get("/:id", async (req, res) => {
  const order = await Order.findById(req.params.id);
  if (!order) return res.status(404).json({ error: "Order not found" });

  const etag = generateETag(order);
  res.set("ETag", etag);

  const clientETag = req.headers["if-none-match"];
  if (clientETag === etag) {
    return res.status(304).end();
  }

  res.json(order);
});

router.put("/:id", async (req, res) => {
  const order = await Order.findByIdAndUpdate(
    req.params.id,
    req.body,
    { new: true }
  );

  if (!order) return res.status(404).json({ error: "Order not found" });

  res.json(order);
});

module.exports = router;
```