CMPE282

Akshay Mishra 011476673

**Q1**.

* 1. List technologies, software (including version), and platforms for dev tools, REST client, REST server, and NoSQL database.
  2. How did you build your REST server?
  3. Is the format of the data object based on XML or JSON?
  4. A sample HTTP request URI and request body for POST to create a new employee based on the XML format or JSON format (depending on your answer to c). Also indicate if there is any additional setup (e.g., HTTP header, etc.).
  5. If the NoSQL needs to create DB objects beforehand, such as Cassandra, include the script to create DB objects.

**Answers:**

**Dev tools:**

Eclipse Java EE IDE for Web Developers.

Version: Neon.3 Release (4.6.3)

on

MacOS Sierra 10.12.6

**REST Client:**

curl 7.54.0 (x86\_64-apple-darwin16.0) libcurl/7.54.0 SecureTransport zlib/1.2.8

on

MacOS Sierra 10.12.6

**REST Server:**

Spring Boot v1.2.6, Tomcat 8.5

on

MacOS Sierra 10.12.6

**NoSQL database:**

MongoDB shell version v3.4.9

on

MacOS Sierra 10.12.6

b.

**REST Server:** Built a Spring Boot Application and then deploy to Tomcat 8.5 for development, Packaged as a jar accessible APIs on Tomcat 8.5 at port 8080

c. Based on **JSON** only.

d.

Request URI: POST /cmpe282Akshay673/rest/employee

Request body:

{

"id": 1,

"fname": "Abc",

"lname": "Def"

}

No additional setup

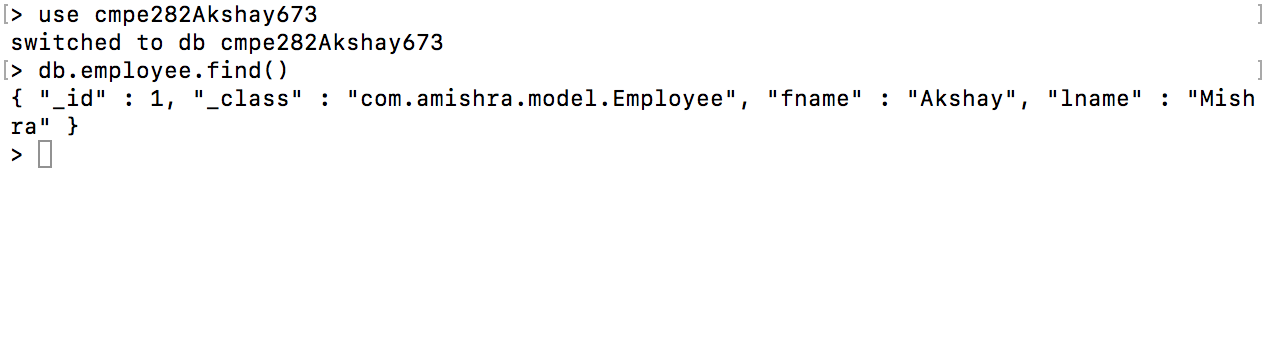
* 1. **MongoDB script:** No script needed

**Q2: Functionality Matrix**

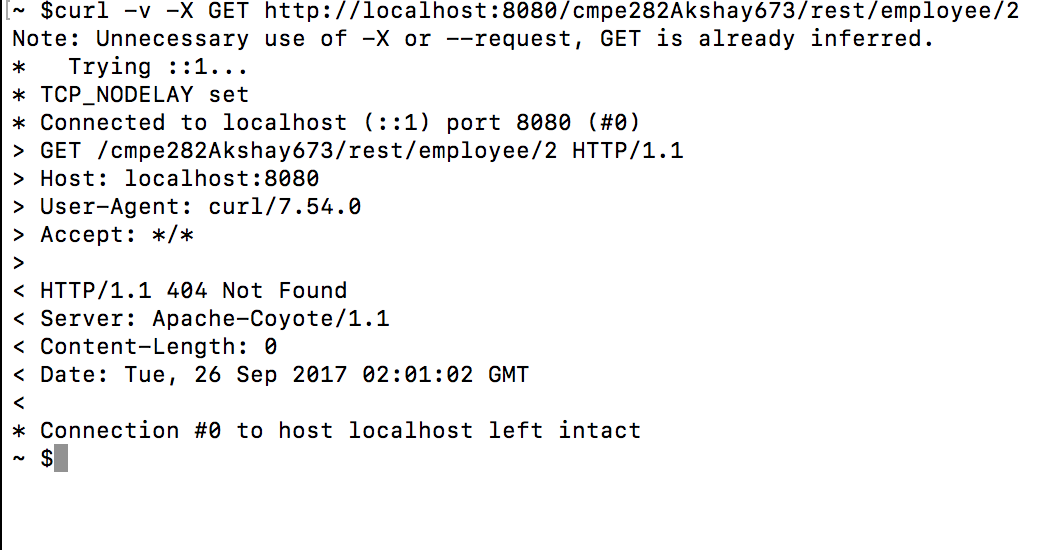
|  |  |
| --- | --- |
| **HTTP Method + URI** | **Status** |
| GET /.../rest/employee/m | **done** |
| GET /.../rest/project/n | **done** |
| POST /.../rest/employee | **done** |
| POST /.../rest/project | **done** |
| PUT /.../rest/employee/m | **done** |
| PUT /.../rest/project/n | **done** |
| DELETE /.../rest/employee/m | **done** |
| DELETE /.../rest/project/n | **done** |
| GET /.../rest/employee | **done** |
| GET /.../rest/project | **done** |

1. GET /.../rest/employee/m

**Before:**

****

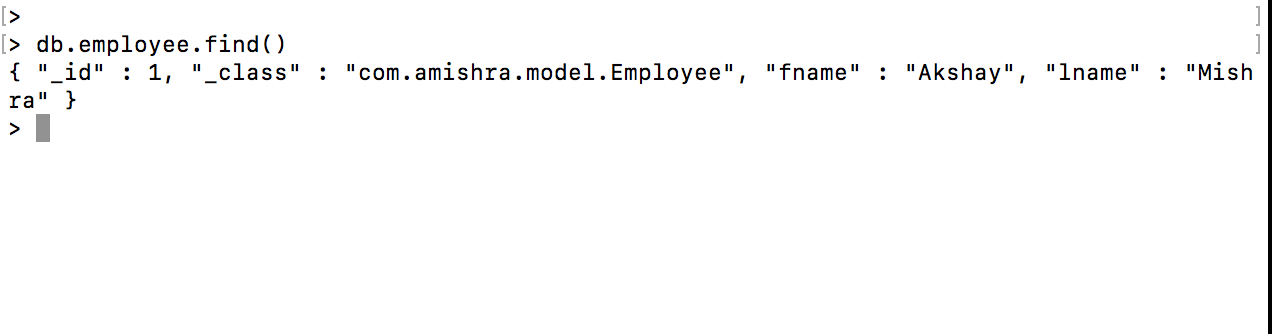
**Error:**

****

**Success:**

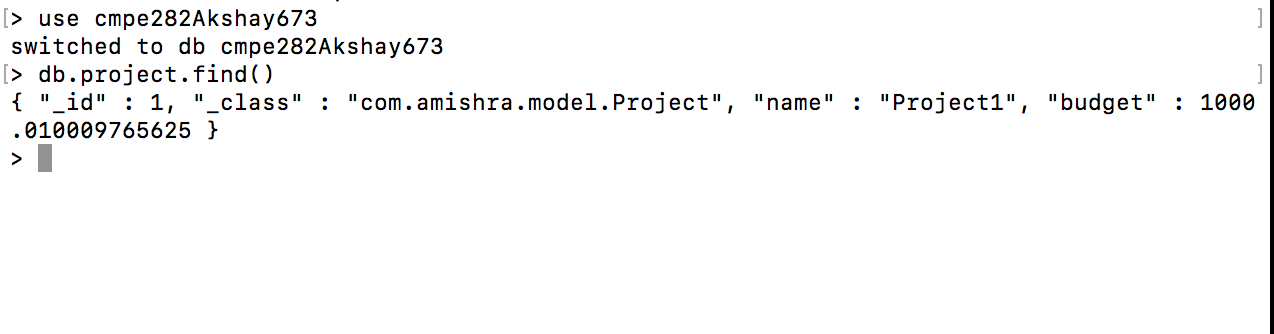
****

**After:**

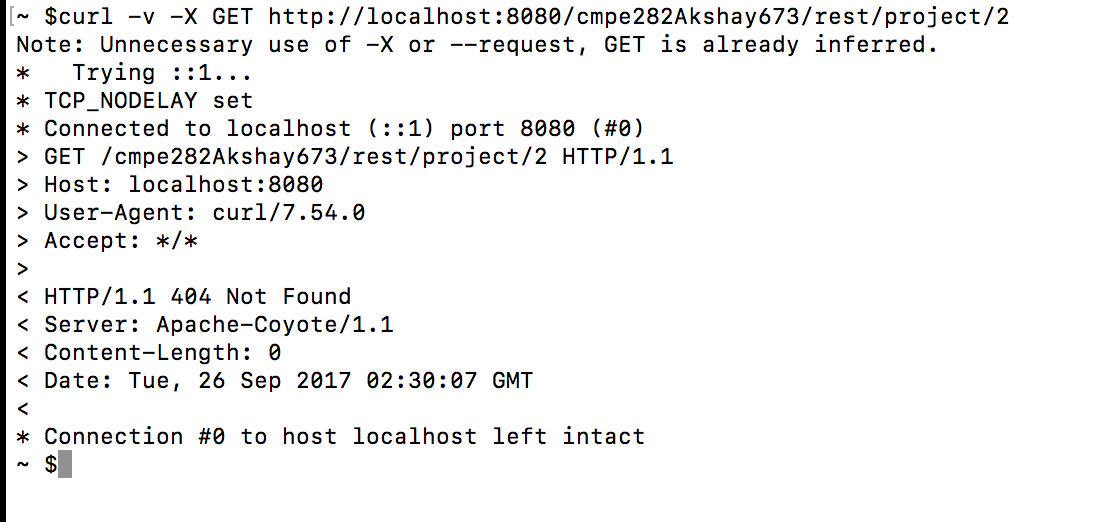
****

1. GET /.../rest/project/n

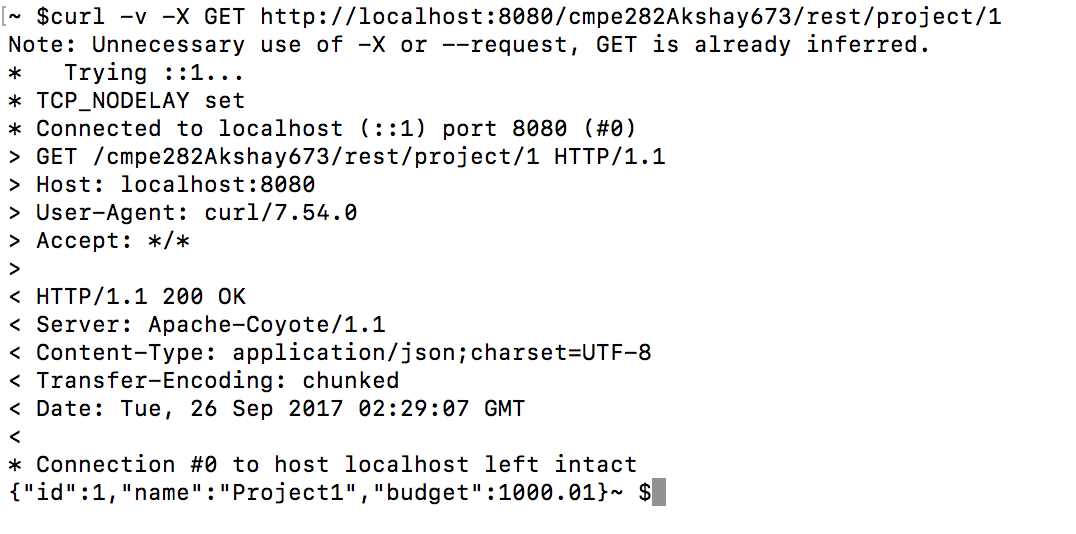
**Before:**

****

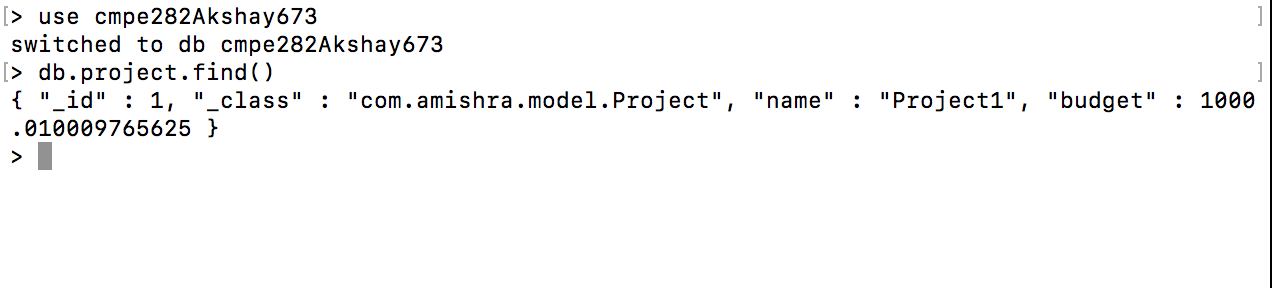
**Error:**

****

**Success:**

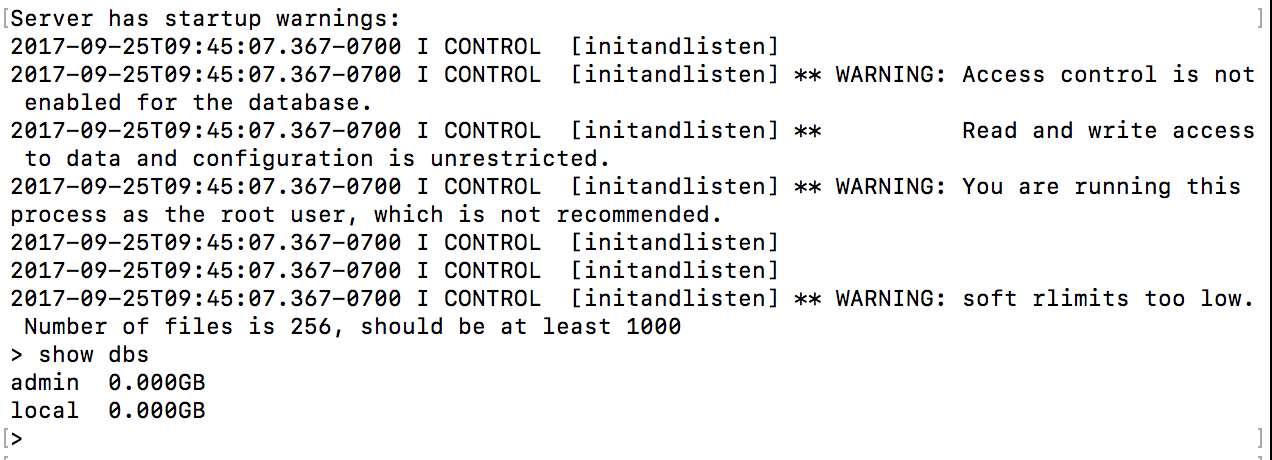
****

**After:**

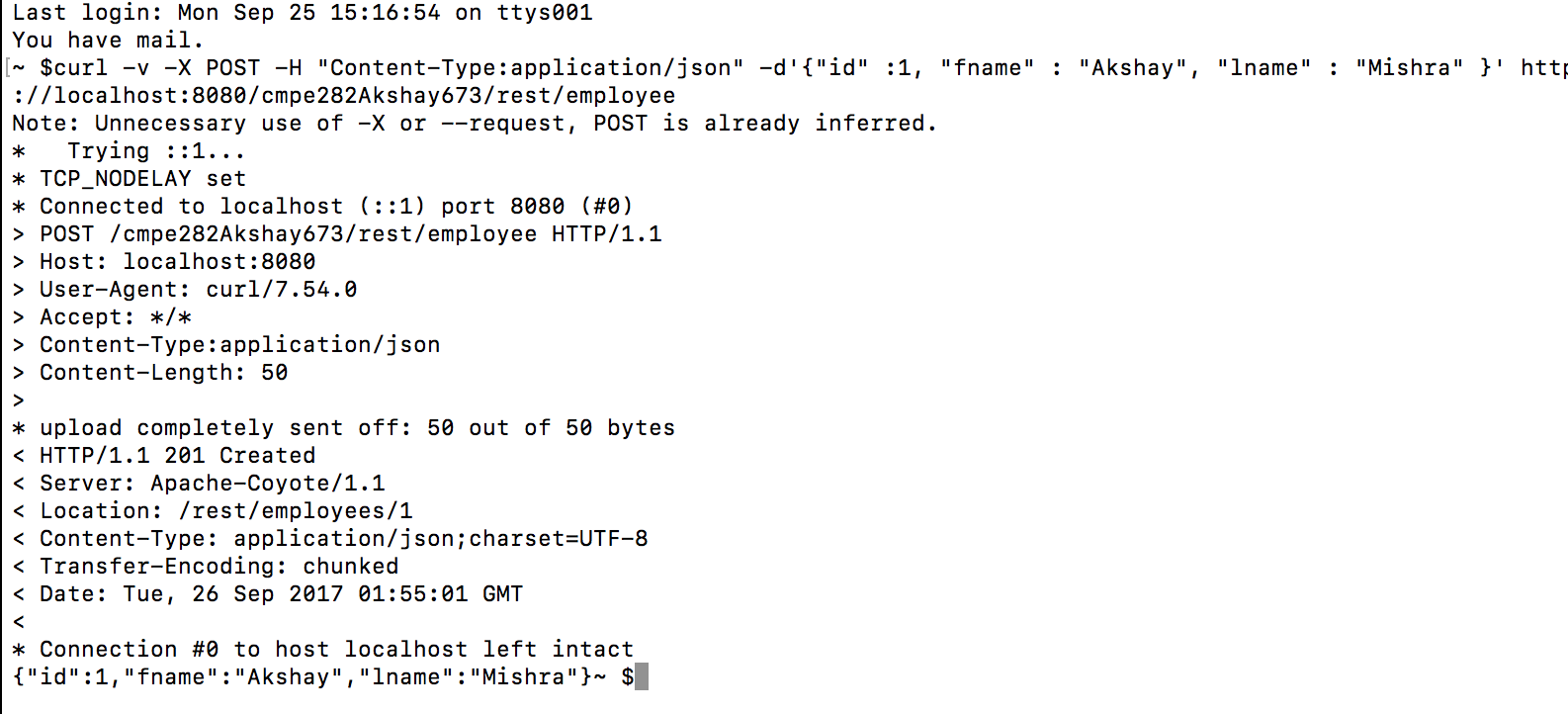
****

1. POST /.../rest/employee

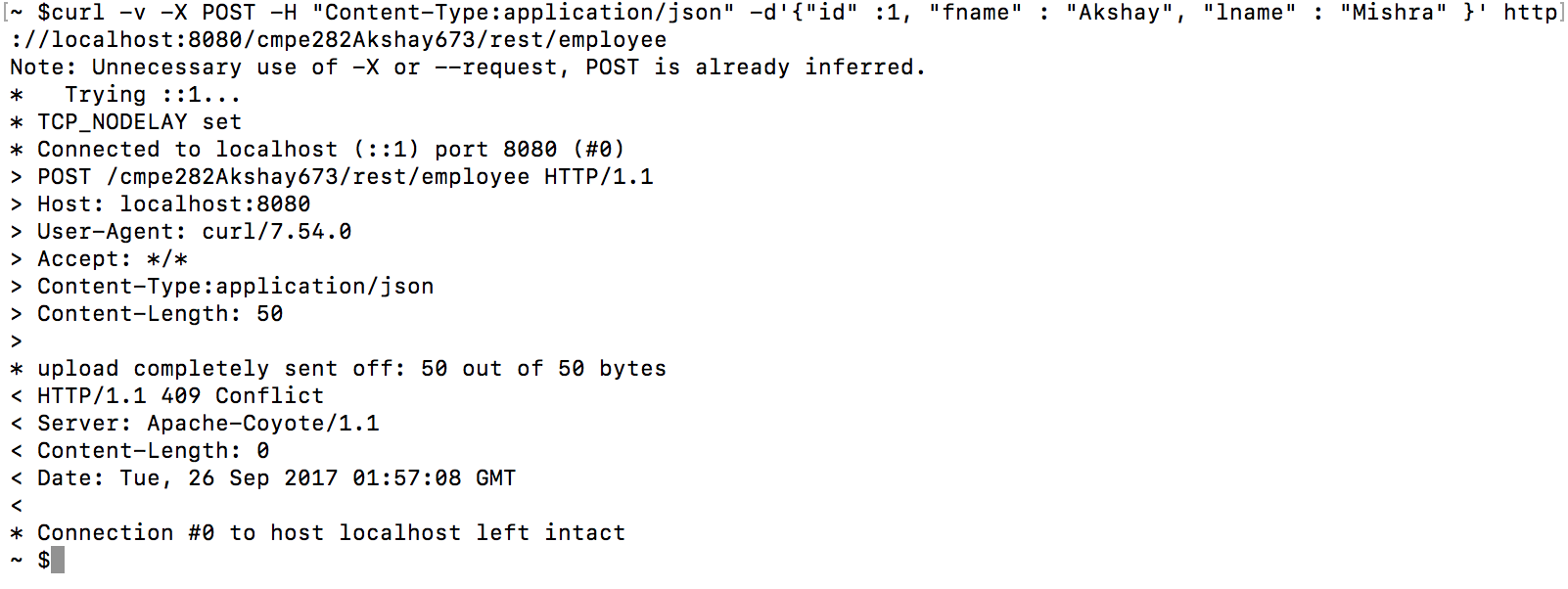
**Before:**

****

**Success:**

****

**Error:**

****

**After:**

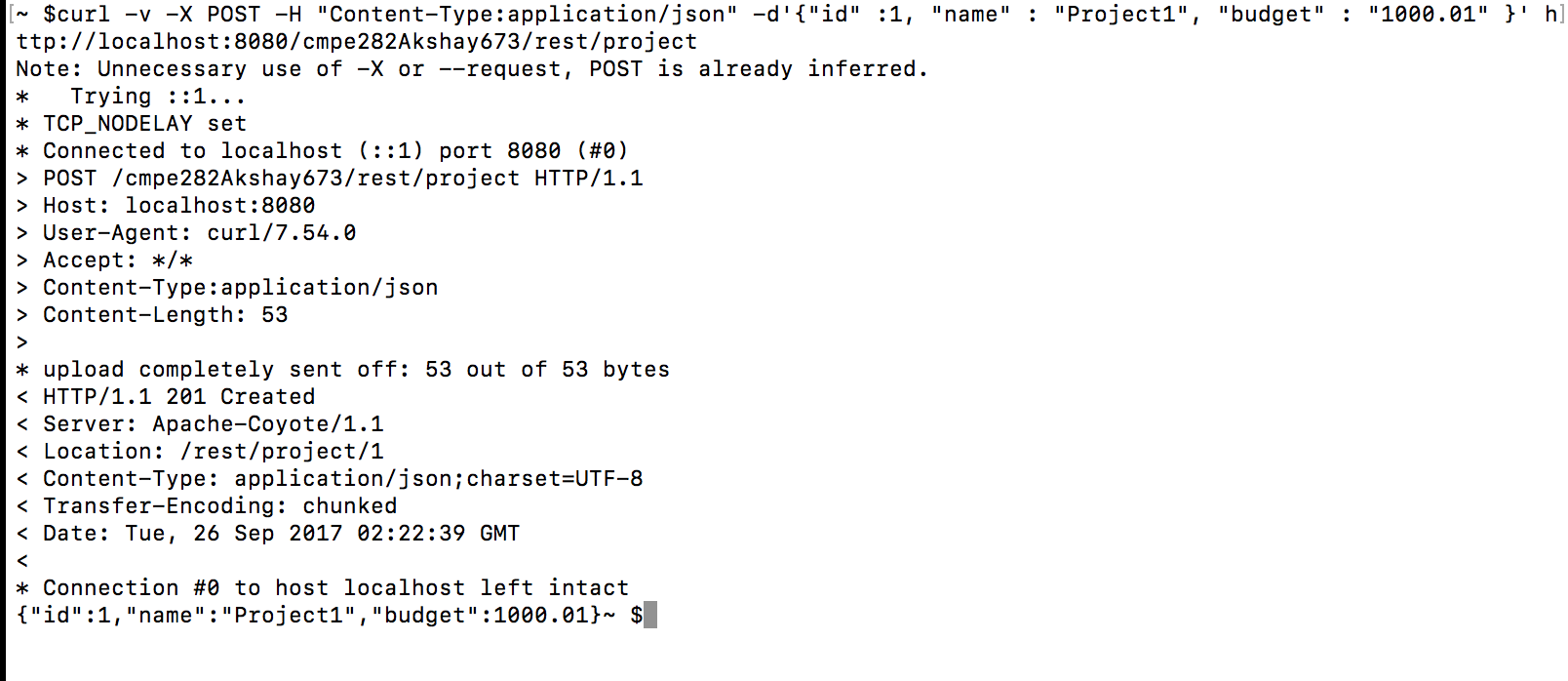
****

1. POST /.../rest/project

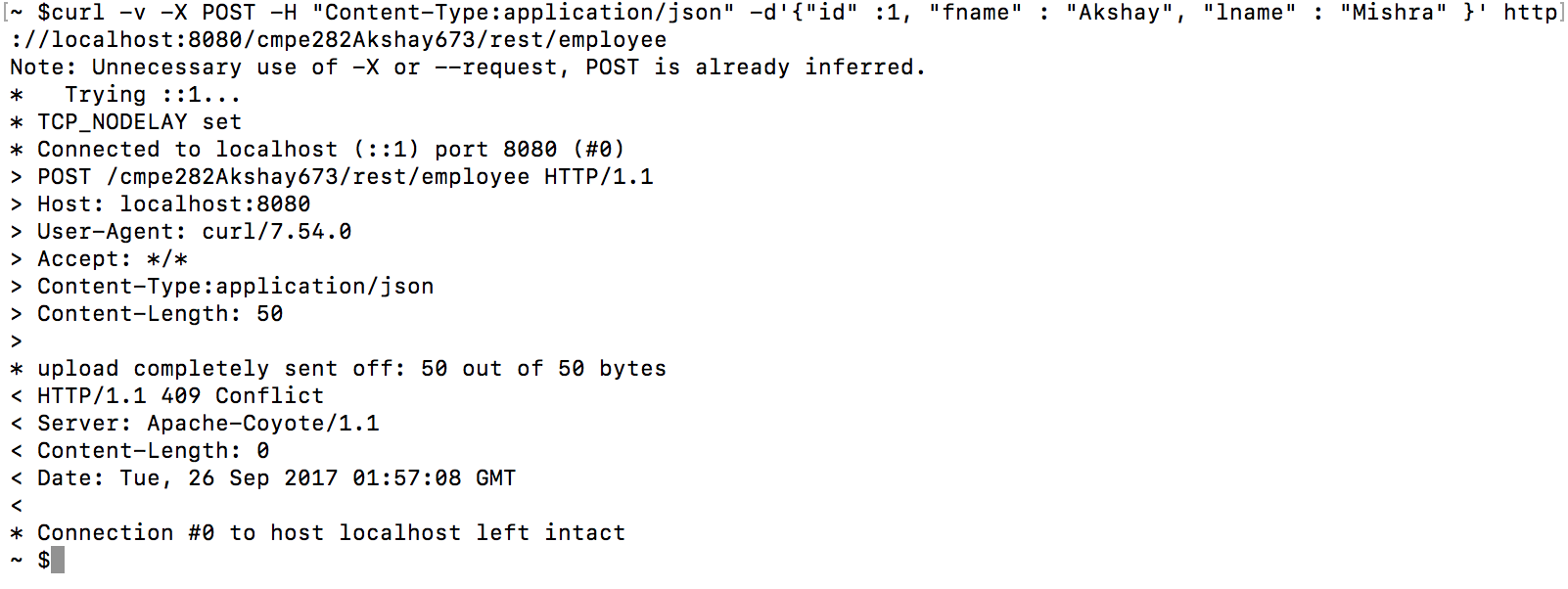
**Before:**

****

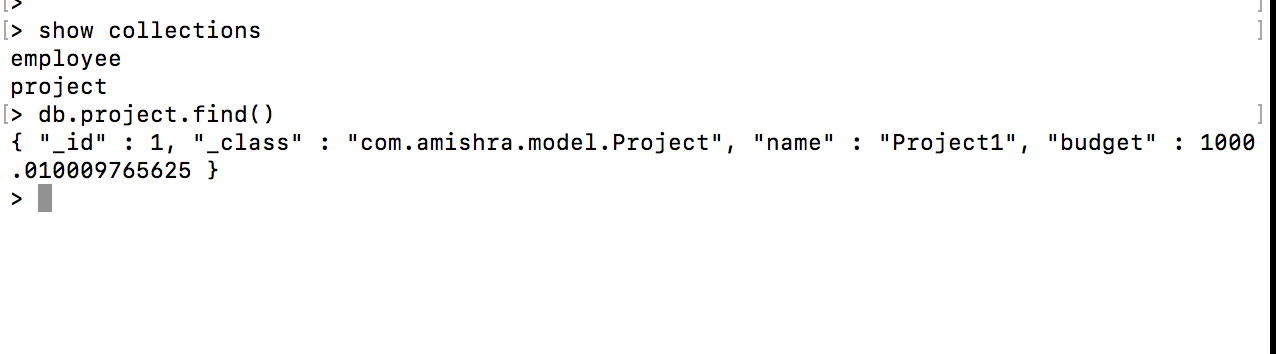
**Success:**

****

**Error:**

****

**After:**

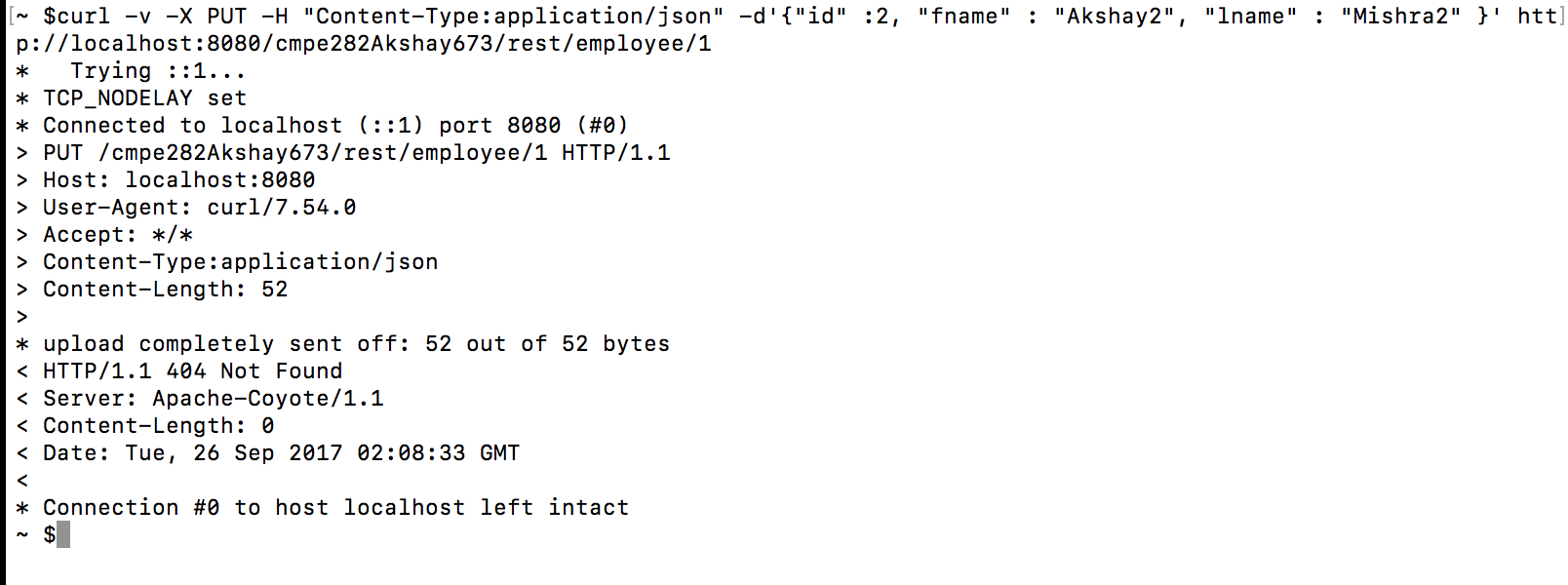
****

1. PUT /.../rest/employee/m

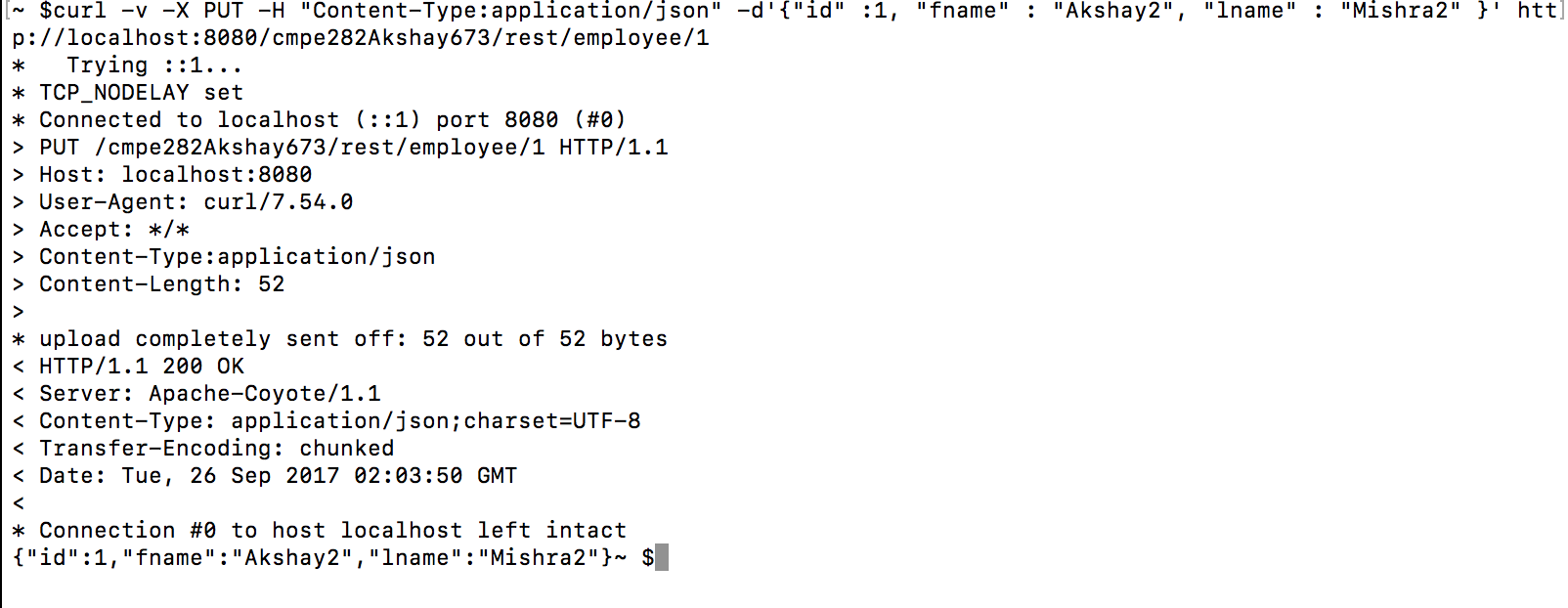
**Before:**

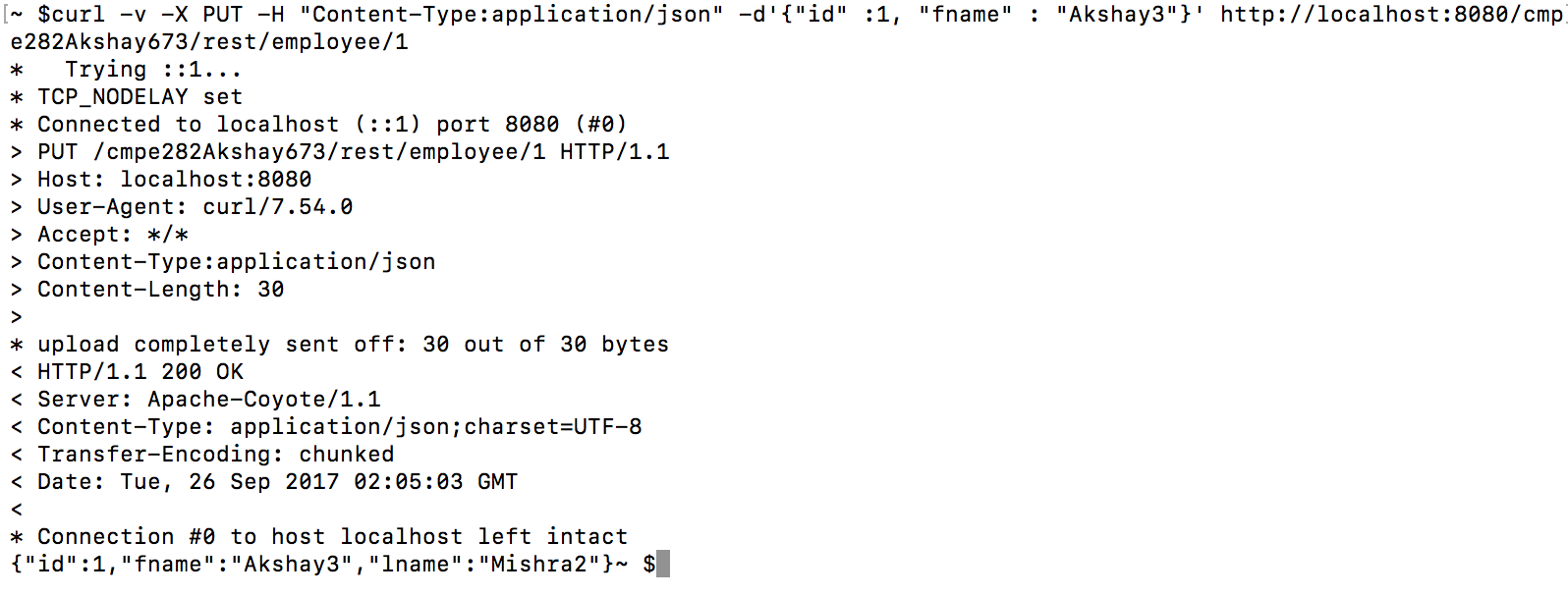
****

**Error:**

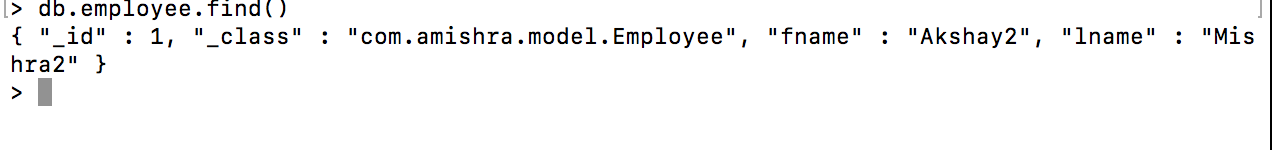
****

**Success:**

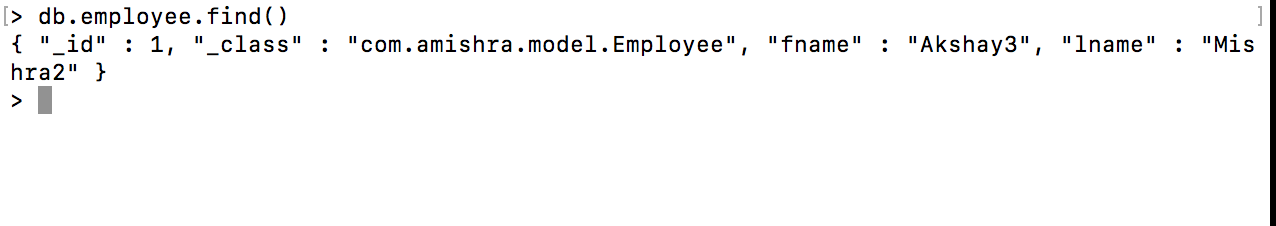
****

****

**After Full Update Success:**

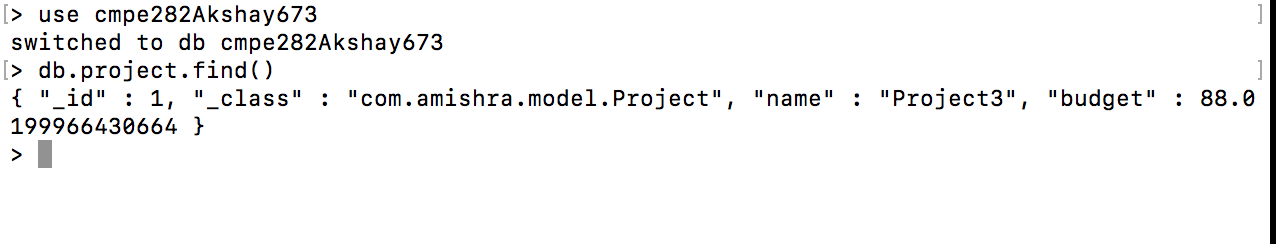
****

**After Partial Update Success:**

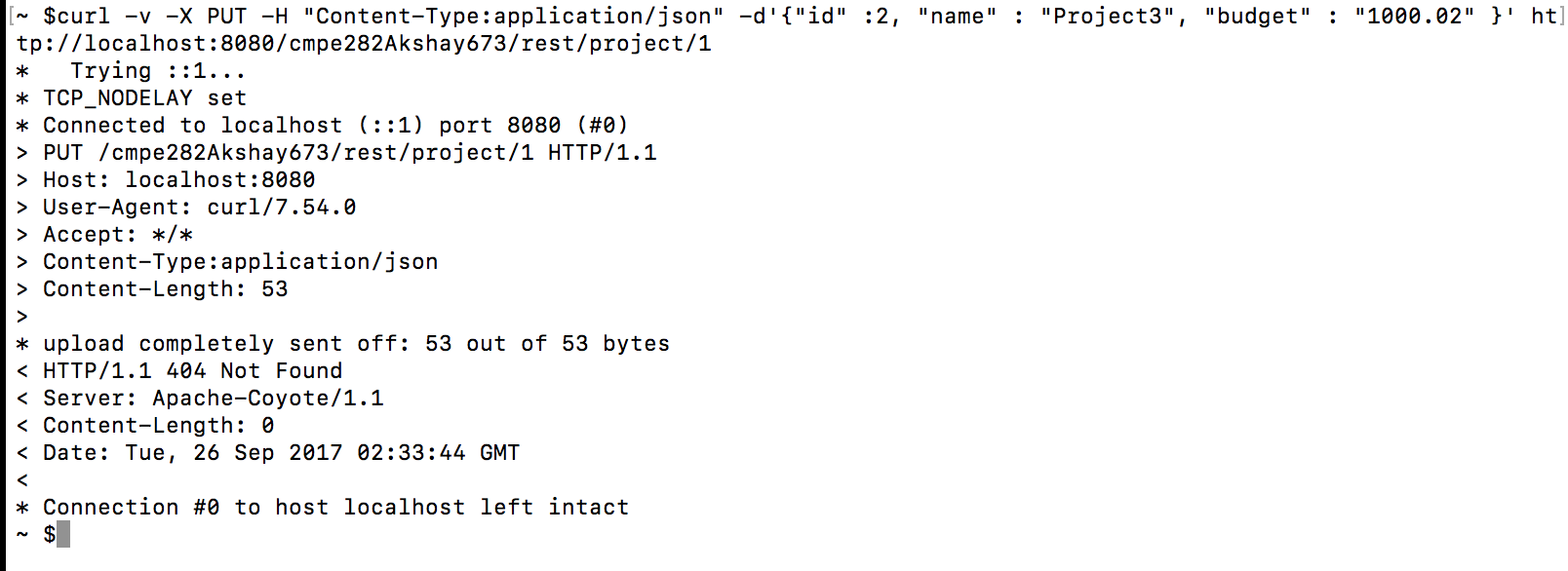
****

1. PUT /.../rest/project/n

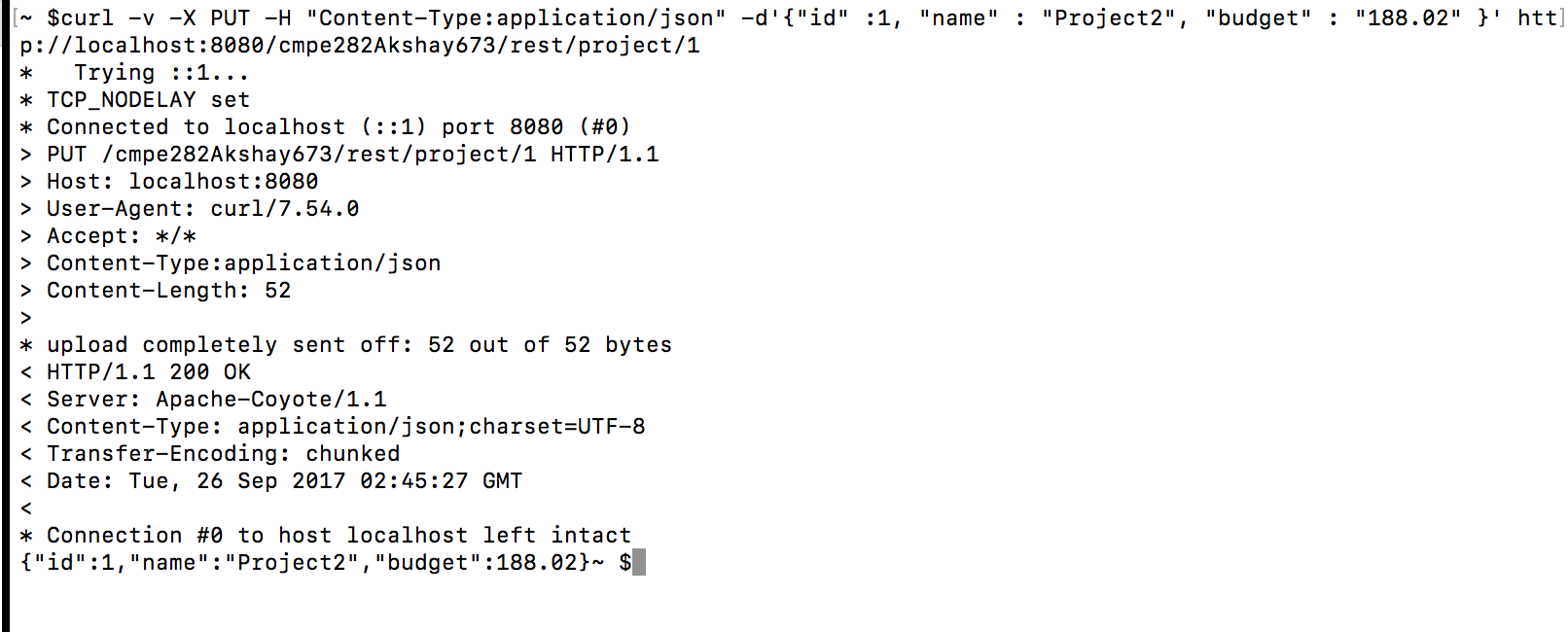
**Before:**

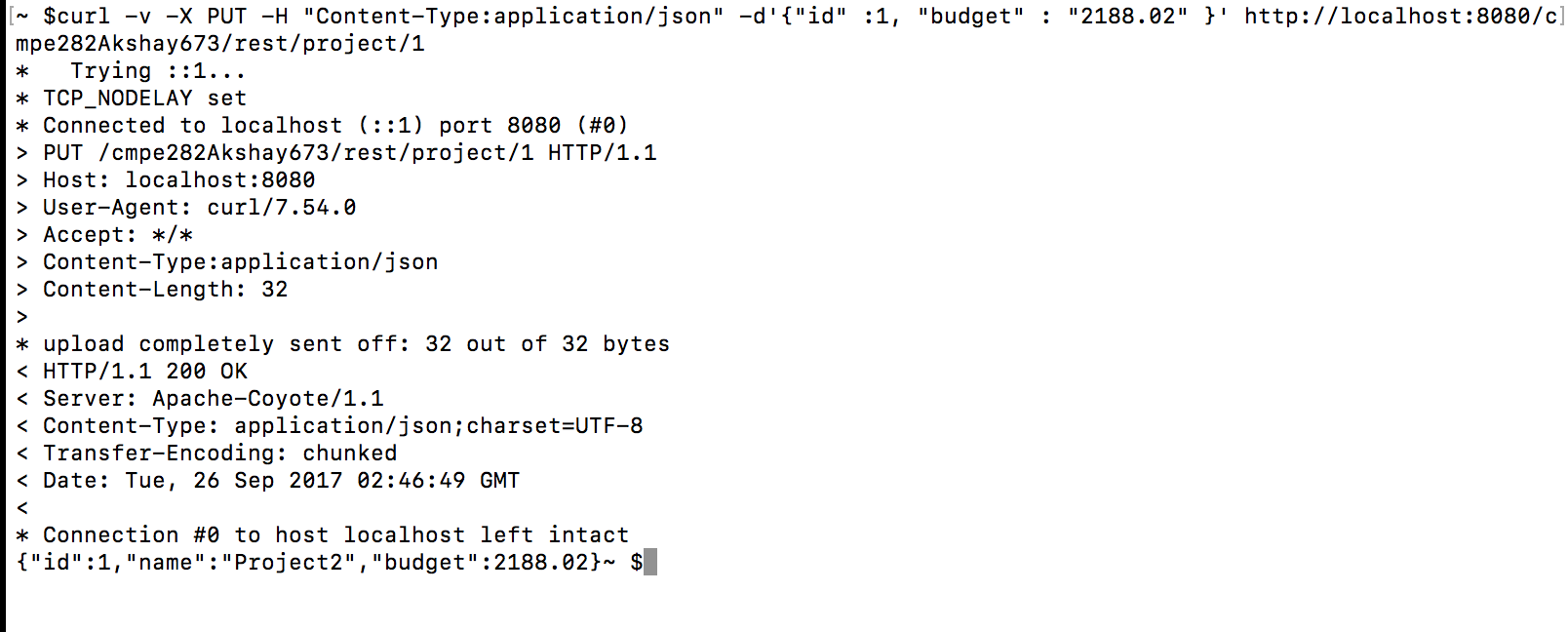
****

**Error:**

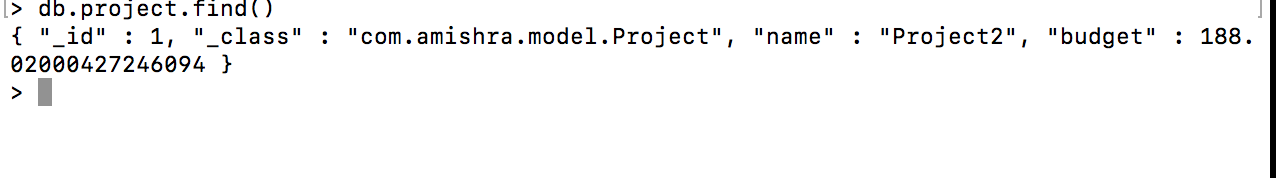
****

**Success:**

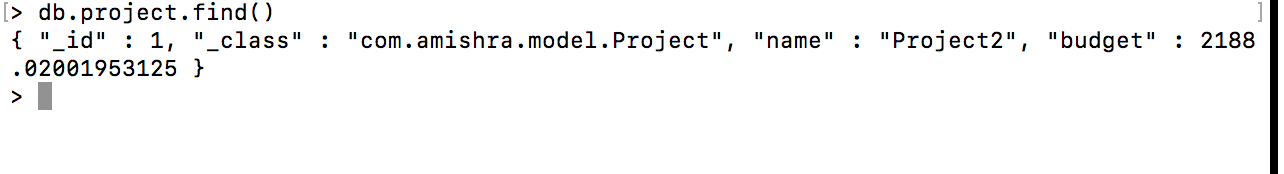
****

****

**After Full update Success:**

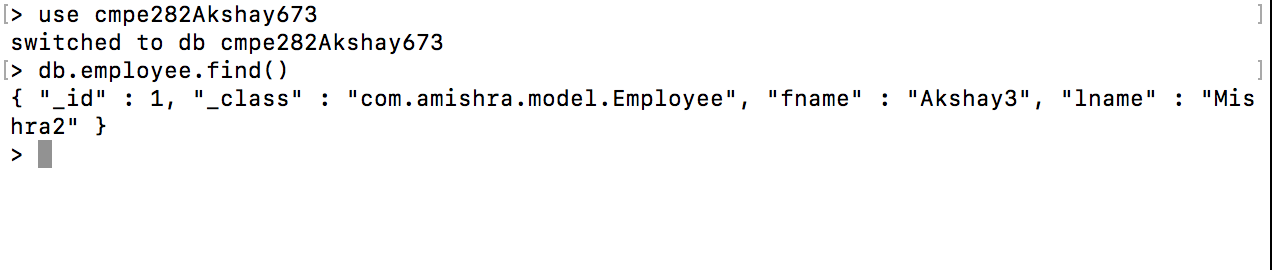
****

**After Partial update Success:**

****

1. DELETE /.../rest/employee/m

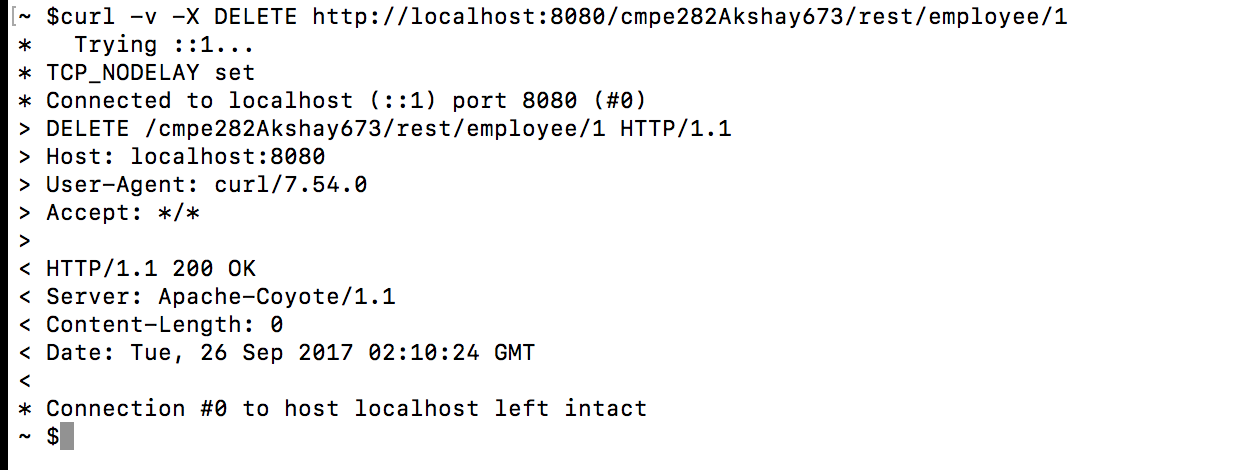
**Before:**

****

**Error:**

****

**Success:**

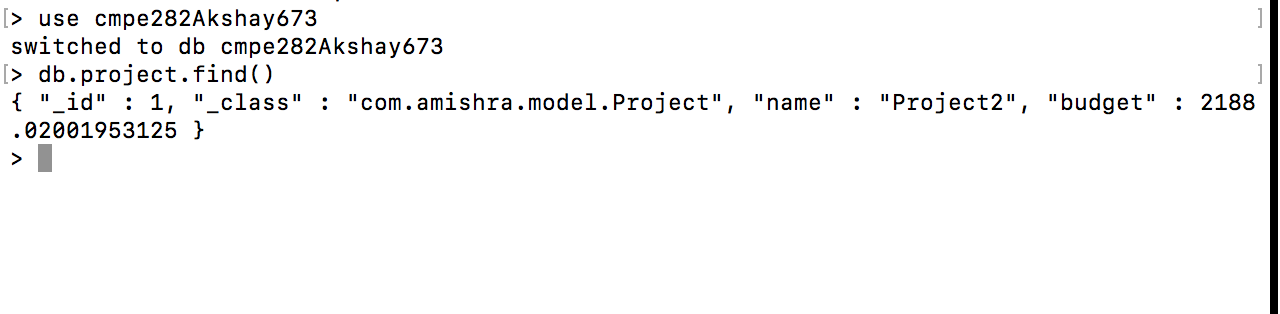
****

**After:**

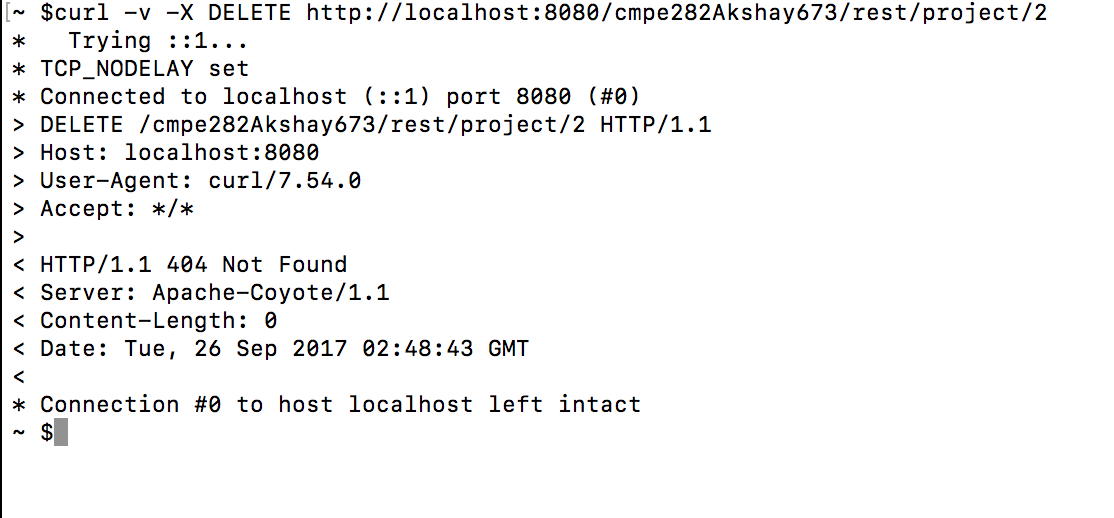
****

1. DELETE /.../rest/project/n

**Before:**

****

**Error:**

****

**Success:**

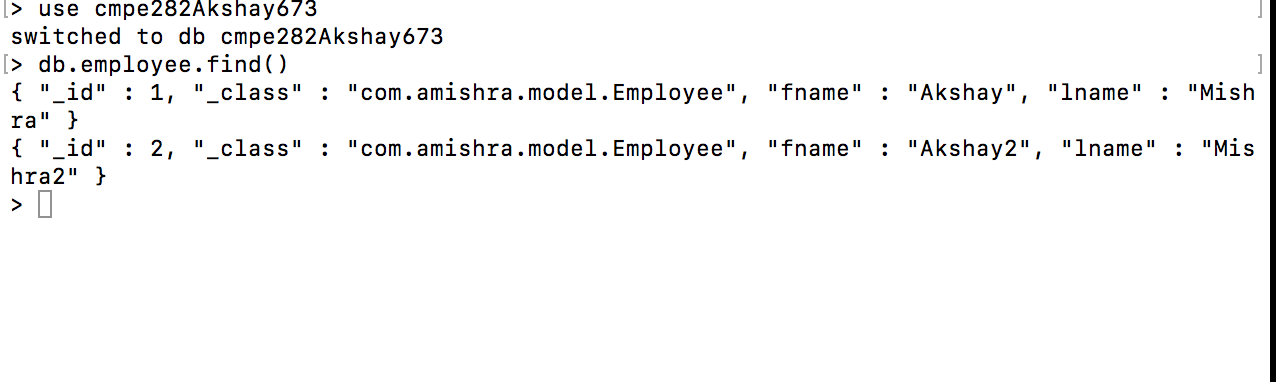
****

**After:**

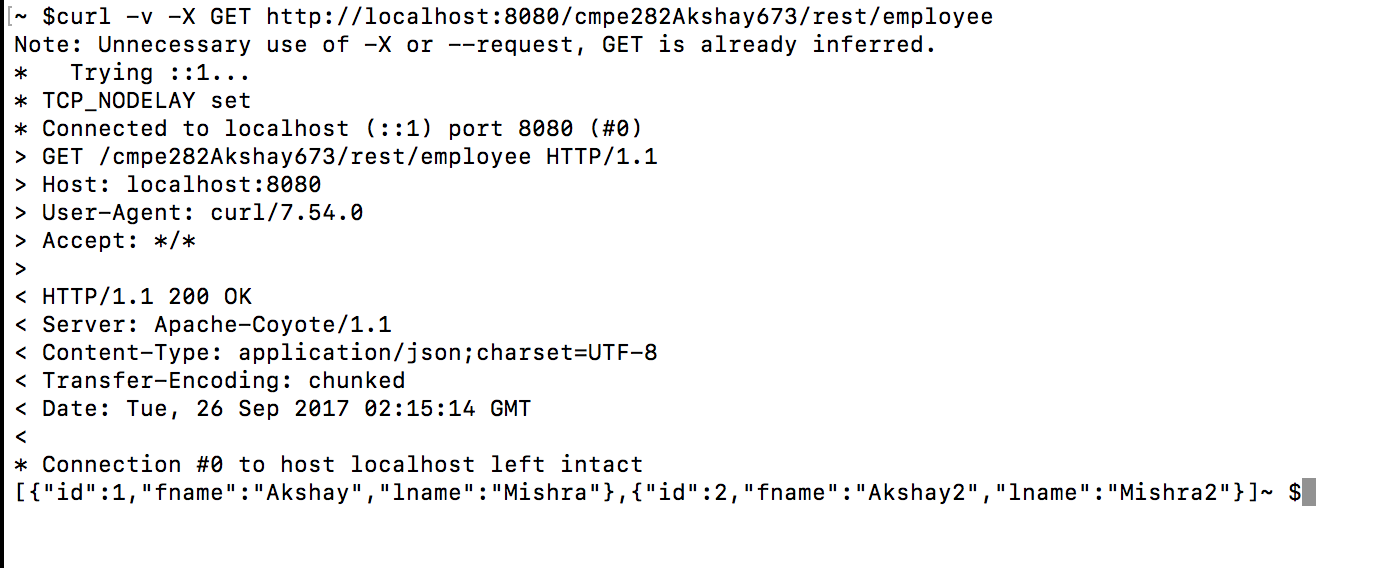
****

1. GET /.../rest/employee

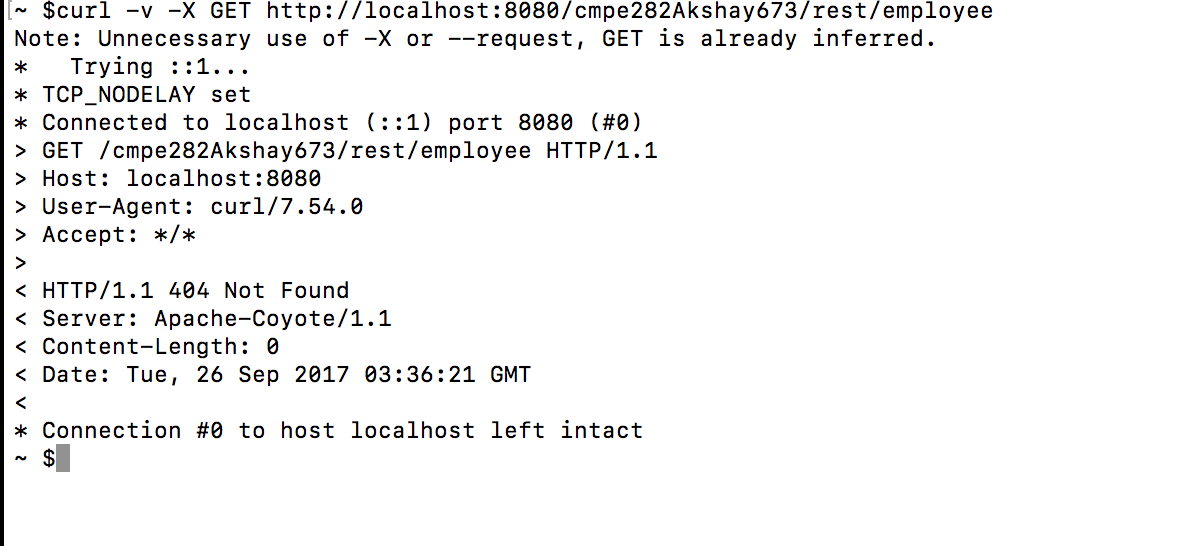
**Before:**

****

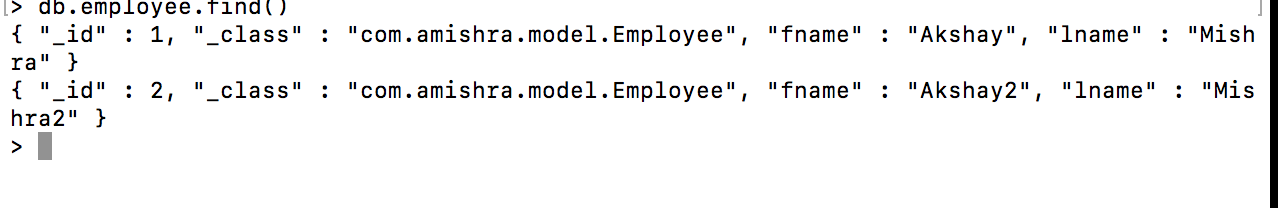
**Success:**

****

**Error: (some exception)**

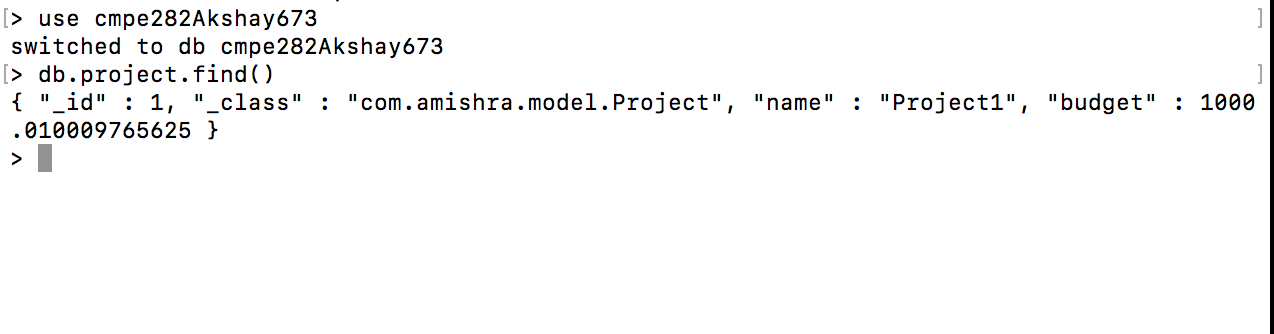
****

**After:**

****

1. GET /.../rest/project

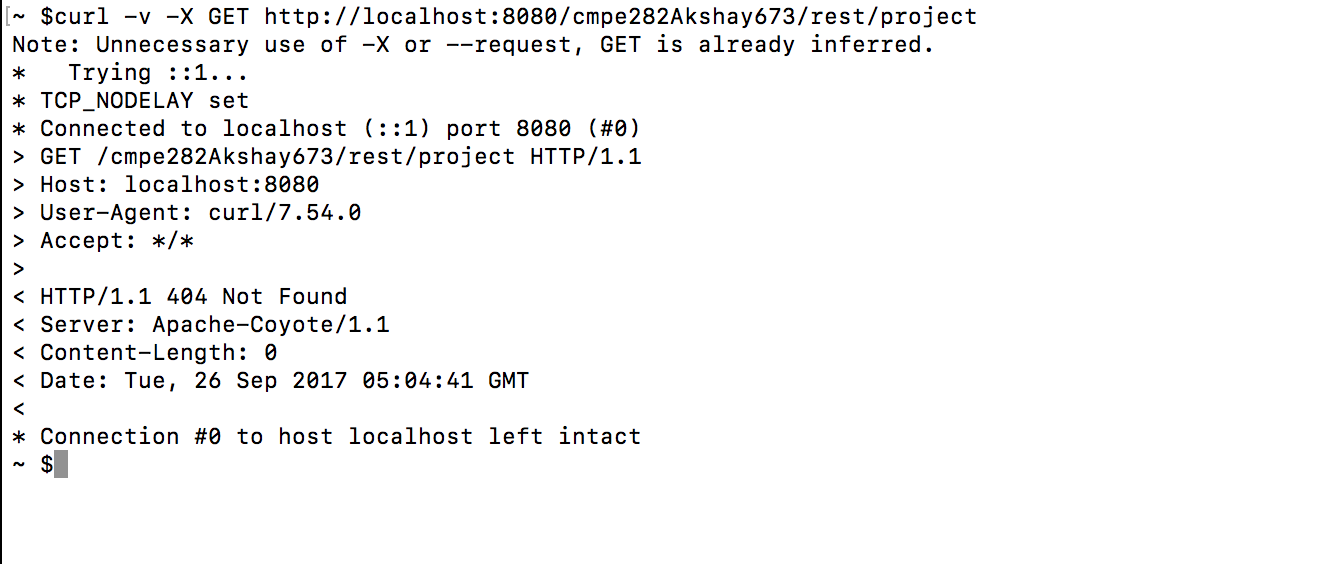
**Before:**

****

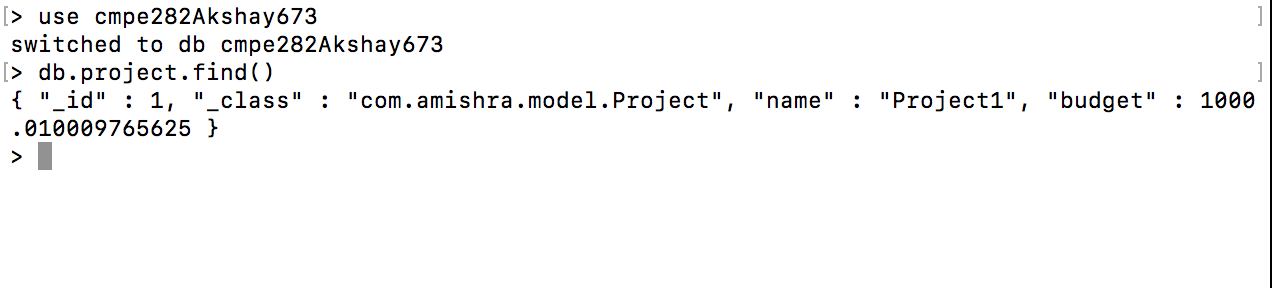
**Success:**

****

**Error:**

****

**After:**

****

**Known issues:** None

**Additional unique design or features:**

1. It uses a DAO to communicate with the DB, which adds another layer of abstraction for the application logic, clearly separating the concerns related to DB access from the controller.
2. No additional script is needed to create the collection in Mongo, At the time of insert, the collection will be created if not present already for that particular model.
3. Custom Exceptions defined, which send back a user-friendly message. These exceptions can also prevent data corruption. For example, a POST request with negative id will render a response that an id cannot be negative.