- ⏱ a minute
- ✅ Completed

Application Programming Interfaces Objectives

Modern Web applications can serve both HTML *and* data. You see that when you go to Facebook, Twitter, and TurboTax Online. There are some widely-accepted practices for how to architect your applications so that they can do this. After homework, lecture, and practicing, you should be able to

- Recall that REST is an acronym for Representational State Transfer
- Describe how RESTful endpoints differ from traditional remote-procedure call (RPC) services
- Identify and describe the RESTful meanings of the combinations of HTTP verbs and endpoint types for both HTML-based applications and APIs
  - HTTP verbs: GET, POST, PUT, PATCH, and DELETE
  - Endpoint types: collections of resources and singular resources
- Recall that RESTful is *not* a standard (like ECMAScript or URLs), but a common way to organize data interactions
- Explain how RESTful APIs are meant to be stateless
- Given a data model, design RESTful endpoints that interact with the data model to define application functionality
- Use the `express.json()` middleware to parse HTTP request bodies with type `application/json`
- Determine the maximum data an API response needs and map the content from a Sequelize object to a more limited data object
- Define a global Express error handler to return appropriate responses and status codes given a specific Accept header in the HTTP request
- Define Cross-Origin Resource Sharing (CORS) and how it is implemented in some Web clients
- Explain that CORS is an opt-in feature of Web clients
- Configure your API to use CORS to prevent unauthorized access from browser-based Web requests

Did you find this lesson helpful?