Consuming REST Services



Brice Wilson

@brice_wilson www.BriceWilson.net



What is a REST Service?



Representational State Transfer

Web API or HTTP API

Uses HTTP verbs to specify CRUD operations

URL conventions address individual as well as collections of resources

HTTP response codes indicate success/failure of server action



R



C

Create

- POST http://localhost/api/books
- If successful, returns HTTP 201 Created

R

U

D

C

R

U

D

Create

- POST http://localhost/api/books
- If successful, returns HTTP 201 Created

Read

- GET http://localhost/api/books OR
 http://localhost/api/books/5
- If successful, returns HTTP 200 OK

C

R

U

D

Create

- POST http://localhost/api/books
- If successful, returns HTTP 201 Created

Read

- GET http://localhost/api/books OR
 http://localhost/api/books/5
- If successful, returns HTTP 200 OK

Update

- PUT http://localhost/api/books/5
- If successful, returns HTTP 204 No Content

C

R

U

D

Create

- POST http://localhost/api/books
- If successful, returns HTTP 201 Created

Read

- GET http://localhost/api/books OR
 http://localhost/api/books/5
- If successful, returns HTTP 200 OK

Update

- PUT http://localhost/api/books/5
- If successful, returns HTTP 204 No Content

Delete

- DELETE http://localhost/api/books/5
- If successful, returns HTTP 204 No Content

```
getAllBooks(): Observable<Book[]> {
   return this.http.get<Book[]>('/api/books');
}
```



```
getAllBooks(): Observable<Book[]> {
   return this.http.get<Book[]>('/api/books');
}
```



```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
}
```



```
getAllBooks(): Observable<Book[]> {
   return this.http.get<Book[]>('/api/books');
}
```



```
getAllBooks(): Observable<Book[]> {
   return this.http.get<Book[]>('/api/books');
}
this.dataService.getAllBooks()
   .subscribe(
```



```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
this.dataService.getAllBooks()
  .subscribe(
    (data: Book[]) => this.allBooks = data,
```



```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
this.dataService.getAllBooks()
  .subscribe(
    (data: Book[]) => this.allBooks = data,
    (err: any) => console.log(err),
```

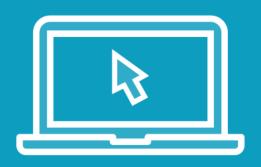
```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
this.dataService.getAllBooks()
  .subscribe(
    (data: Book[]) => this.allBooks = data,
    (err: any) => console.log(err),
    () => console.log('All done getting books.')
```



```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
this.dataService.getAllBooks()
  .subscribe(
    (data: Book[]) => this.allBooks = data,
    (err: any => console.log(err),
    () => constle.log('All done getting books.')
```

```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
this.dataService.getAllBooks()
  .subscribe(
    (data: Book[]) => this.allBooks = data,
    (err: any => console.log(err),
    () => constle.log('All done getting books.')
```

Demo



Retrieving a collection from a RESTful service



Demo



Retrieving a single item from a RESTful service



Using RxJS Operators

Operate on an Observable and return an Observable

May be chained together to perform complex transformations

Flexibility to transform data into exactly the shape you need



Demo



Transforming data with RxJS



Demo



Creating, updating, and deleting data in a RESTful service

