Handling More Complex Scenarios

Shawn Wildermuth
MICROSOFT MVP, INSTRUCTOR AND FILMMAKER

@shawnwildermuth https://wilderminds.com

Agenda

Handling More Complex Scenarios

- Associated Resources
- Paging
- Error Handling
- Caching
- Functional APIs
- Async APIs

Designing Associations

```
/api/customers/123/Invoices

/api/games/halo-3/ratings

/api/invoices/2003-01-24/payments
```

- **For sub-objects – Use URI Navigation**

```
/api/customers/123/invoices

/api/invoices
```

- **Should return List – Same Shapes**

Designing Associations

```
/api/customers/123/invoices

/api/customers/123/payments

/api/customers/123/shipments
```

◂ **Can have multiple associations**

Designing Associations

```
/api/Customers?st=GA

/api/Customers?st=GA&salesid=144

/api/Customers?hasOpenOrders=true
```

◀ **Search should use queries**

Demo

Associations

Paging

- Lists should support paging
- Query strings are commonly used:

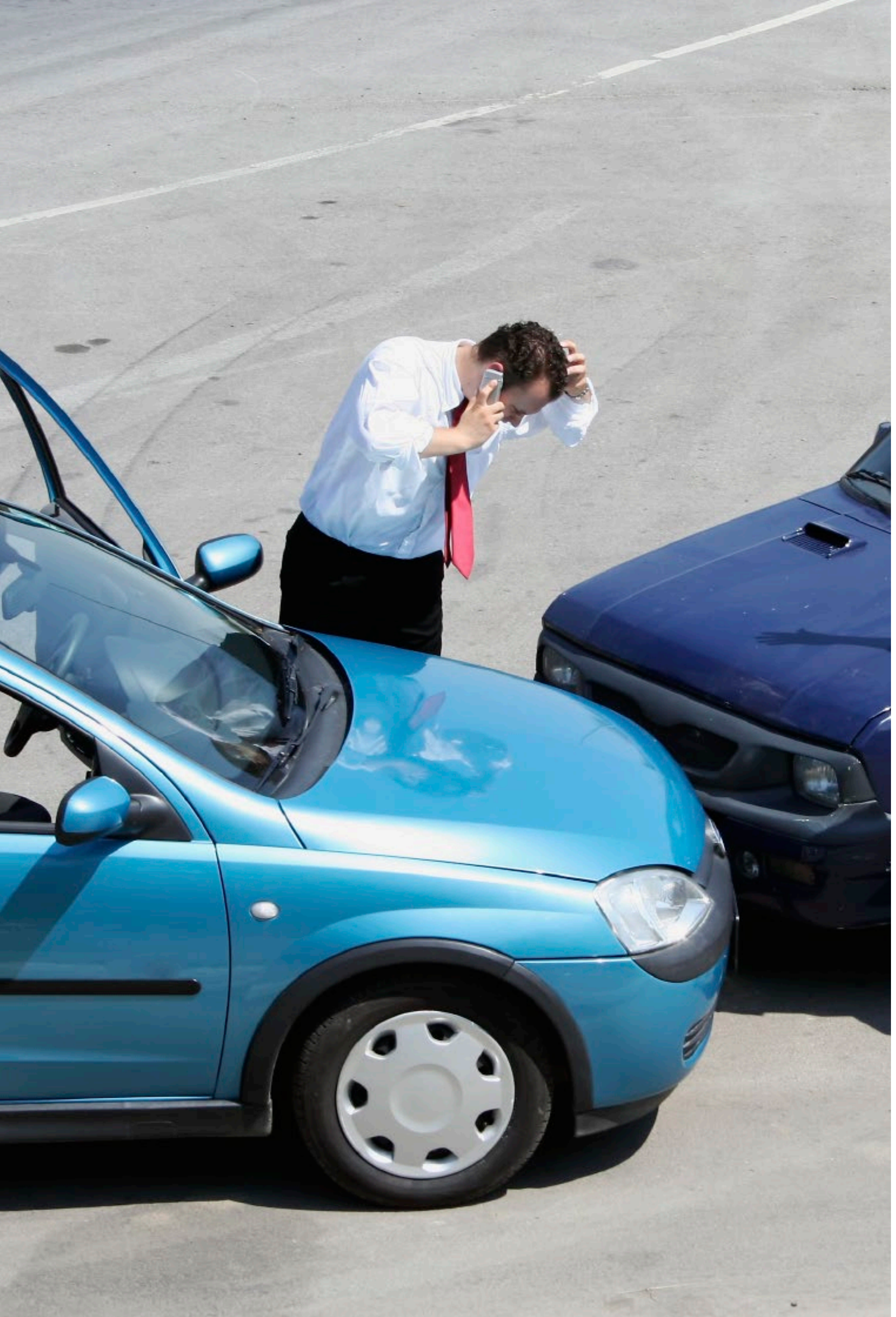
```
/api/sites?page=1&page_size=25
```

- Use wrappers to imply paging:

```
{
  totalResults: 255,
  nextPage: "/api/sites?page=5",
  prevPage: "/api/sites?page=3",
  results: [...]
}
```

Demo

Paging



Error Handling:

- Not just status codes
- How to you communicate errors
- How do you help the user recover

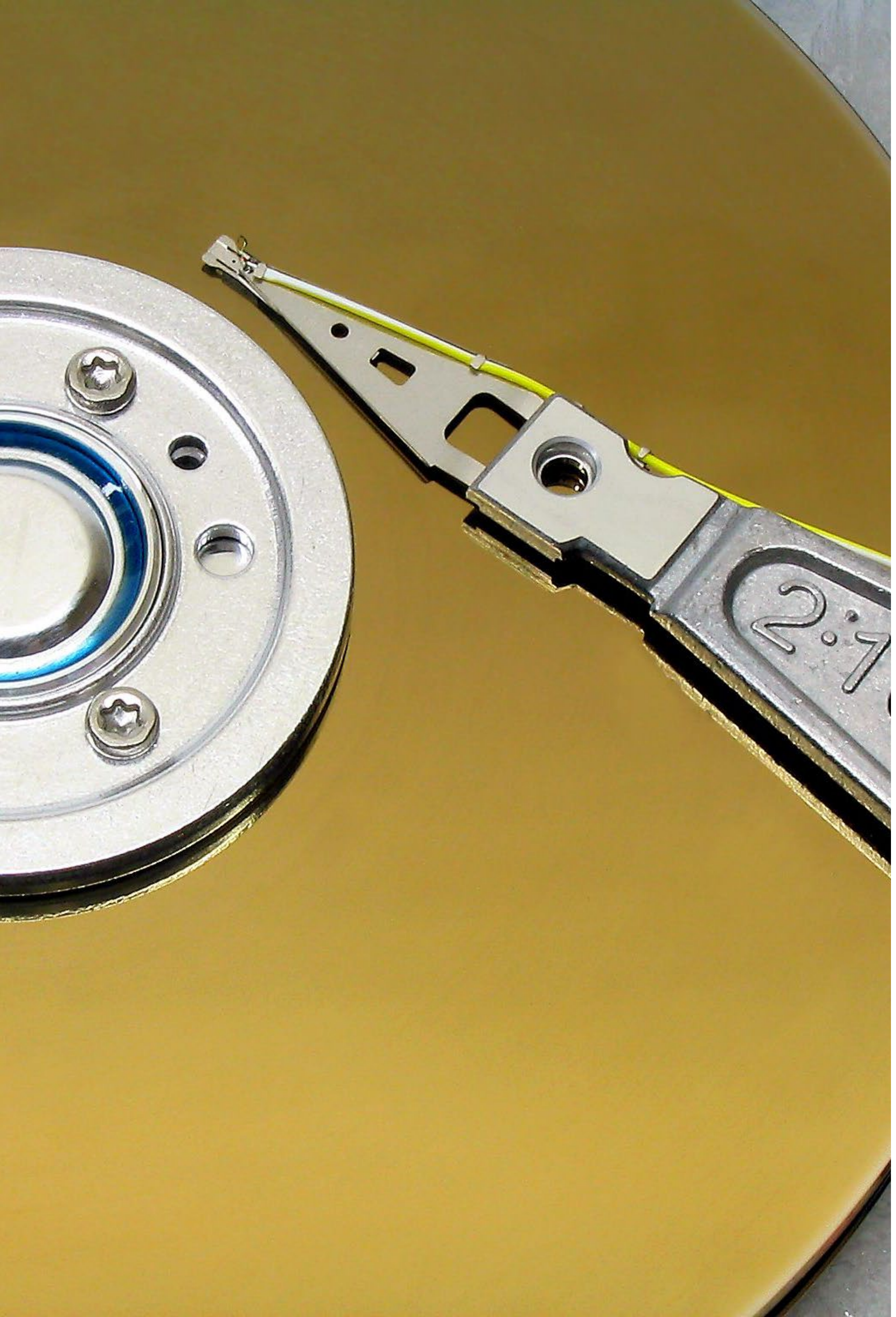
Error Handling

```
400 Bad Request

{ error: "Failed to supply id" }


404 Not Found
```

- Return object with error info
- Not necessary for obvious errors

Demo

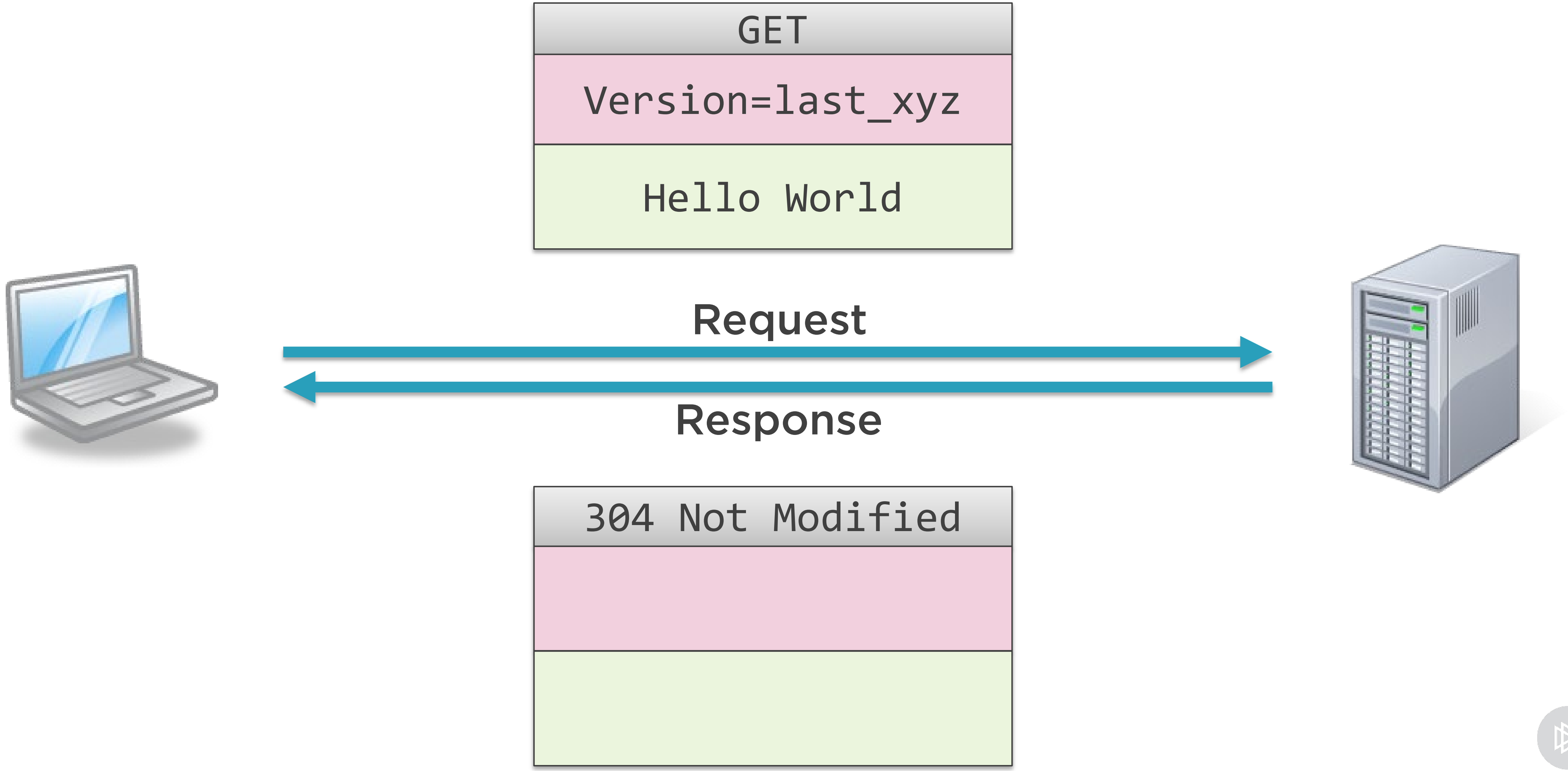
Error Handling



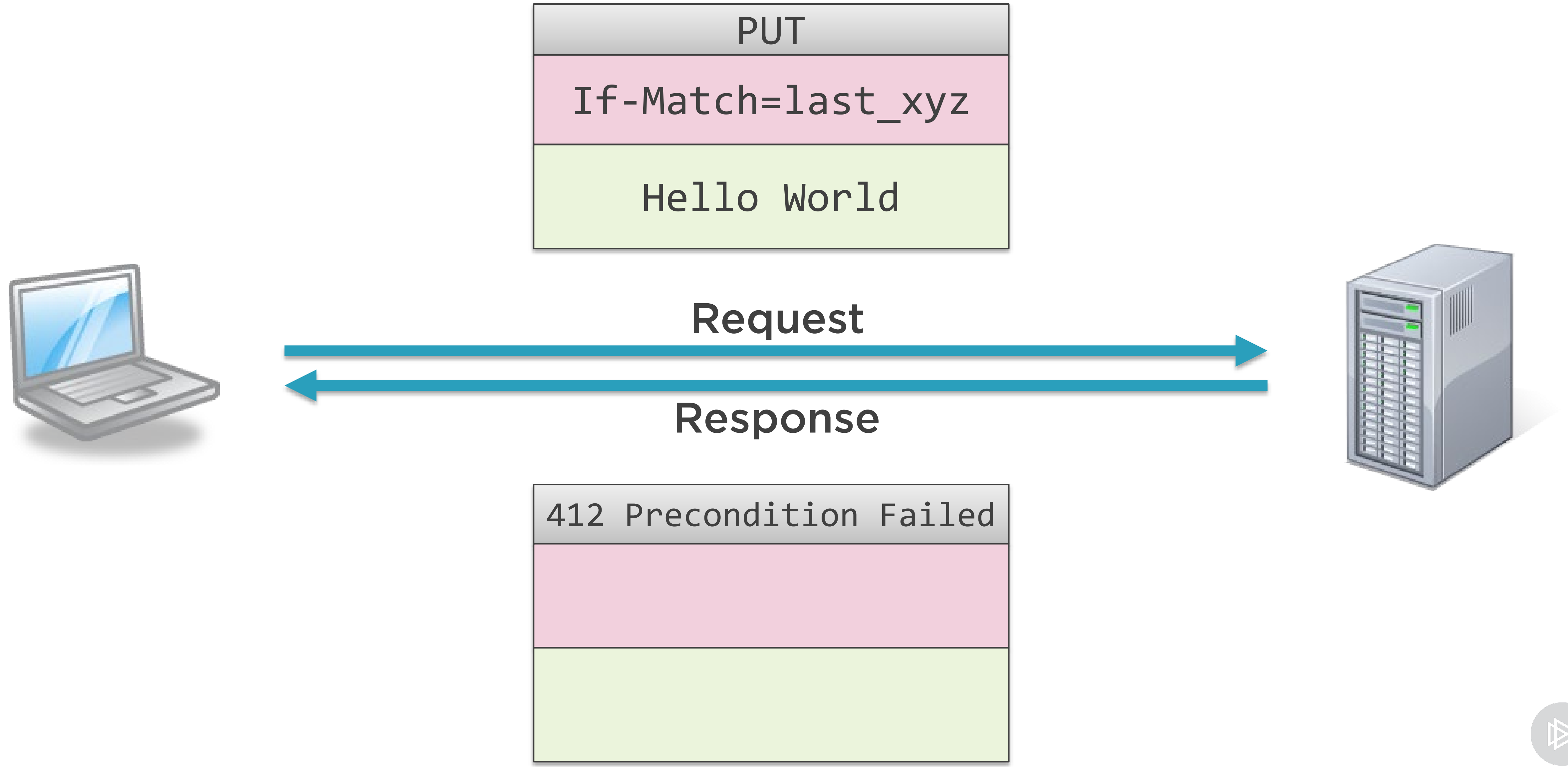
Caching

- Basic Tenet of REST APIs
- Server-side caching is good
- But isn’t what they mean
- Use HTTP for caching mechanism

HTTP Caching



HTTP Caching



Entity Tags (ETags)

- Strong and Weak Caching Support
- Returned in the Response

```
HTTP/1.1 200 OK
Content-Type: text/xml;
Date: Thu, 23 May 2013 21:52:14 GMT
ETag: W/"4893023942098"
Content-Length: 639
```

Entity Tags (ETags)

- Request with If-None-Match

```
GET /api/games/2 HTTP/1.1
Accept: application/json, text/xml
Host: localhost:8863
If-None-Match: "4893023942098"
```

- Use 304 to indicate that it's cached

```
HTTP/1.1 304 Not Modified
```

Entity Tags (ETags)

- For PUT/DELETE

```
PUT /api/games/2 HTTP/1.1
Accept: application/json, text/xml
Host: localhost:8863
If-Match: "4893023942098"
...
```

- Use 412 to indicate that not same

```
HTTP/1.1 412 Precondition Failed
```

Demo

Caching with ETags

Functional APIs

- Be pragmatic
- Make sure these are documented
- Should be completely functional
- Not an excuse to build an RPC API

```
/api/calculateTax?state=GA&total=149.99

/api/restartServer?isColdBoot=true

/api/beginWorldDomination?isVolcanoLairRequired=true
```

Functional APIs

Should be the exception rather than the rule…

Demo

Functional APIs

Async APIs

- Some APIs aren't RESTful in nature
- Need long-life, polling
- Non-REST Solutions are useful

Async API Solutions to Consider

- Comet
- gRPC
- SignalR
- Firebase
- Socket.IO
- Etc.

What We've Learned

Design Associations to make your API more intuitive

Caching is a basic part of REST and you must plan for it

Functional APIs are important but should be the exception

Coming Up:
Versioning Your API