Working with Data in an Angular Application

Deborah Kurata

@deborahkurata
http://blogs.msmvps.com/deborahk
deborahk@insteptech.com

Deborah Kurata

- Independent Consultant | Developer | Mentor
  - Web (Angular), .NET
- Pluralsight Author
  - AngularJS Line of Business Applications
  - Angular Front to Back with Web API
  - Object-Oriented Programming Fundamentals in C#
- Microsoft MVP

Session Materials & Sample Code

https://github.com/DeborahK/ab2015

Rate Yourself on Data Access

- New to Angular - no proficiency
- Just starting out - limited proficiency
- Doing it but not fully understanding it - working proficiency
- Been there, done that, can help others - full proficiency

Overview

Request/Response flow

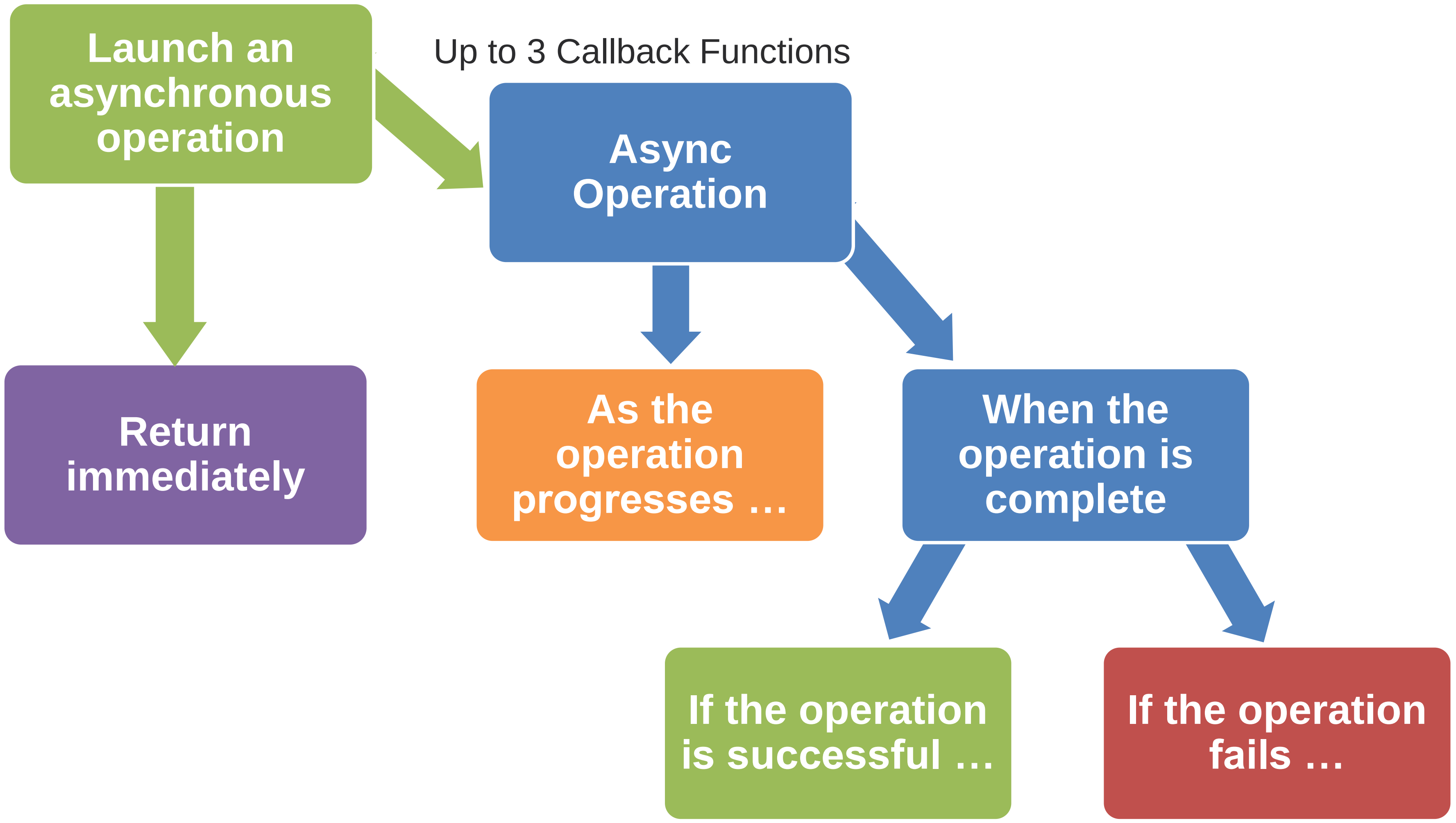
Angular services for accessing data

Building a reusable data access service

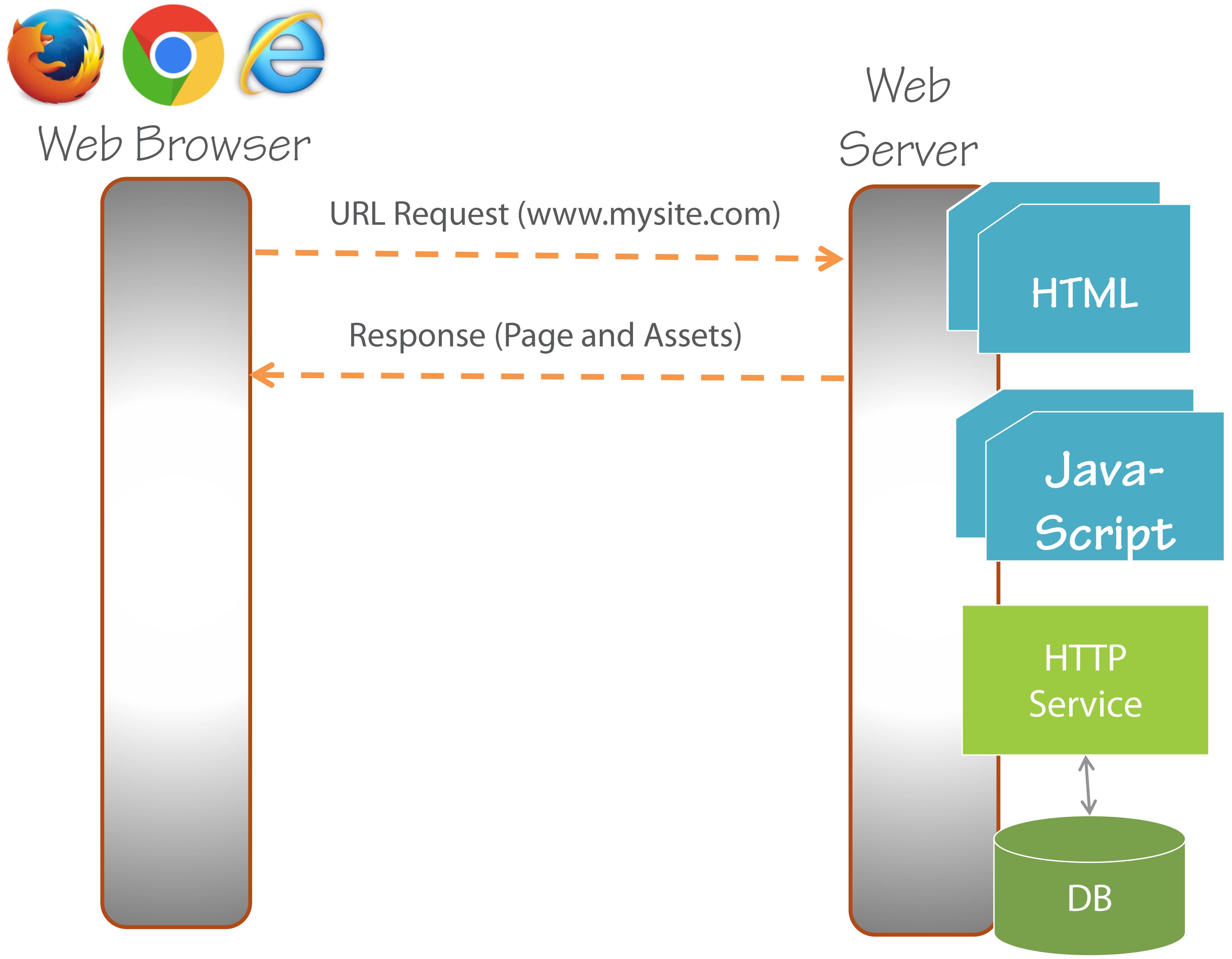
Faking data access

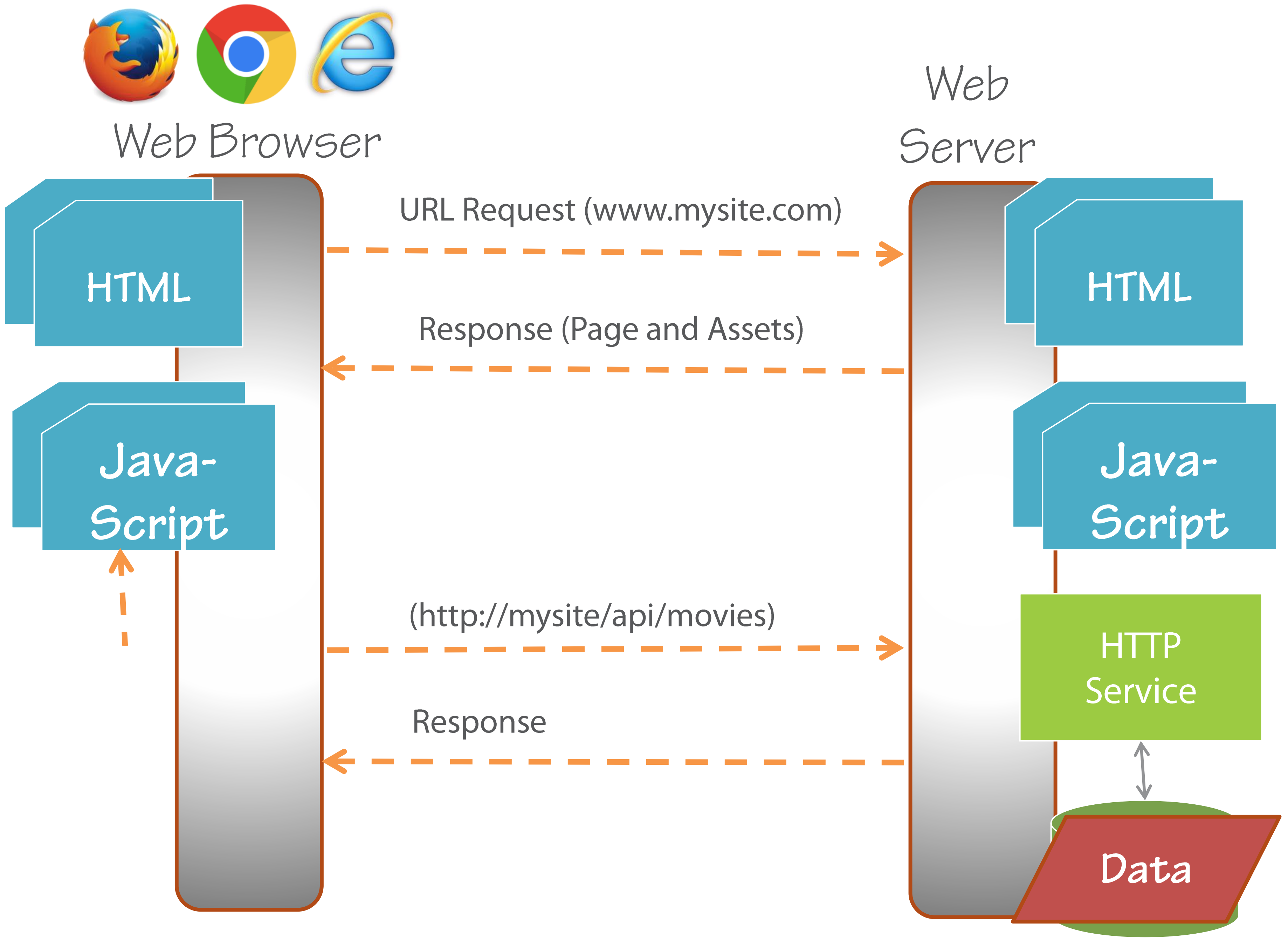
OData queries

JavaScript Promise

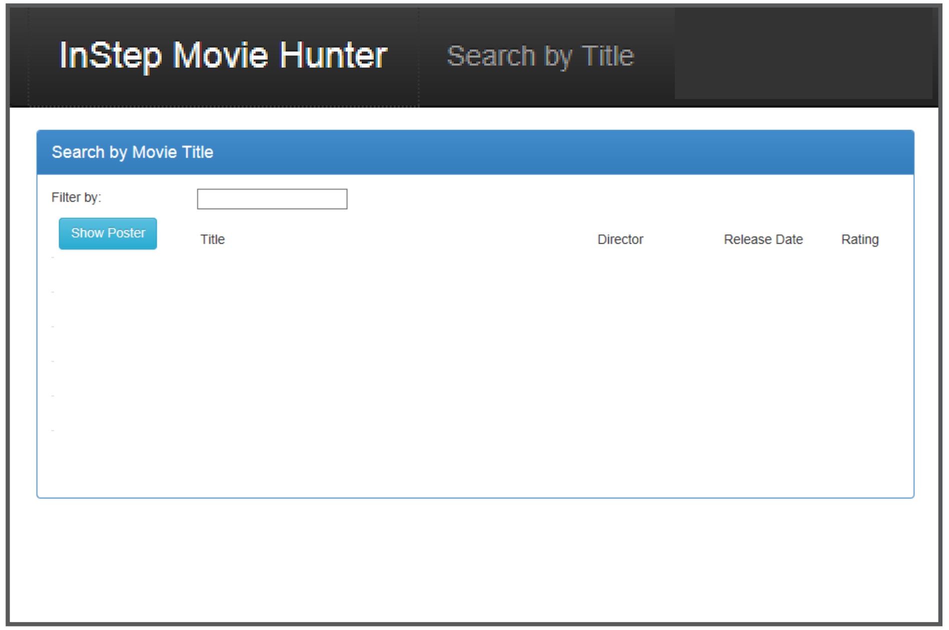

Launch an asynchronous operation
Up to 3 Callback Functions
Async Operation
Return immediately
As the operation progresses …
When the operation is complete
If the operation is successful …
If the operation fails …

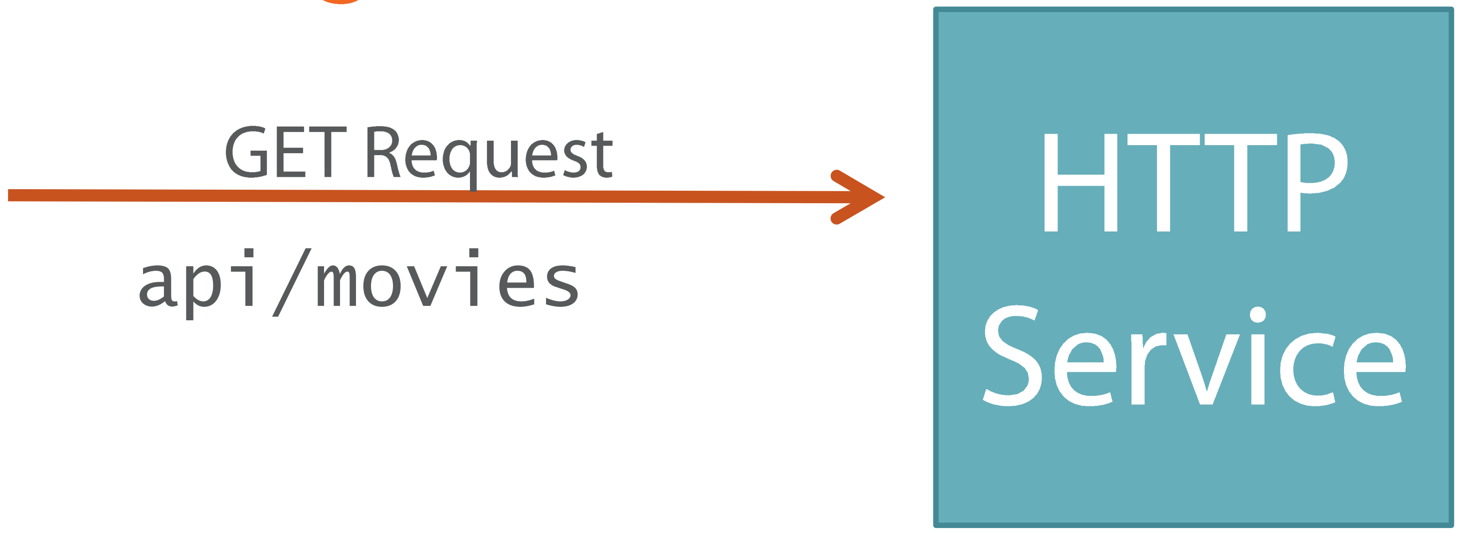
Demo: Angular in Action


Web Browser
Web Server
URL Request (www.mysite.com)
Response (Page and Assets)
HTML
Java-Script
HTTP Service
DB

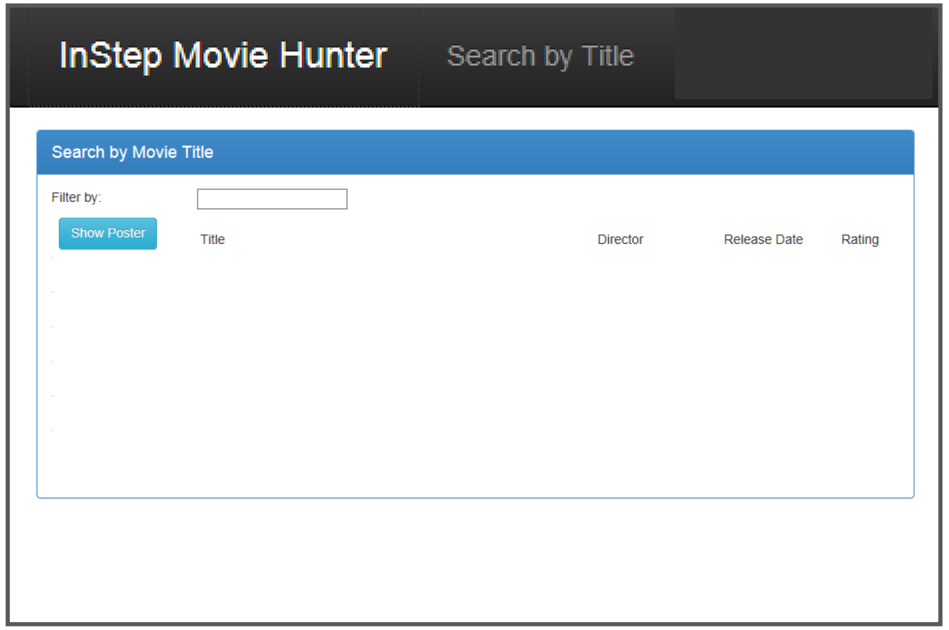

Web Browser
Web Server
HTML
Java-Script
URL Request (www.mysite.com)
Response (Page and Assets)
(http://mysite/api/movies)
Response
HTML
Java-Script
HTTP Service
Data

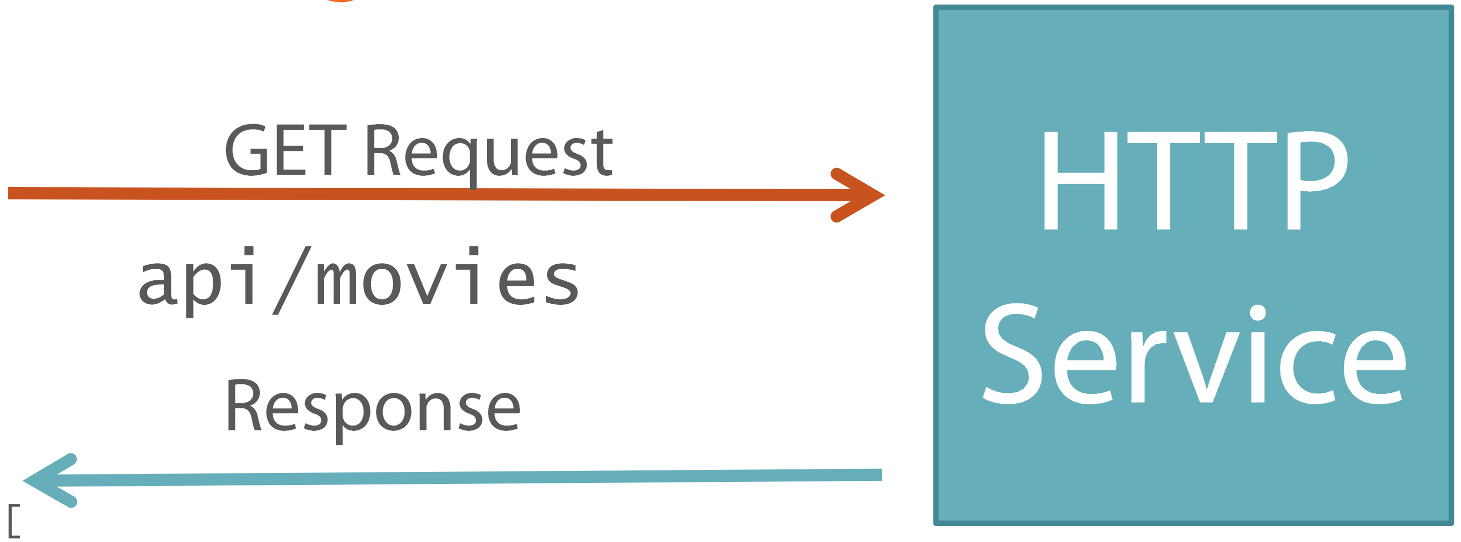
Retrieving Data





Retrieving Data





```
[
 {  director: "Peter Jackson",
    movieId: 1,
    mpaa: "pg-13",
    releaseDate: "2001-12-19T00:00:00",
    title: "The Lord of the Rings: The Fellowship of the Ring"
  },
  { director: "Peter Jackson",
    movieId: 2,
    mpaa: "pg-13",
    releaseDate: "2002-12-18T00:00:00",
    title: "The Lord of the Rings: The Two Towers"
   },
   ...
]
```

Angular Services for Accessing Data

$http

$resource

$httpBackend

```
$http.get("/api/movies/")
     .then(function(response) {
          vm.movies = response.data;
      });
```

$http

Built into Angular Core

Facilitates communication with a back-end service (get, post, put, delete, …)

Asynchronous

REST

- REpresentational State Transfer
- Requests and responses involve the transfer of resources
- Resources are identified with a URL

  `/api/movies/`
- Requests utilize a standard set of HTTP verbs

  `GET | POST | PUT | DELETE`
- And much more!

```
function movieResource($resource) {
  return $resource("/api/movies/:movieId")
        }
```

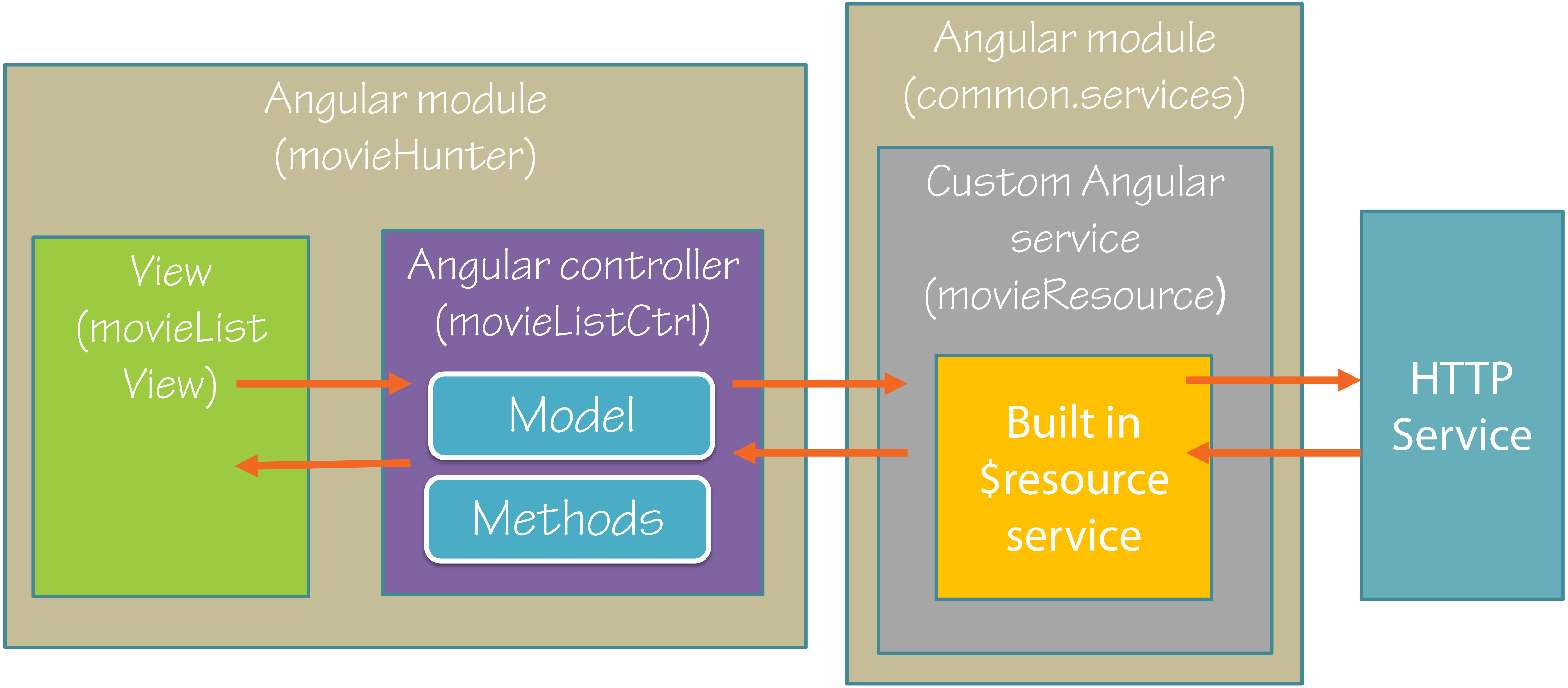
$resource

Separate Angular component: angular-resource (ngResource)

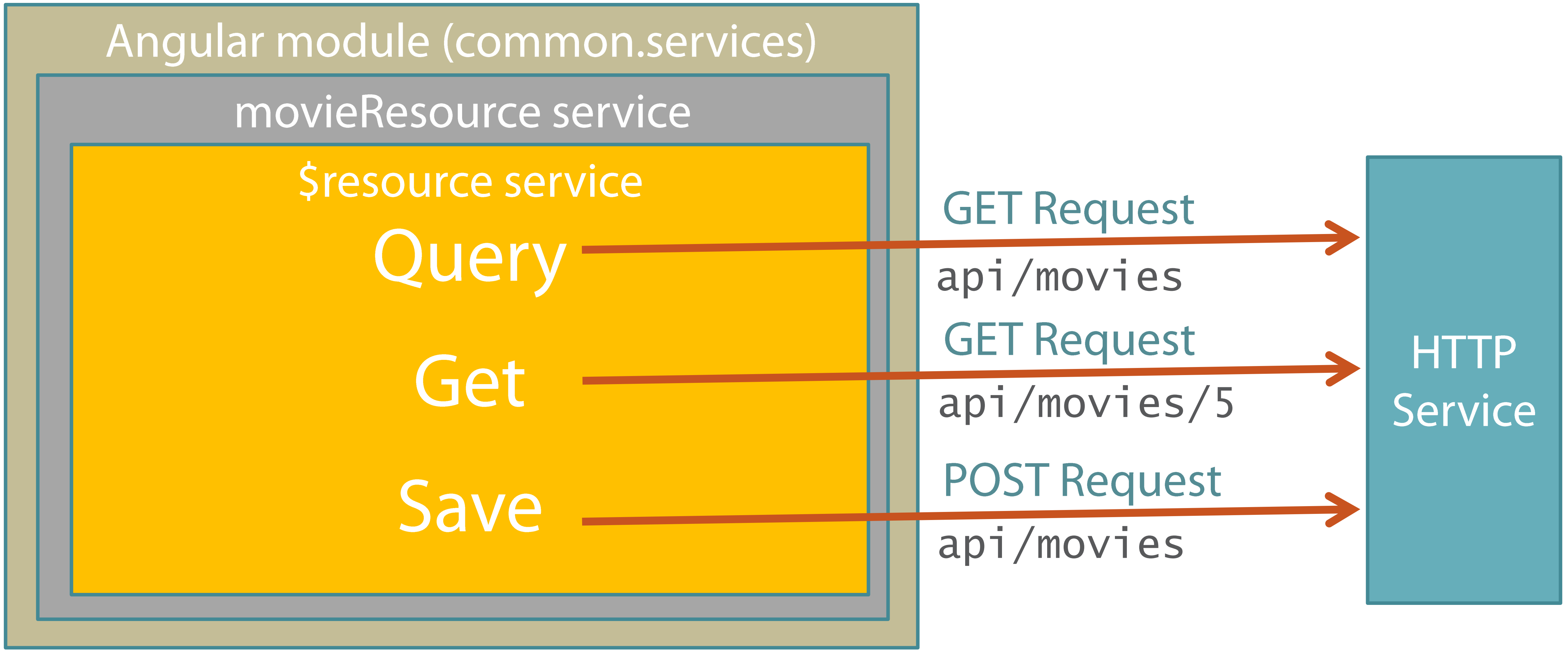
Abstraction on top of $http for calling RESTful services

Requires less code

Reusable Data Access Service



Built-In $resource Methods



$resource Methods

Method Name	Verb	URL	Description
query	GET	/api/movies	Retrieves all of the movies
get	GET	/api/movies/5	Retrieves the specified movie
save	POST	/api/movies	Saves modifications to the movie

```
movieResource.query(function (data) {
   vm.movies = data;
});
```

```
movieResource.get({ movieId: $routeParams.movieId },
    function (data) {
        vm.movie = data;
});
```

$resource Methods (cont)

Method Name	Verb	URL	Description
query	GET	/api/movies	Retrieves all of the movies
get	GET	/api/movies/5	Retrieves the specified movie
save	POST	/api/movies	Saves modifications to the movie

```
vm.movie.$save(function (data) {
   vm.message = "Save successful.";
});
```

Post vs. Put

POST (api/movies)

- Posts data for a resource or set of resources
- Used to:
  - Create a new resource when the server assigns the Id
  - Update a set of resources
- Not idempotent

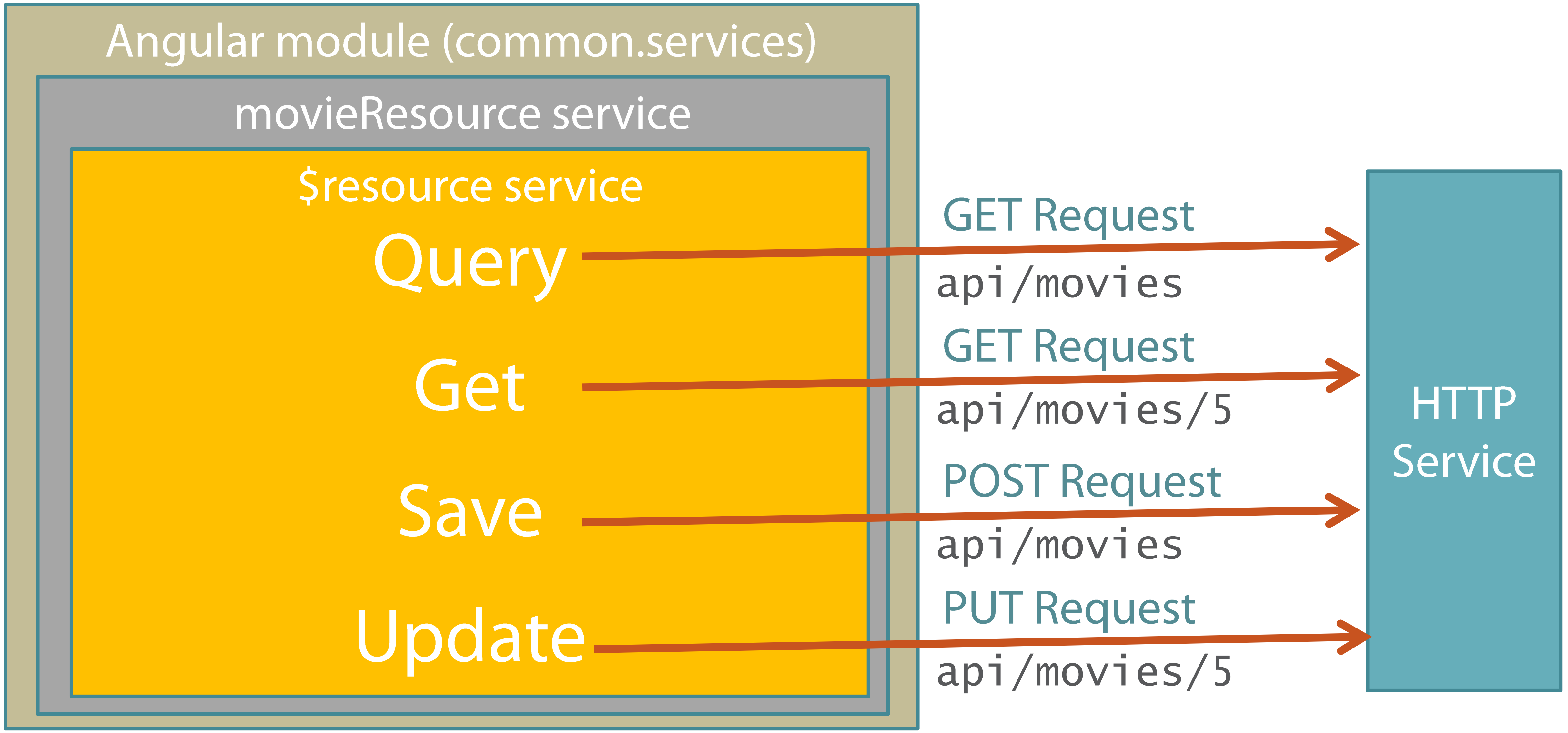
PUT (api/movies/5)

- Puts data for a specific resource with an Id
- Used to:
  - Create a new resource when the client assigns the Id
  - Update the resource with the Id
- Idempotent

```
function movieResource($resource) {
  return $resource("/api/movies/:movieId", null,
     {
        'update':{method:'PUT'}
     });
```

Creating a Custom $resource Method

$resource Methods



Demo: Retrieving Data

$httpBackend

- Angular's fake HTTP backend implementation
- Mocks the calls to the Web Service
- Returns static data to the application
- Two implementations:
  - ngMock: for unit testing applications
  - ngMockE2E: for end-to-end testing or backend-less development

Steps to Mocking the Web Server

Create a new module that depends on ngMockE2E

Add the new module as a dependency in the Main Module

Set up static data

Define the fake responses to the Web Service calls

Demo: Faking Data Retrieval

OData

- Open Data Protocol
- Standard way to access data using HTTP
- Protocol largely follows REST conventions
- And there is a large spec

OData Queries

- Allow construction of complex data queries
- Filter, shape, sort, and select data
- Examples:
  - `$filter: "contains(Title,'Rings')"`

  -
    ```
    $filter: "contains(Title,'Rings') and
              StarRating gt 4.5"
    ```

  -
    ```
    $filter: "contains(Title,'Rings') and
              StarRating ge 4.8 and StarRating le 5",
    $orderby: "StarRating desc"
    ```

OData Query Options

Option	Description	Example
$top	Returns the top n results	$top: 3
$skip	Skips n results	$skip: 1
$orderby	Sorts the results	$orderby: "StarRating desc"
$filter	Filters the results based on an expression	$filter: "StarRating gt 4.5"
$select	Selects the properties to include in the response	$select: "Title, MovieId"

http://www.odata.org/ Developers -> Documentation

Demo: OData Queries

Take Away

Building a reusable data access service

Faking data access

OData queries

Thank You!

- @deborahkurata
- deborahk@insteptech.com
- http://blogs.msmvps.com/deborahk
- https://github.com/DeborahK/ab2015