

Revisiting the Application



Antonio Goncalves

JAVA CHAMPION

@agoncal www.antoniogoncalves.org



Previous Module



Service

HTTP

Reactive programming

Swagger CodeGen

Parse JSON contract

Generate code

Angular front-end invokes Java EE back-end



Overview



Revisit application

Focus on changes we made

Java EE back-end

Angular front-end

External references



BookStore Application



Angular



Java EE



Java EE



Defining the Domain Model



Angular



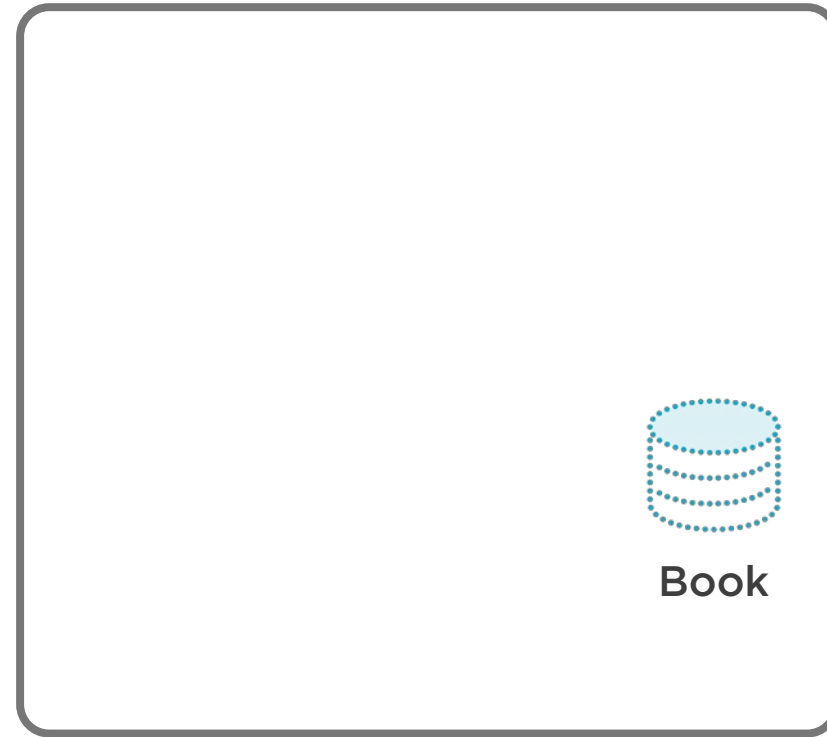
Java EE



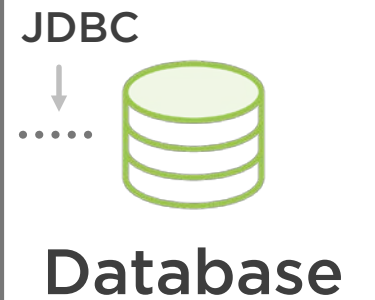
Defining the Domain Model



Angular



Java EE



Book Entity with JPA

@Entity

```
public class Book {
```

```
    @Id @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    private Long id;
```

```
    @Column(length = 200)
```

```
    private String title;
```

```
    @Column(length = 10000)
```

```
    private String description;
```

```
    @Column(name = "publication_date")
```

```
    @Temporal(TemporalType.DATE)
```

```
    private Date publicationDate;
```

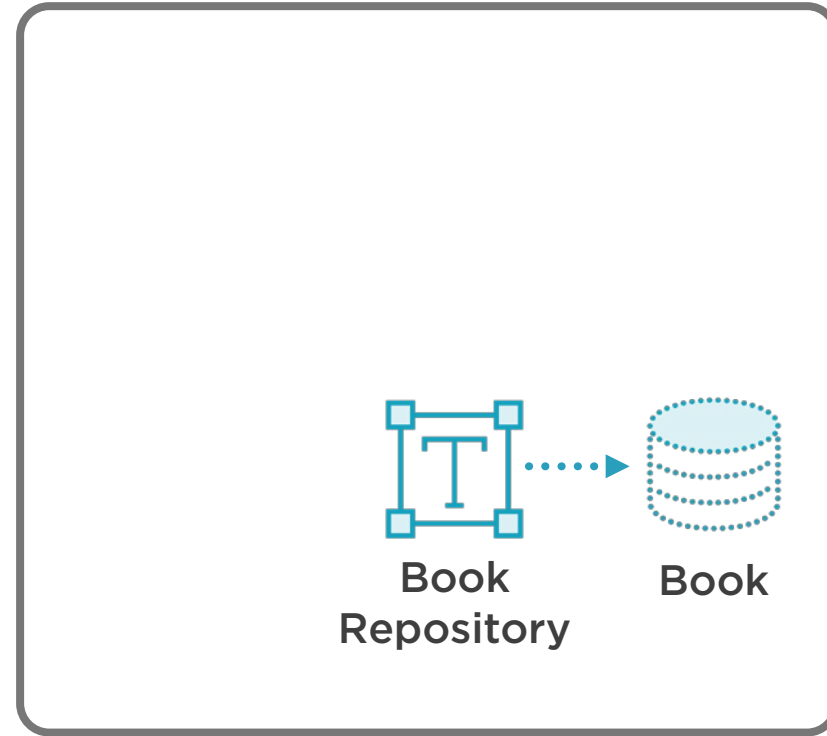
```
    // ...
```



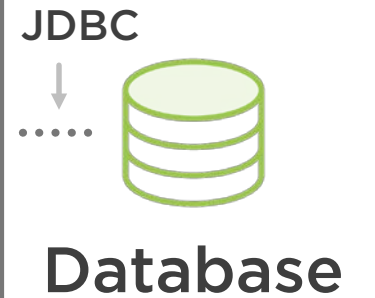
Adding a Transactional Repository



Angular



Java EE



Database



Book Repository with JPA

```
@Transactional(SUPPORTS)
public class BookRepository {

    @PersistenceContext(unitName = "bookStorePU")
    private EntityManager em;

    public Book find(Long id)    {
        return em.find(Book.class, id);
    }

    @Transactional(REQUIRED)
    public Book create(Book book) {
        em.persist(book);
        return book;
    }

    // ...
}
```



Book Repository with JPQL

```
// ...
```

```
public List<Book> findAll() {  
    TypedQuery<Book> query = em.createQuery(  
        "SELECT b FROM Book b ORDER BY b.title DESC", Book.class);  
    return query.getResultList();  
}
```

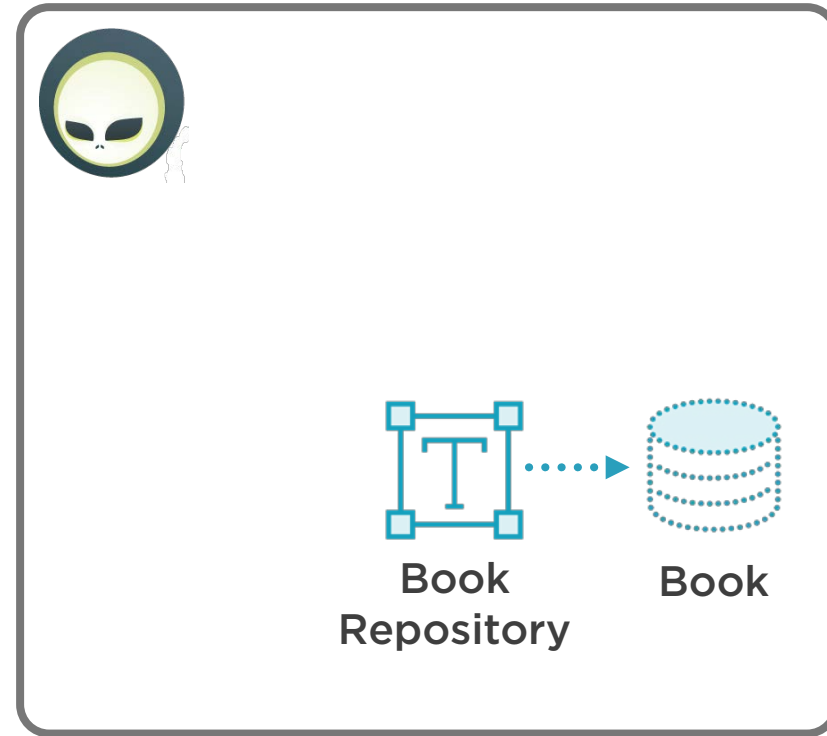
```
public Long countAll() {  
    TypedQuery<Long> query = em.createQuery(  
        "SELECT COUNT(b) FROM Book b", Long.class);  
    return query.getSingleResult();  
}
```



Testing the Java EE Application



Angular

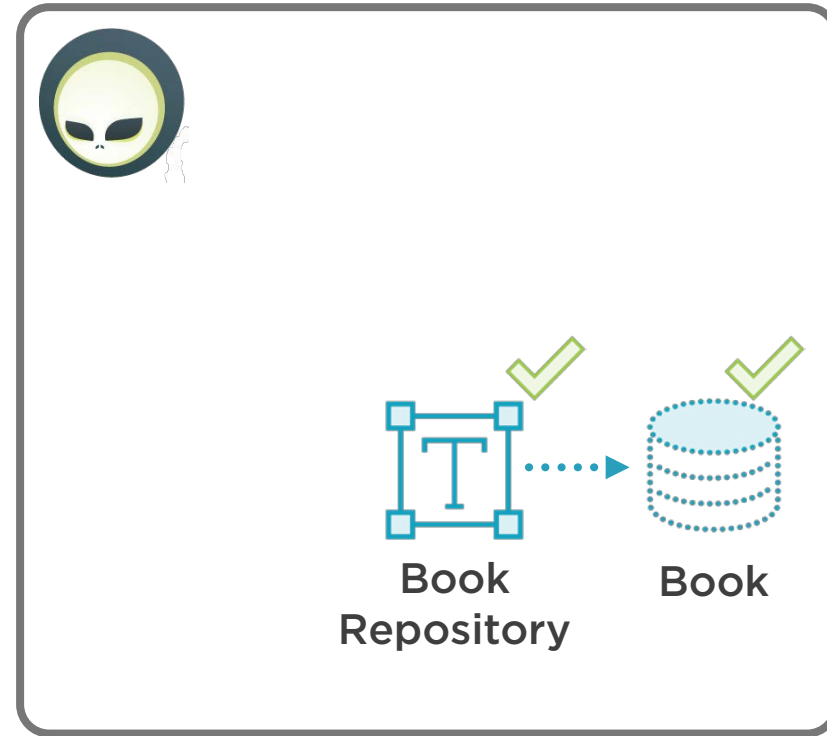


Java EE

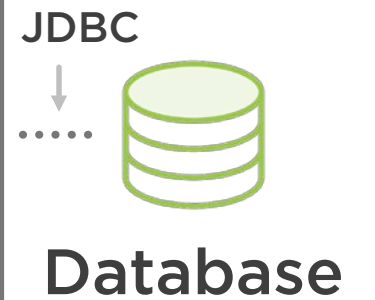
Validating Data



Angular



Java EE



Book Entity with Constraints

```
@Entity
public class Book {

    @NotNull @Size(min = 1, max = 200)
    private String title;

    @Size(min = 1, max = 10000)
    private String description;

    @NotNull @Min(1)
    private Float unitCost;

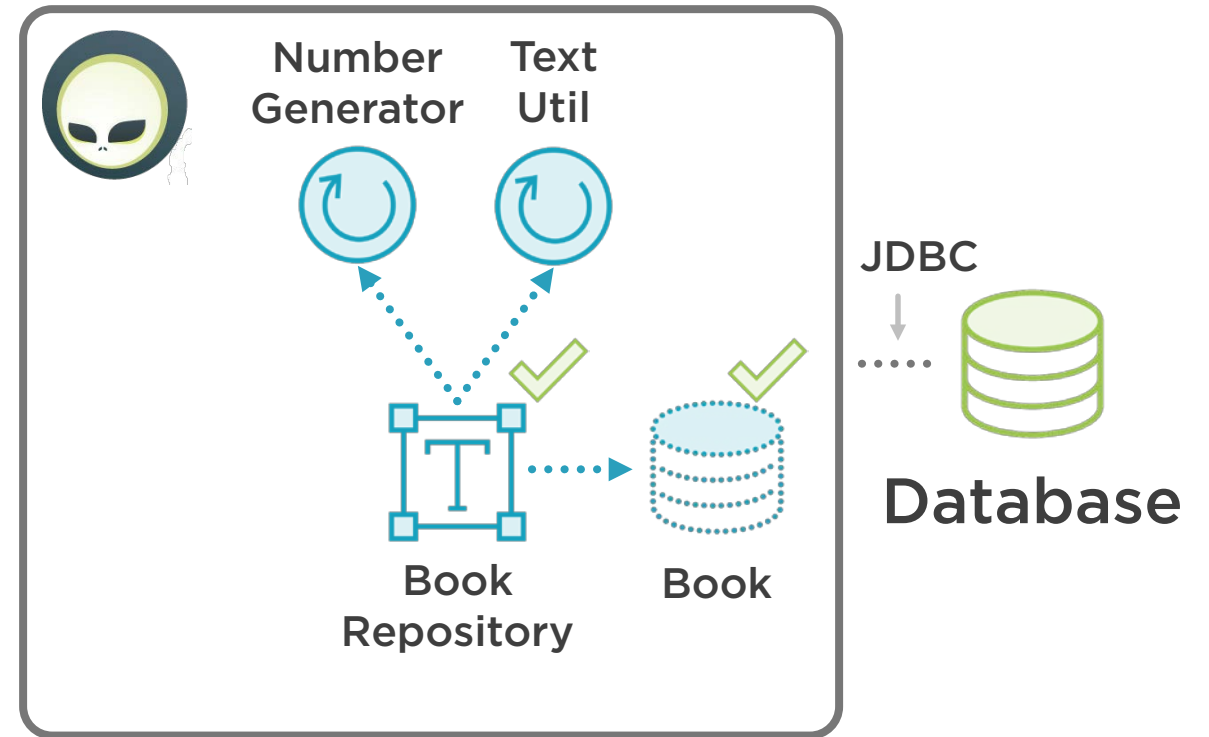
    @Past
    private Date publicationDate;
    // ...
}
```



Injecting Beans



Angular



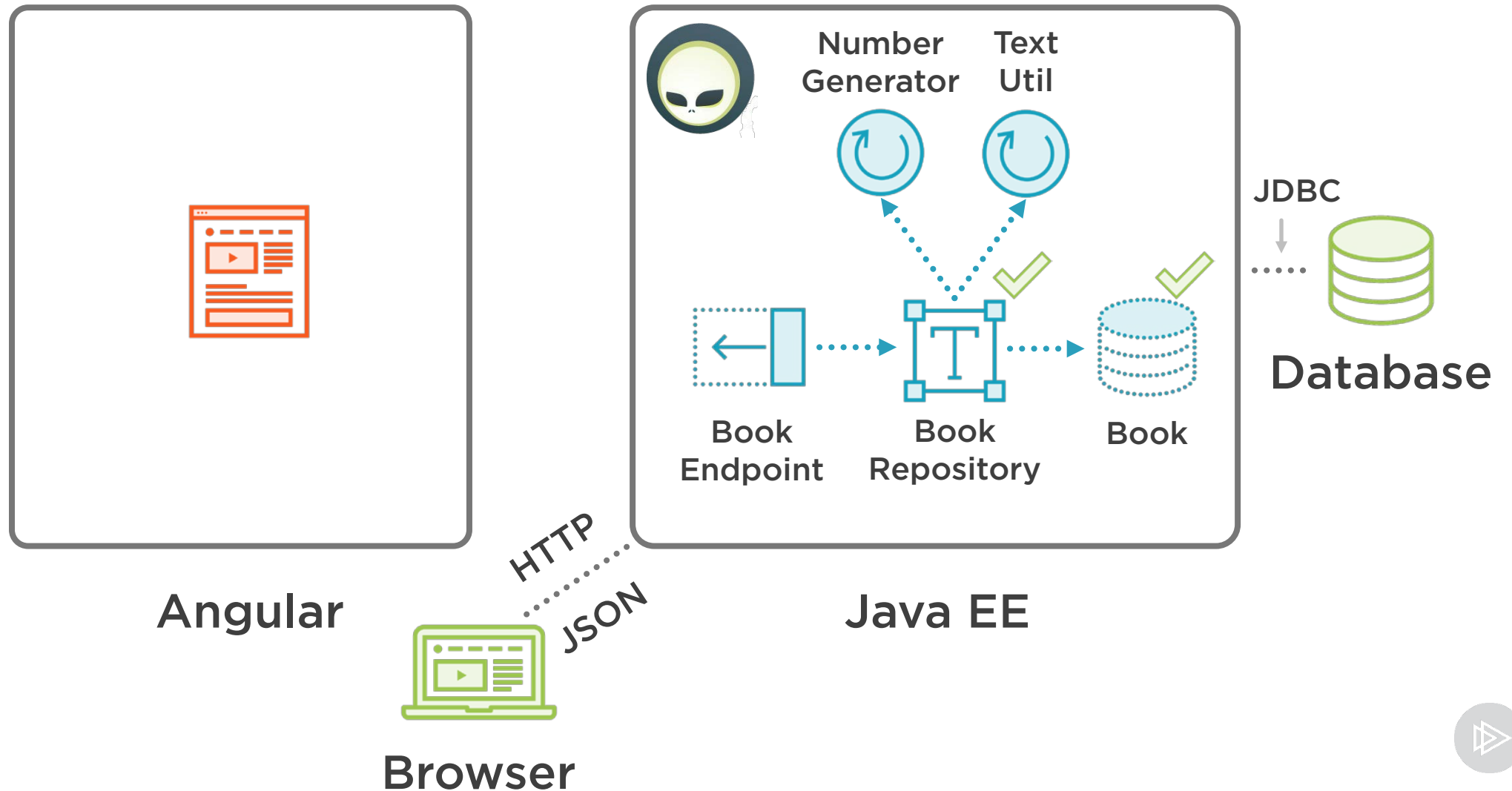
Java EE

Injecting Beans with CDI

```
public class BookRepository {  
  
    @Inject  
    private NumberGenerator generator;  
  
    @Inject  
    private TextUtil textUtil;  
  
    @Transactional(REQUIRED)  
    public Book create(@NotNull Book book) {  
        book.setIsbn(generator.generateNumber());  
        book.setTitle(textUtil.sanitize(book.getTitle()));  
        em.persist(book);  
        return book;  
    }  
}
```



Exposing a REST Service



Book Endpoint with JAX-RS

@Path("/books")

public class BookEndpoint {

GET <http://www.bookstore.com/books>

@Inject

private BookRepository bookRepository;

@GET

@Produces(APPLICATION_JSON)

public Response getBooks() {

 List<Book> books = bookRepository.findAll();

if (books.size() == 0)

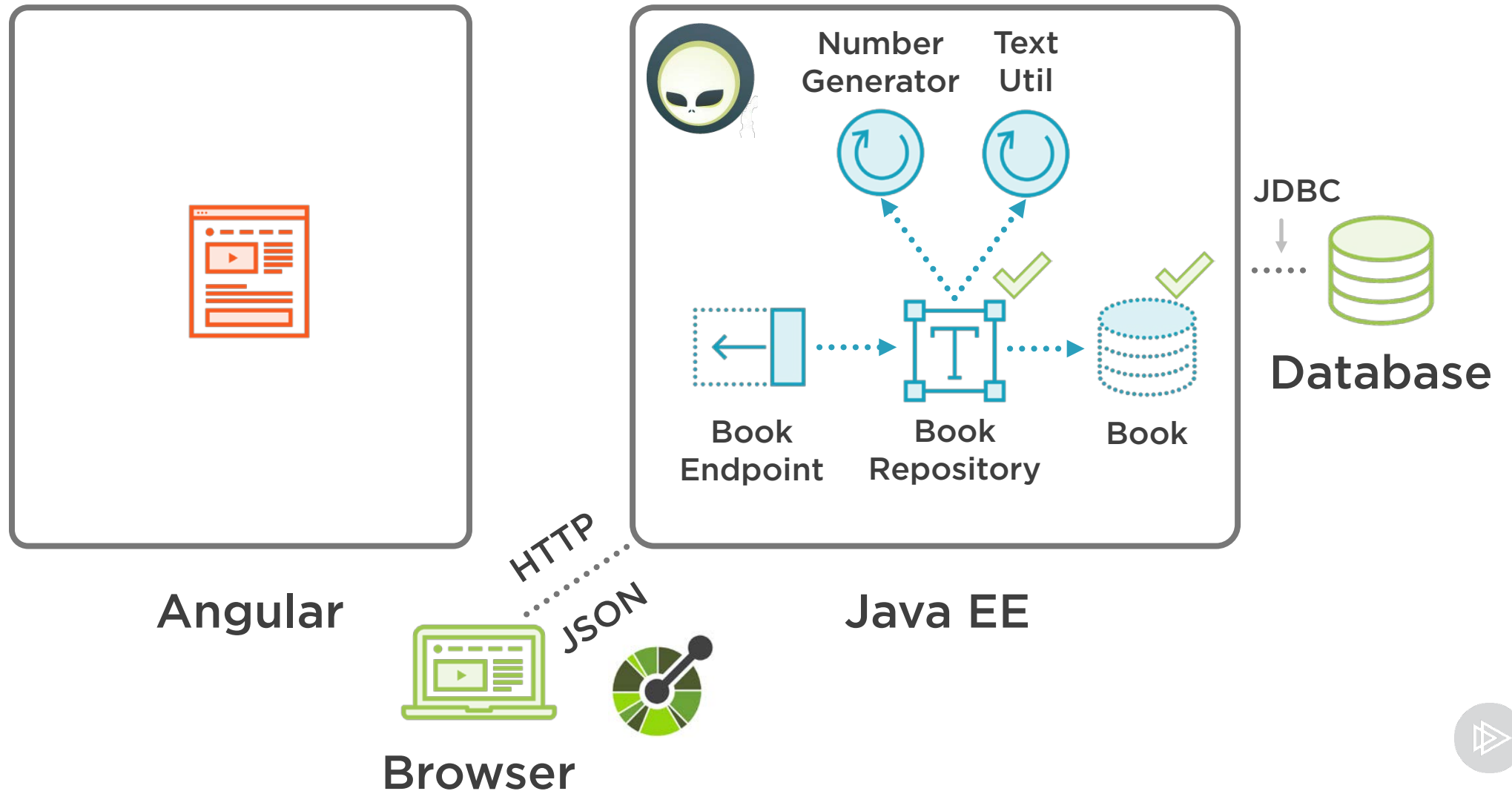
return Response.status(Response.Status.NO_CONTENT).build();

return Response.ok(books).build();

}



Documenting the REST Service



```

@SwaggerDefinition(
    info = @Info(description = "BookStore"),
    host = "localhost:8080",
    basePath = "/bookstore-back/api",
    schemes = {HTTP, HTTPS}
)
@Path("/books")
public class BookEndpoint {

    @GET @Path("/{id : \\d+}")
    @Produces(APPLICATION_JSON)
    @ApiOperation("Returns a book",
        response = Book.class)
    @ApiResponses({
        @ApiResponse(code = 404,
            message = "Not found"),
        @ApiResponse(code = 200,
            message = "Book found"),
    })
    public Response getBook(@PathParam("id")
        Long id) {
    }
}

```

```

{
    "swagger": "2.0",
    "info": {"description": "BookStore"},
    "host": "localhost:8080",
    "basePath": "/bookstore-back/api",
    "schemes": [ "http", "https"],
    "paths": {
        "/books/{id}": {
            "get": {
                "operationId": "getBook",
                "produces": ["application/json"],
                "parameters": [{ "name": "id",
                    "type": "integer",
                    "pattern": "\\d+"}],
                "summary": "Returns a book",
                "responses" : {
                    "404" : { "description" : "Not found"}
                    "200" : {
                        "description" : "Book found",
                        "schema": {
                            "$ref" : "#/definitions/Book"}},
            }
        }
    }
}

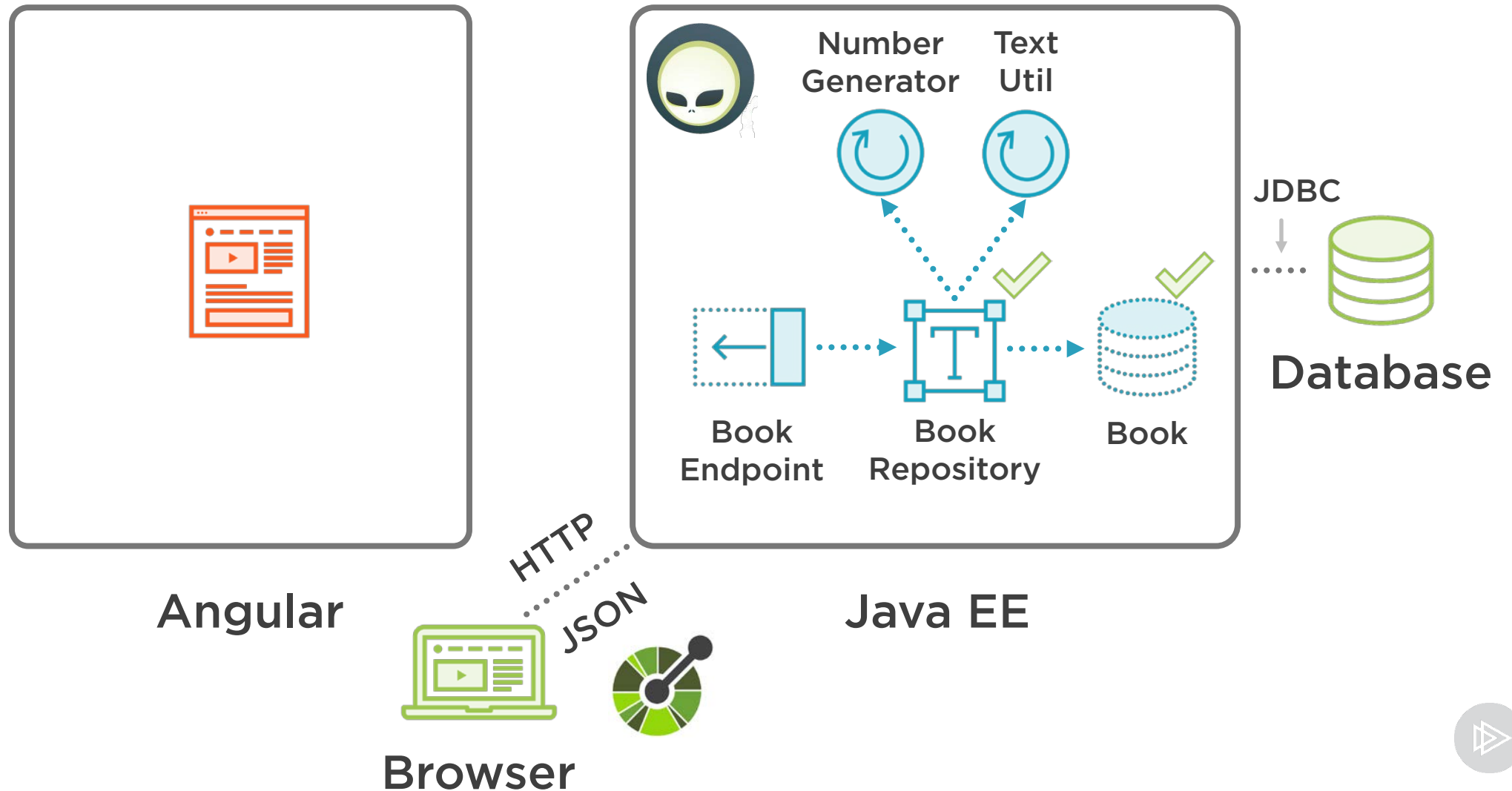
```



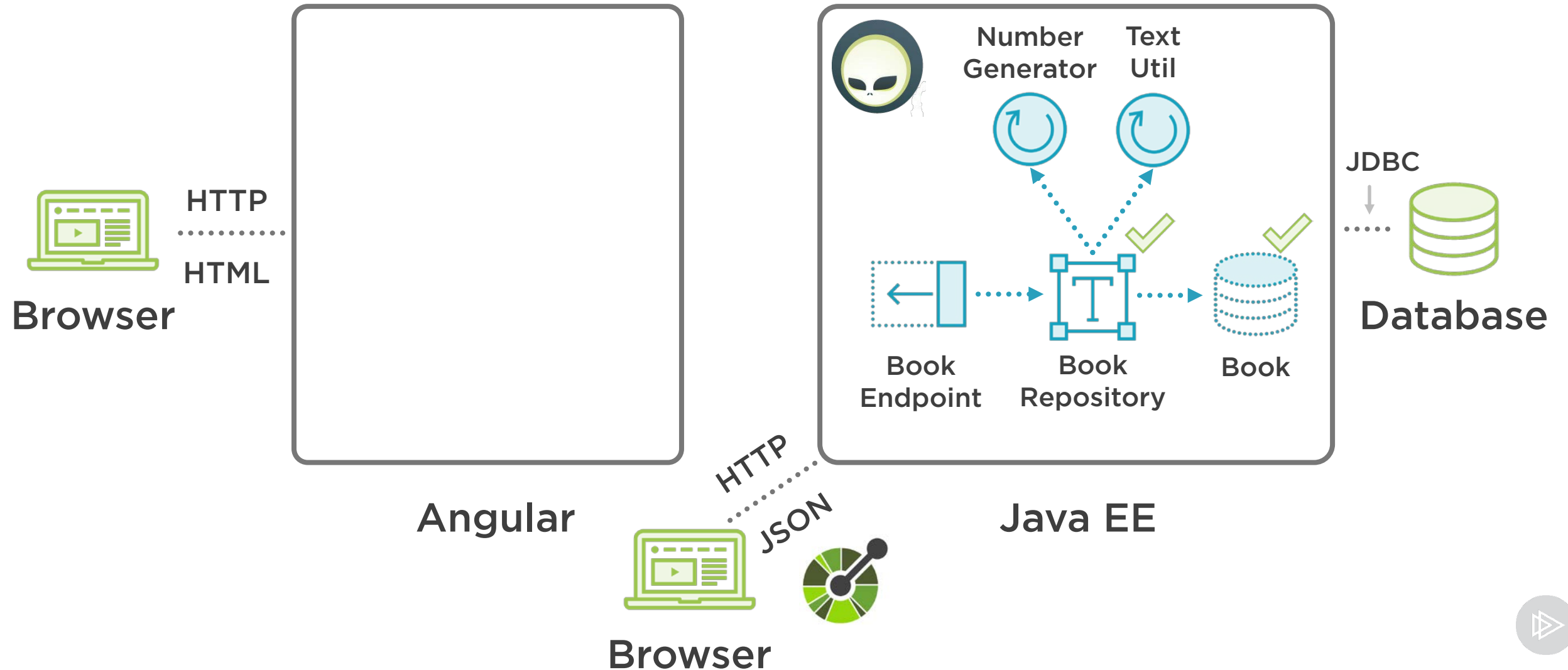
Angular



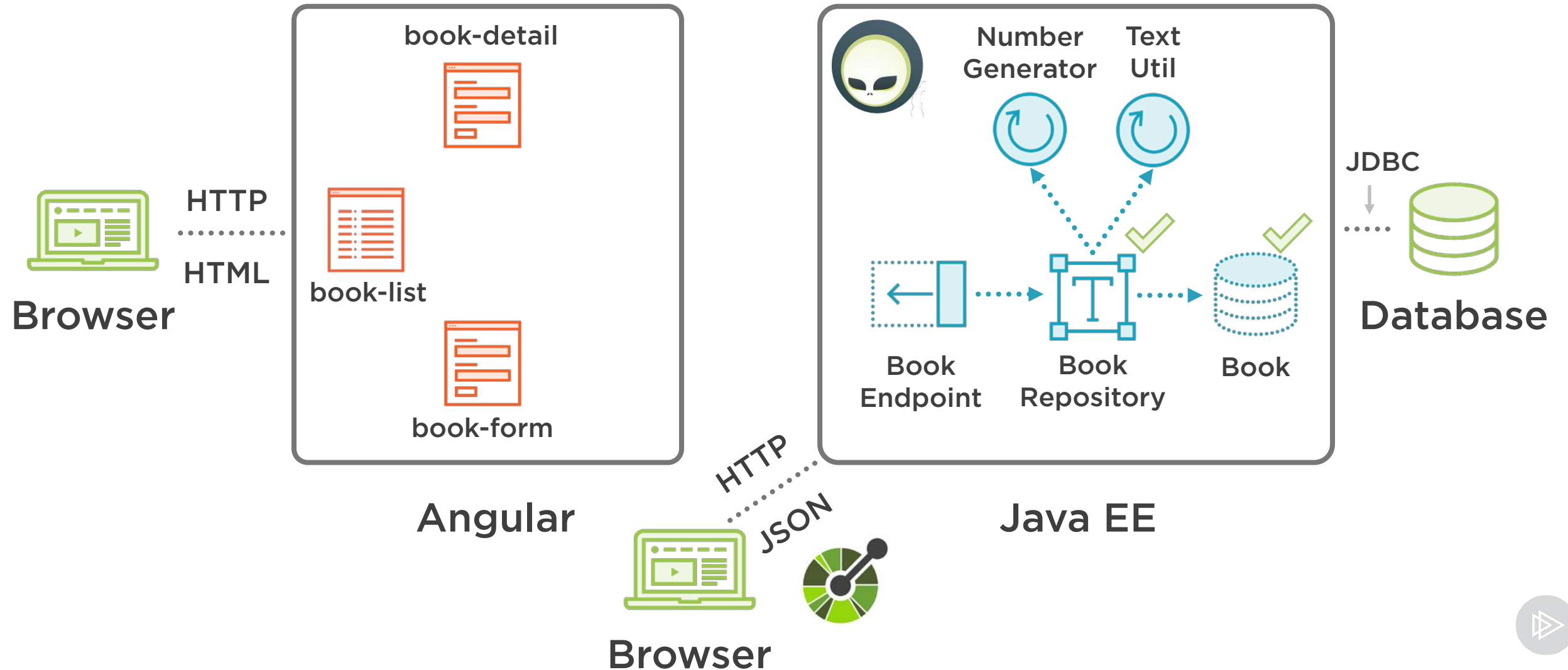
Designing the User Interface



Designing the User Interface



Designing the User Interface




```
<nav class="navbar">
  <a href="">{{title}}</a>
  <div class="collapse">
    <ul>
      <li class="nav-item">
        <a href="">List</a>
      </li>
      <li class="nav-item">
        <a href="">Create</a>
      </li>
    </ul>
  </div>
</nav>
```

app.component.html

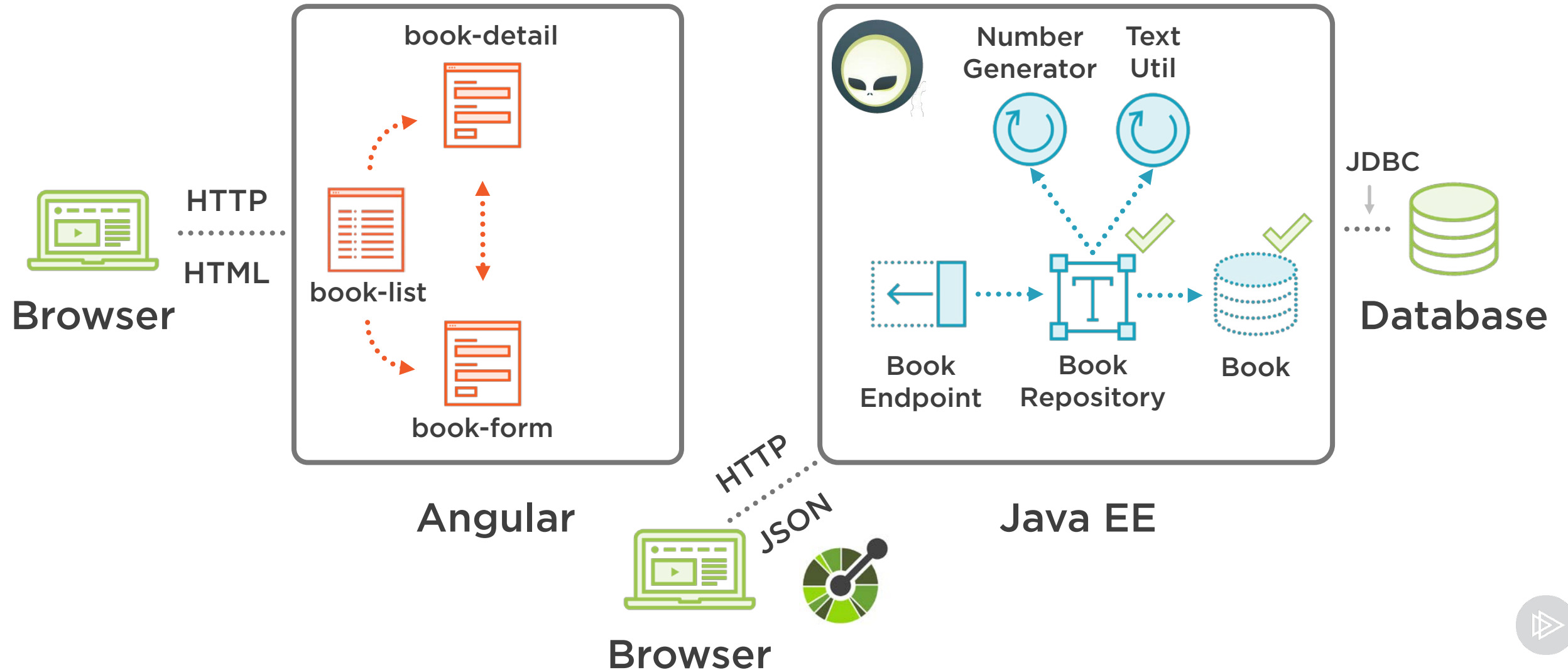
```
import { Component } from
    '@angular/core';

@Component({
  selector: 'bs-root',
  templateUrl:
    './app.component.html'
})
export class AppComponent {
  title = 'BookStore';
}
```

app.component.ts



Navigating Through Components



```
<a [routerLink]="[' /book-list']">
  {{title}}
</a>

<ul>
  <li>
    <a [routerLink]="[' /book-list']">
      List
    </a>
  </li>
  <li>
    <a [routerLink]="[' /book-form']">
      Create
    </a>
  </li>
</ul>
```

app.component.html

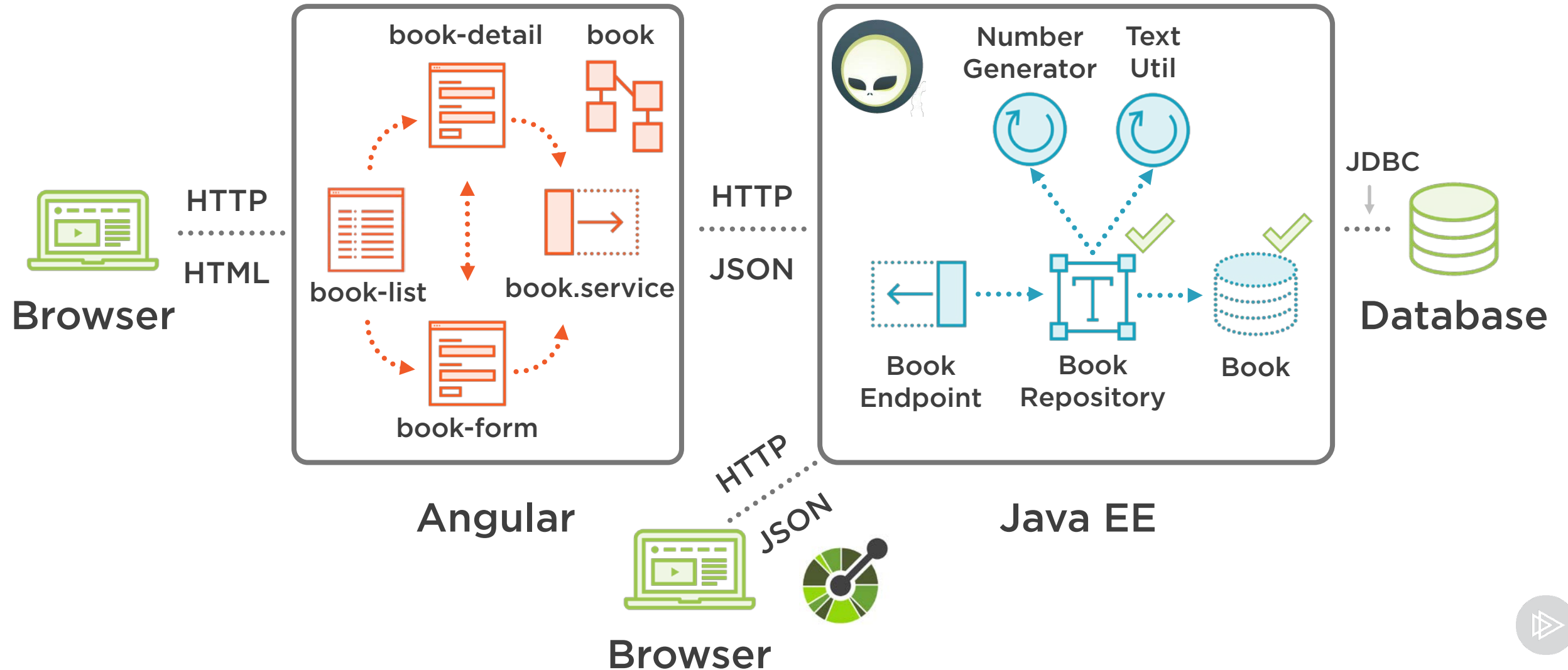
```
const routes: Routes = [
  {path: 'book-list', component:
    BookListComponent},
  {path: 'book-form', component:
    BookFormComponent}
];

@NgModule({
  imports:
    [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

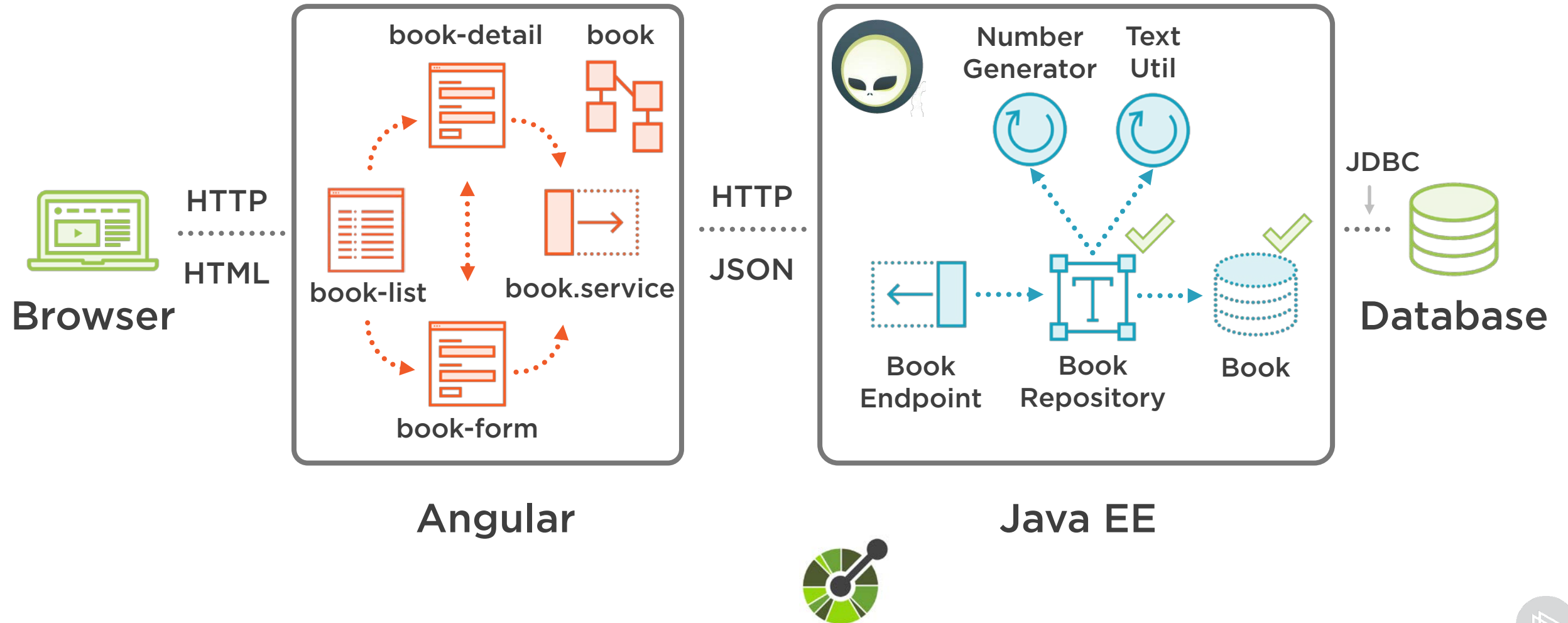
app-routing.module.ts



Invoking the REST Service



Invoking the REST Service



```

@Injectable()
export class BookService {
  protected basePath =
    'http://localhost:8080/bookstore-back/api';

  constructor(protected http: Http) { }

  public getBook(id: number):
    Observable<Book> {...}

  public countBooks(): Observable<number> { }

  public createBook(body: Book):
    Observable<{}> {...}

  public deleteBook(id: number):
    Observable<{}> {...}

  public getBooks():
    Observable<Array<Book>> {}
}

```

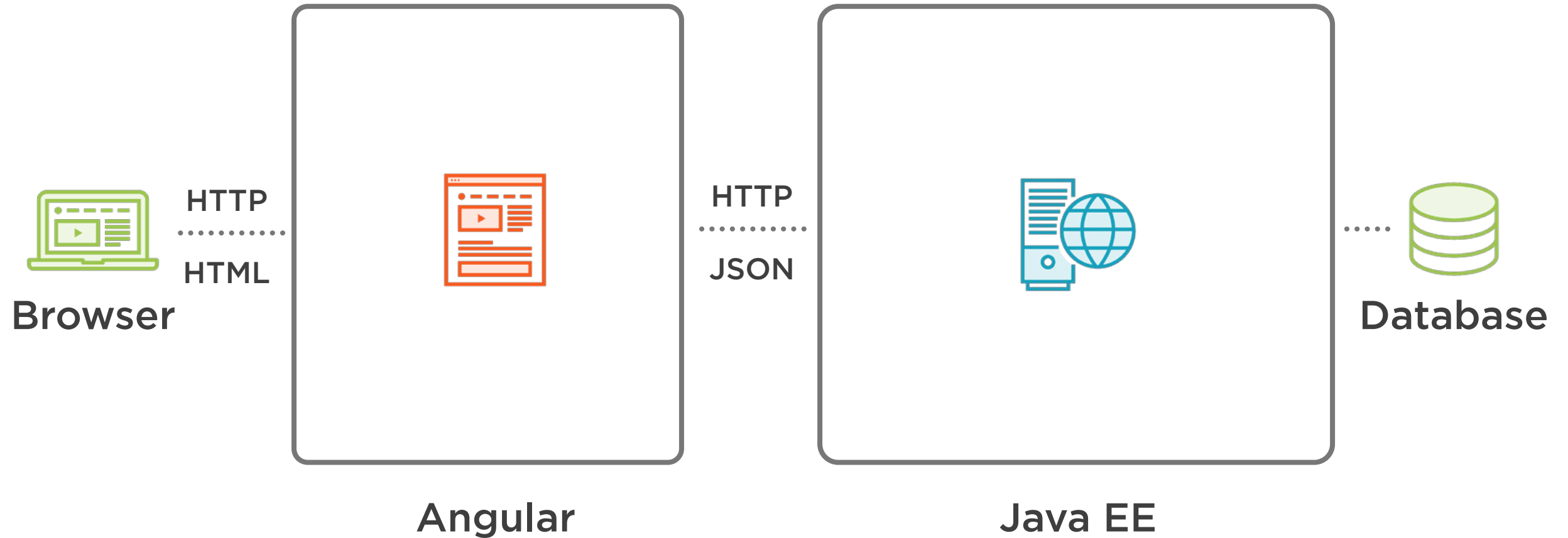
```

{
  "swagger": "2.0",
  "info": {"description": "BookStore"},
  "host": "localhost:8080",
  "basePath": "/bookstore-back/api",
  "schemes": [ "http", "https"],
  "paths": {
    "/books/{id}": {
      "get": {
        "operationId": "getBook",
        "produces": ["application/json"],
        "parameters": [{ "name": "id",
                          "type": "integer",
                          "pattern": "\\d+"}],
        "summary": "Returns a book",
        "responses": {
          "404": { "description": "Not found"},
          "200": {
            "description": "Book found",
            "schema": {
              "$ref": "#/definitions/Book"
            }
          }
        }
      }
    }
  }
}

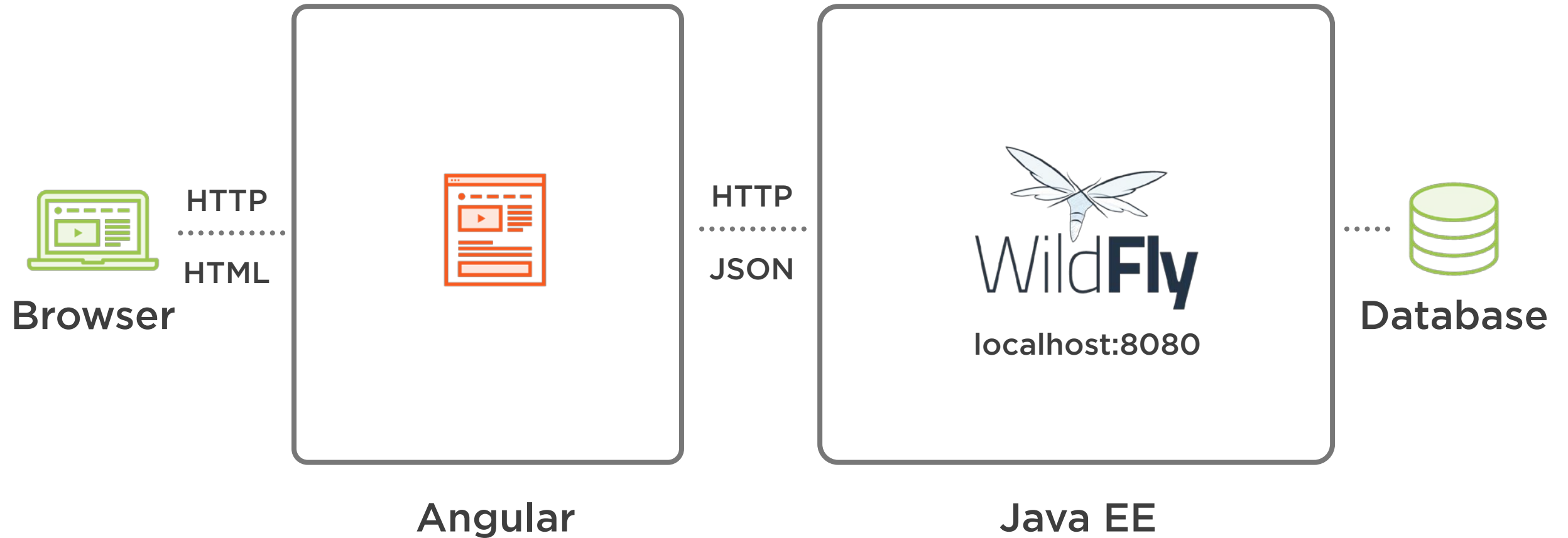
```



BookStore Application Deployment



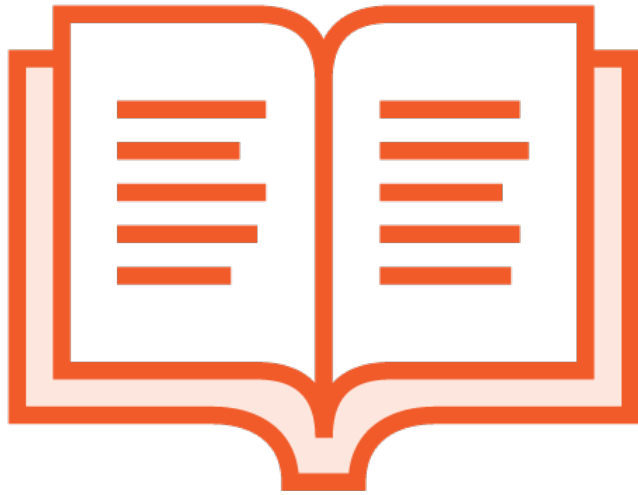
BookStore Application Deployment



BookStore Application Deployment



References



Articles

Technical blogs

Business analysis

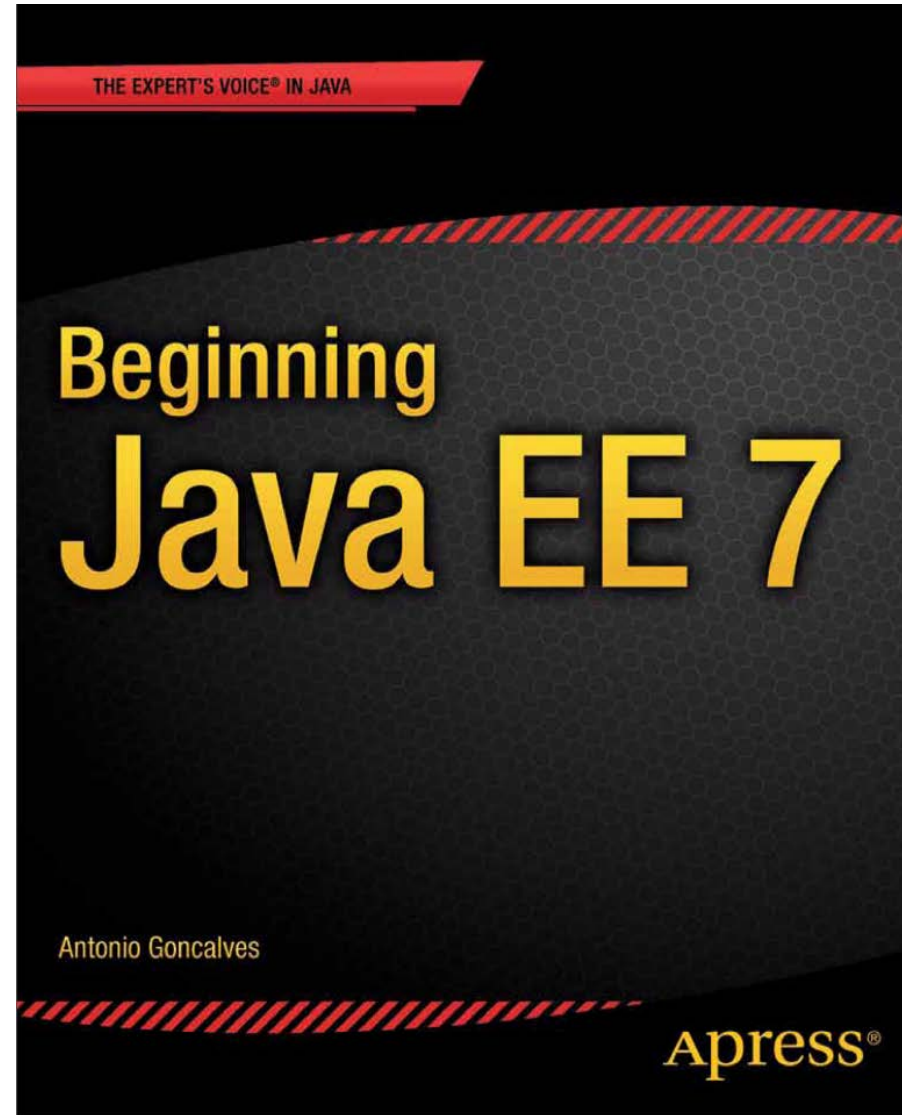
Java EE 7 specification

- <http://jcp.org/en/jsr/detail?id=342>

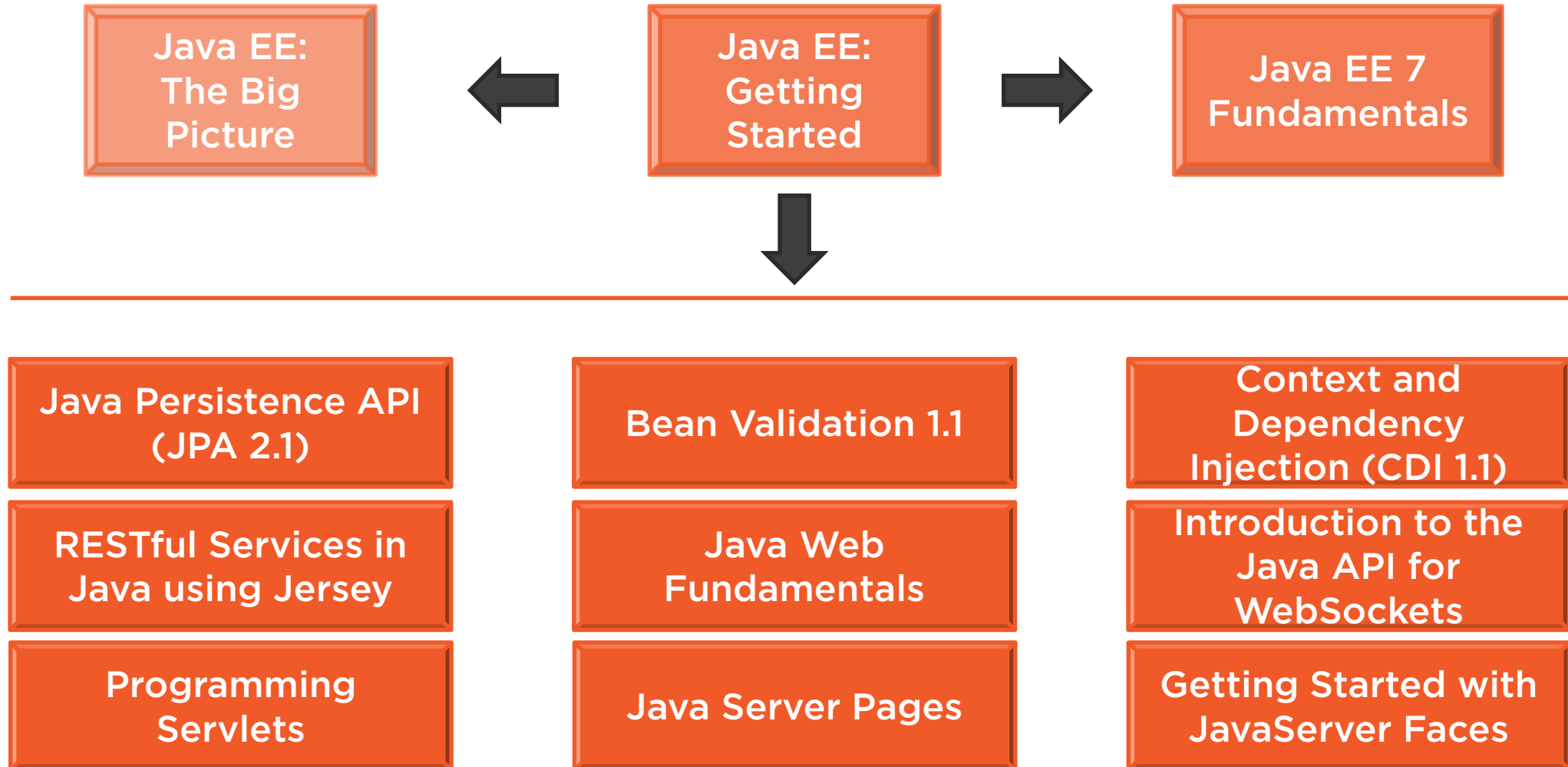
Angular tutorial

- <https://angular.io/docs/ts/latest/tutorial>

www.amazon.com/author/agoncal



Java EE Pluralsight Courses



FILTER BY

ROLES

SUBJECTS TO LEARN


TOOLS

SKILL LEVELS

AUTHORS

All Paths **Courses**

Courses: 5 results

Java Persistence API (JPA) 2.1 


by Antonio Goncalves

Intermediate

May 19, 2014

3h 41m

★★★★☆ (354)

Java EE 7 Fundamentals 



by Antonio Goncalves

Beginner

Aug 12, 2016

5h 23m

★★★★☆ (62)

 **Java EE: The Big Picture** 

by Antonio Goncalves

Beginner

Oct 28, 2015

1h 7m

★★★★☆ (278)

Context and Dependency Injection (CDI 1.1)


by Antonio Goncalves

Intermediate

Mar 24, 2015

3h 42m

★★★★☆ (125)



Bean Validation 1.1

by Antonio Goncalves

Intermediate

Jan 28, 2014

2h 29m

★★★★☆ (96)

Angular Pluralsight Courses

Angular 2:
Getting
Started



Angular 2
Fundamentals



Angular Routing

Angular 2 Forms

Angular 2: Reactive
Forms

Angular CLI

Angular 2
End-to-end

Angular Front to
Back with Web API



Search | Pluralsight

Antonio

← → ↺ 🏠

🔒 Sécurisé | <https://app.pluralsight.com/library/search?q=angular>

☆ ⋮

🔍 angular

Antonio

antonio.goncalves@gmail.com

⋮

🏠

🎮

📶

🔥

📡

📖

📅

🎤

📄

📋

🎤

🔍

FILTER BY

ROLES +

SUBJECTS TO LEARN +

TOOLS +

SKILL LEVELS +

AUTHORS +

All Paths **Courses**

Relevance ▾

Courses: 110 results

🔖

Angular 2: Getting Started

CC

by Deborah Kurata

Beginner

Oct 19, 2016

5h 26m

★★★★★ (1090)

⋮

🔖

Angular 2 Fundamentals

CC

by Jim Cooper and J...

Intermediate

May 09, 2017

9h 59m

★★★★★ (276)

⋮

🔖

Building a Web App with ASP.NET Core, MVC 6, EF Core, and Angu

CC

by Shawn Wildermuth

Beginner

Jul 21, 2016

9h 33m

★★★★★ (514)

⋮

🔖

Angular 2: First Look

CC

by John Papa

Intermediate

Nov 09, 2016

4h 31m

★★★★★ (913)

⋮

🔖

Angular Fundamentals

CC

⋮

Java EE: Getting Started



Antonio Goncalves

JAVA CHAMPION

@agoncal www.antoniogoncalves.org

