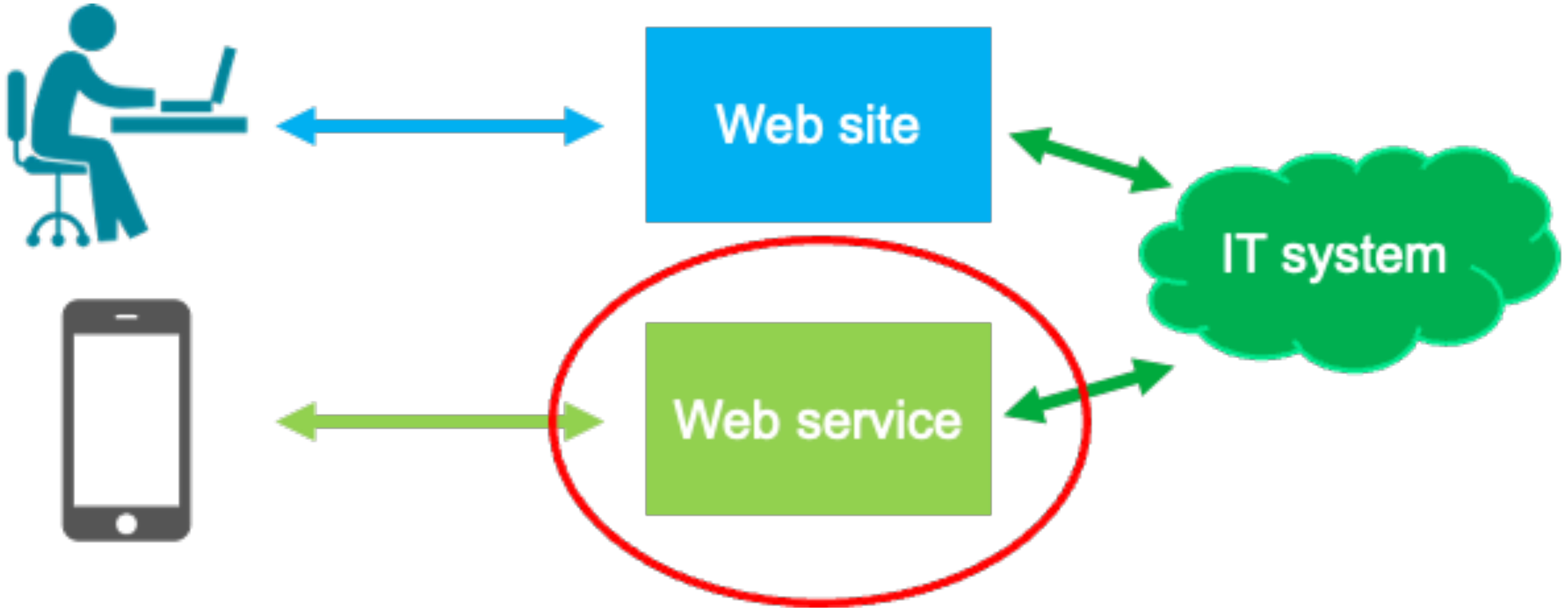
Creating REST Services

Creating REST Services



What Is a REST API?

- REST = Representational State Transfer
- A REST API exposes resources via HTTP
- Each resource has a URL (endpoint) and is accessed using HTTP methods
- Example resource: /books

Common HTTP Methods in REST APIs

- HTTP methods are like CRUD for databases:
    - POST - Create
    - GET - Read
    - PUT - Update
    - DELETE - Delete

URI - Uniform Resource Identifier

HTTP method	Example URI	Description
GET	/customer	Returns a list of all customers
GET	/customer/1	Returns one customer with the id 1
POST	/customer	Creates a new customer (Customer object in request body)
PUT	/customer/1	Updates a customer with the id 1 (Customer object in request body)
DELETE	/customer/1	Deletes the customer with the id 1

REST API Example Workflow

- Client sends a POST to /books to add a book
- Client sends a GET to /books/1 to read it
- Client sends a PUT to /books/1 to update it
- Client sends a DELETE to /books/1 to remove it

REST API with Spring Boot

```
@RestController
public class BookController {
    // define endpoints here
}
```

- What is @RestController?
  - A Spring annotation that marks a class as a REST controller
  - Combines @Controller + @ResponseBody
  - All methods return data (usually JSON), not HTML views

HTTP Method Annotations in Spring

HTTP method	Example URI	Annotation example
GET	/customer	@GetMapping("/customer")
GET	/customer/1	@GetMapping("/customer/{id}")
POST	/customer	@PostMapping("/customer")
PUT	/customer/1	@PutMapping("/customer/{id}")
DELETE	/customer/1	@DeleteMapping("/customer/{id}")

Demo 1 - REST Service

- A REST Service with a Controller and some Controller methods
- Integration Tests that can test the Controller methods

Exercise 1 - REST Services

- Download and open the CreatingRESTServicesStarter project
- Look at the integration tests, try to run them and they will all fail
- Look at the DogController class for tips about creating the 5 methods that will make the 5 Integration tests pass
- Use the correct method in the mapping annotation, use the repository as expected by the REST standard and return the expected values
- Use the @RequestBody Dog dog as input argument in the @PostMapping and the @PutMapping method to get the dog object directly into a Java object

Returning Objects from a REST API

```java
@GetMapping("/books")
public List<Book> getAllBooks() {
    return bookRepository.findAll();
}
```

- In early examples, controllers return:
  - A single object → e.g., Book
  - A list of objects → e.g., List<Book>
- Spring Boot automatically converts these objects to JSON (default with @RestController)

What’s Missing?

- No way to control:
    - HTTP status code (defaults to 200 OK)
    - Headers, such as Content-Type
    - Empty or error cases (e.g., item not found)
- Clients may need different responses in different scenarios:
    - 200 OK with data
    - 201 Created after POST
    - 404 Not Found if resource is missing
    - 204 No Content after deletion

Why Status Codes Matter

- Clients use status codes to understand what happened
- Examples:
    - GET /book/123 → returns 404 Not Found if the book doesn't exist
    - POST /book → should return 201 Created with a Location header
    - DELETE /book/123 → should return 204 No Content

Controlling the Response with ResponseEntity

```
@GetMapping("/books")
public ResponseEntity<List<Book>> getAllBooks() {
    List<Book> books = bookRepository.findAll();

    return ResponseEntity
            .status(HttpStatus.OK)
            .contentType(MediaType.APPLICATION_JSON)
            .body(books);
}
```

- ResponseEntity<T> gives full control over:
    - HTTP status code
    - Headers (like Content-Type)
    - Returned body

Demo 2 - Using ResponseEntity

- Change the REST API to start using ResponseEntity
- Be able to set status and content type in the response

Exercise 2 - REST Services

- Use the solution from Exercise 1
- Change all REST API methods to use ResponseEntity as the return type
- Use the ResponseEntity object to set the status, content type and body