Pontificia Universidad Católica del Ecuador
Seréis mis testigos
ESMERALDAS

CONTENT

Guide prepared by: JESÚS JOSÉ BONE CAICEDO

Student of IT at the Pontifical Catholic University of Ecuador - Esmeraldas Campus

1 CHAPTER 1 – FUNCTIONAL TESTS GENERATED BY CHATGPT

To minimize human intervention in test case generation using ChatGPT, a standard prompt was applied uniformly across all selected functional methods. The model (ChatGPT 5 Plus, web version, October 2025) received only the code snippet corresponding to the method under test, along with a fixed set of generic instructions for test generation. This strategy avoided biases associated with prompt design or the researcher's prior knowledge, thus ensuring the transparency and replicability of the process.

Additionally, to prevent contamination effects between interactions and ensure the model did not accumulate prior context, each test generation was performed in a new, independent chat. This prevented the AI from being influenced by previous responses or becoming overwhelmed with unnecessary information.

The prompts were designed with a strictly neutral approach, without including any hints that could favor either of the approaches being compared. Furthermore, during the validation stage, if an AI-generated test failed to execute correctly, it was not regenerated. The decision not to request new versions avoided introducing human corrections that could have guided the model regarding expected output or parameters. Regenerating the test would have provided indirect feedback to the AI, compromising the study's neutrality and the independence of the model's reasoning. **It should be noted that these prompts were mainly created in Spanish but were translated into English for this guide.**

1.1 CLASS: OwnerController METHOD: showOwner()

Prompt used

Create a functional test case in JUnit 5 for the following Spring Boot controller method.

Use MockMvc to make an HTTP request and verify the endpoint's response.

The test should:

- Run the endpoint **GET /owners/{ownerId}**.
- Validate the corresponding HTTP status code according to the flow defined in the method.
- Verify that the returned view is "owners/ownerDetails".
- Check that the "owner" attribute is present in the model if applicable.
- Include the necessary imports and explanatory comments.
- Use the annotations **@SpringBootTest** and **@AutoConfigureMockMvc**.

Here's the method to try:

```
@GetMapping("/owners/{ownerId}")
public ModelAndView showOwner(@PathVariable("ownerId") int ownerId) {
    ModelAndView mav = new ModelAndView("owners/ownerDetails");
    Optional<Owner> optionalOwner = this.owners.findById(ownerId);
    Owner owner = optionalOwner.orElseThrow(() -> new IllegalArgumentException(
            "Owner not found with id: " + ownerId + ". Please ensure the ID is correct "));
    mav.addObject(owner);
    return mav;
}
```

Code generated for this method

File OwnerControllerShowOwnerTestIA.java (check in folder)

PATH: "experimental\ia\funcionalesCHATGPT\OwnerControllerShowOwnerTestIA.java"

Result obtained in the test

```
[INFO]
[INFO] Results:
[INFO]
[ERROR] Errors:
[ERROR]   OwnerControllerShowOwnerTestIA.shouldReturnClientErrorWhenOwnerNotFound:65 » Servlet Request processing failed: java.lang.IllegalArgumentException: Owner not found with id: 99. Please ensure the ID is correct and the owner exists in the database.
[INFO]
[ERROR] Tests run: 2, Failures: 0, Errors: 1, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  18.958 s
[INFO] Finished at: 2025-10-27T14:51:23-05:00
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:3.2.5:test (default-test) on project spring-petclinic:
[ERROR]
[ERROR] Please refer to C:\Users\doleh\Downloads\development\spring-petclinic\target\surefire-reports for the individual test results.
[ERROR] Please refer to dump files (if any exist) [date].dump, [date]-jvmRun[N].dump and [date].dumpstream.
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
```

Comentado [JB1]: In this AI-generated functional test, the method's behavior was verified when querying both an existing owner and one not registered in the database. The successful scenario performed as expected, returning the corresponding view and loading the Owner object into the model. However, the unsuccessful scenario resulted in a runtime error because the IllegalArgumentException thrown by the System Under Test (SUT) is not handled by an error management mechanism, resulting in a 500 error instead of the expected 4xx code. It is important to note that no changes were implemented in the SUT to catch this exception, in order to maintain the study's neutrality and avoid modifying the system's actual behavior to favor the AI-generated tests.

1.2 CLASS: PetController METHOD: processCreationForm()

Prompt used

Create a functional test case in JUnit 5 for the following Spring Boot controller method.
Use MockMvc to make an HTTP request and verify the endpoint's response.

The test must:

- Run the endpoint **POST /owners/{ownerId}/pets/new**.
- Validate the corresponding HTTP status code according to the flow defined in the method.
- Verify that, if successful, it correctly redirects to "redirect:/owners/{ownerId}".
- Verify that, in case of a validation error, the view "pets/createOrUpdatePetForm" is returned.
- Include the necessary imports and explanatory comments.
- Use the annotations **@SpringBootTest** and **@AutoConfigureMockMvc**.

Here's the method to try:

```
@PostMapping("/pets/new")
public String processCreationForm(Owner owner, @Valid Pet pet, BindingResult result,
        RedirectAttributes redirectAttributes) {


    if (StringUtils.hasText(pet.getName()) && pet.isNew() && owner.getPet(pet.getName(), true)
!= null)
        result.rejectValue("name", "duplicate", "already exists");


    LocalDate currentDate = LocalDate.now();
    if (pet.getBirthDate() != null && pet.getBirthDate().isAfter(currentDate)) {
        result.rejectValue("birthDate", "typeMismatch.birthDate");
    }


    if (result.hasErrors()) {
        return VIEWS_PETS_CREATE_OR_UPDATE_FORM;
    }


    owner.addPet(pet);
    this.owners.save(owner);
    redirectAttributes.addFlashAttribute("message", "New Pet has been Added");
    return "redirect:/owners/{ownerId}";
}
```

Code generated for this method

File PetControllerProcessCreationFormTestIA.java (check in folder)

PATH: "experimental\ia\funcionalesCHATGPT\PetControllerProcessCreationFormTestIA.java"

Result obtained in the test

```
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 7.744 s -- in
org.springframework.samples.petclinic.owner.PetControllerProcessCreationFormTestIA

[INFO]

[INFO] Results:

[INFO]

[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0

[INFO]

[INFO]

[INFO] --- jacoco:0.8.13:report (report) @ spring-petclinic ---

[INFO] Loading execution data file C:\Users\doleh\Downloads\development\spring-petclinic\target\jacoco.exec

[INFO] Analyzed bundle 'petclinic' with 22 classes

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time:  16.922 s

[INFO] Finished at: 2025-10-27T15:02:15-05:00

[INFO] ------------------------------------------------------------------------
```

Comentado [JB2]: The AI-generated functional tests for this method executed successfully, validating both the successful pet creation flow and the proper handling of validation errors, particularly in the case of an invalid birthdate. The observed behavior matches the logic established in the System Under Test (SUT), returning the form view when errors occur and correctly redirecting the owner when the operation is successful. It is important to note that no modifications were made to the SUT, maintaining the neutrality of the experiment and avoiding any adjustments that could favor the AI-generated test cases.

1.3 CLASS: VisitController METHOD: processNewVisitForm()

Prompt used

Create a functional test case in JUnit 5 for the following Spring Boot controller method.
Use MockMvc to make an HTTP request and verify the endpoint's response.

The test must:

- Run the endpoint **POST /owners/{ownerId}/pets/{petId}/visits/new**.
- Validate the corresponding HTTP status code according to the flow defined in the method.
- Verify that, if successful, it correctly redirects to "redirect:/owners/{ownerId}".
- Verify that, in case of a validation error, the view "pets/createOrUpdateVisitForm" is returned.
- Include the necessary imports and explanatory comments.
- Use the annotations **@SpringBootTest** and **@AutoConfigureMockMvc**.

Here's the method to try:

```
@PostMapping("/owners/{ownerId}/pets/{petId}/visits/new")
    public String processNewVisitForm(@ModelAttribute Owner owner, @PathVariable int petId,
@Valid Visit visit,
            BindingResult result, RedirectAttributes redirectAttributes) {
        if (result.hasErrors()) {
            return "pets/createOrUpdateVisitForm";
        }

        owner.addVisit(petId, visit);
        this.owners.save(owner);
        redirectAttributes.addFlashAttribute("message", "Your visit has been booked");
        return "redirect:/owners/{ownerId}";
    }
```

Code generated for this method

File VisitControllerProcessNewVisitFormTestIA.java (check in folder)

PATH: "experimental\ia\funcionalesCHATGPT\VisitControllerProcessNewVisitFormTestIA.java"

Result obtained in the test

```
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 7.728 s -- in
org.springframework.samples.petclinic.owner.VisitControllerProcessNewVisitFormTestIA

[INFO]

[INFO] Results:

[INFO]

[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0

[INFO]

[INFO]

[INFO] --- jacoco:0.8.13:report (report) @ spring-petclinic ---

[INFO] Loading execution data file C:\Users\doleh\Downloads\development\spring-petclinic\target\jacoco.exec

[INFO] Analyzed bundle 'petclinic' with 22 classes

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time:  17.153 s

[INFO] Finished at: 2025-10-27T15:17:01-05:00

[INFO] ------------------------------------------------------------------------
```

Comentado [JB3]: The AI-generated functional tests for this method were successfully executed, adequately verifying both the correct visit registration flow and the return to the form in the event of validation errors. In the valid scenario, the system redirects to the owner's details and displays an informational message, while in the invalid scenario, it remains on the form view without attempting to persist the information. This indicates that the actual behavior of the SUT (System Under Test) meets the expected functional logic without requiring modifications to the existing code, thus ensuring the neutrality of the study and avoiding adjustments that might favor the AI-generated test cases.

2 CHAPTER 2 – UNIT TESTS GENERATED BY DIFFBLUE COVER

In the case of unit tests, these were generated using Diffblue Cover (14-day free trial version), which offers features equivalent to the premium version during the trial period. Unlike prompt-based models, Diffblue Cover does not require textual instructions to generate tests; instead, it analyzes the code at the bytecode level and automatically makes decisions about which test scenarios to build.

This approach allowed for a significantly simpler process, as the entire test generation was accomplished by executing a single command. Furthermore, by eliminating human intervention in the design or tuning of instructions, methodological neutrality was maintained and the introduction of bias during unit test generation was avoided.

2.1 Unit test – Owner class, addPet() method

Generation command

dcover create org.springframework.samples.petclinic.owner.Owner.addPet

Code generated for this method

File OwnerDiffblueTestAddPet.java (check in folder)

PATH: "experimental\ia\unitariasDIFFBLUECOVER\addPet\OwnerDiffblueTestAddPet.java"

Result obtained in the test

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running org.springframework.samples.petclinic.owner.OwnerAddPetDiffblueTest
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.074 s -- in
org.springframework.samples.petclinic.owner.OwnerAddPetDiffblueTest
[INFO]
[INFO] Results:
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jacoco:0.8.13:report (report) @ spring-petclinic ---
[INFO] Loading execution data file C:\Users\doleh\Downloads\development\spring-petclinic\target\jacoco.exec
[INFO] Analyzed bundle 'petclinic' with 22 classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  9.273 s
[INFO] Finished at: 2025-10-27T16:03:29-05:00
[INFO] ------------------------------------------------------------------------
```

Comentado [JB4]: The unit tests automatically generated by Diffblue Cover for the `addPet()` method successfully validated two relevant scenarios of the Owner model's behavior. In the first case, it was verified that when the pet does not have an assigned identifier, the object is added to the owner's pet list, reflecting an add operation in memory. In contrast, when the pet already has an ID, it was verified that the method does not add it to the list, leaving the collection unchanged. The results were successful in both cases and align with the logic implemented in the System Under Test (SUT) without requiring modifications, guaranteeing the neutrality of the experiment and demonstrating that the AI-generated tests are capable of accurately representing the model's internal behavior.

2.2 Unit test – Owner class, getPet(String name, boolean ignoreNew) method

Generation command

dcover create org.springframework.samples.petclinic.owner.Owner.getPet

Code generated for this method

File OwnerDiffblueTestGetPet.java (check in folder)

PATH: " experimental\ia\unitariasDIFFBLUECOVER\getPet\OwnerDiffblueTestGetPet.java"

Result obtained in the test

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running org.springframework.samples.petclinic.owner.OwnerGetPetDiffblueTest
[INFO] Tests run: 10, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.089 s -- in
org.springframework.samples.petclinic.owner.OwnerGetPetDiffblueTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 10, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jacoco:0.8.13:report (report) @ spring-petclinic ---
[INFO] Loading execution data file C:\Users\doleh\Downloads\development\spring-petclinic\target\jacoco.exec
[INFO] Analyzed bundle 'petclinic' with 22 classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  9.460 s
[INFO] Finished at: 2025-10-27T17:13:48-05:00
[INFO] ------------------------------------------------------------------------
```

Comentado [JB5]: The unit tests generated by Diffblue Cover for the different getPet() methods successfully evaluated multiple scenarios based on both name and identifier. The results demonstrated that, as expected, null is returned when the owner has no associated pets or when the searched name or ID does not match. It was also validated that when adding a pet without an ID (considered new), it is only returned if the ignoreNew parameter is false, adhering to the system's logic for distinguishing between existing and new pets. All executed cases were successful, confirming that the model's internal behavior is consistent with its implemented rules, without requiring modifications to the System Under Test (SUT) and maintaining the experiment's impartiality with respect to the AI-generated tests.

2.3 Unit test – PetValidator class, validate() method

Generation command

dcover create org.springframework.samples.petclinic.owner.PetValidator.validate

Code generated for this method

File PetValidatorDiffblueTestValidate.java (check in folder)

PATH: “experimental\ia\unitariasDIFFBLUECOVER\validate\PetValidatorDiffblueTestValidate.java"

Result obtained in the test

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running org.springframework.samples.petclinic.owner.PetValidatorDiffblueTest
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.326 s -- in org.springframework.samples.petclinic.owner.PetValidatorDiffblueTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jacoco:0.8.13:report (report) @ spring-petclinic ---
[INFO] Loading execution data file C:\Users\doleh\Downloads\development\spring-petclinic\target\jacoco.exec
[INFO] Analyzed bundle 'petclinic' with 22 classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  10.026 s
[INFO] Finished at: 2025-10-27T17:18:14-05:00
[INFO] ------------------------------------------------------------------------
```

Comentado [JB6]: The unit tests generated by Diffblue Cover successfully evaluated various validation scenarios in the PetValidator's validate() method. It was verified that when the pet contains the required information (name, type, and date of birth), no validation errors are generated. Conversely, when one of these fields is missing or empty, the validator logs the corresponding errors for the affected attribute, adhering to the rules defined in the System Under Test (SUT) to ensure data integrity before registration. The results were successful in all cases, confirming that the validation mechanism is correctly implemented and reinforcing the study's neutrality, as no modifications were made to the source code to facilitate the execution of the AI-generated tests.

3 CHAPTER 3 – FUNCTIONAL TESTS PERFORMED BY THE RESEARCHER

In this case, I was responsible for running the manual tests as the researcher. It's important to clarify that I'm not an expert in unit test development, and my previous experience is limited to sporadic exercises in other languages like JavaScript. However, to ensure a systematic and unbiased process, I followed this procedure:

- **Initial functional exploration:**

  First, I manually evaluated the endpoints corresponding to each method, identifying which parameters they accepted, which they should not accept, the expected HTTP responses (200, 400, 500, etc.), and, in successful cases, the structure of the response body. This stage allowed me to empirically understand the actual behavior of the system.

- **Translation of findings into test code:**

  After completing the manual exploration, I proceeded to implement the tests in code. To do this, I reviewed documentation and examples, using only the behavior I had previously validated in Postman as a reference. My goal was to faithfully reflect that behavior in the automated tests, without incorporating elements unrelated to direct observation.

- **Neutrality and control over AI assistance:**

  To maintain methodological neutrality, I did not reuse code snippets previously generated by ChatGPT or Diffblue Cover. The test logic was based strictly on the results of the manual exploration phase. During implementation, and given my level of experience, I used artificial intelligence only to resolve specific syntactic questions; however, I did not allow the AI to rewrite or adjust the tests according to its own criteria, thus preventing it from influencing the logical or structural decisions of the code.

- **Independence from AI-generated results:**

  Finally, when creating the manual tests, I didn't consider whether the AI had included certain methods in its own test suites. My goal wasn't to "fill in" or "fix" the AI's output, but rather to build the tests solely from the initial analysis performed with Postman, thus maintaining the conceptual independence between the two approaches.

3.1 CLASS: OwnerController METHOD: showOwner()

Code generated for this method

File ShowOwnerManualTest.java (check in folder)

PATH: "experimental\manual\funcionales\ShowOwnerManualTest.java"

Result obtained in the test

```
[INFO] Results:

[INFO]

[ERROR] Errors:

[ERROR]   ShowOwnerManualTest.shouldReturnServerErrorWhenOwnerIdIsNegative:43 » Servlet Request processing failed:
java.lang.IllegalArgumentException: Owner not found with id: -1. Please ensure the ID is correct and the owner exists in the database.

[ERROR]   ShowOwnerManualTest.shouldReturnServerErrorWhenOwnerNotFound:31 » Servlet Request processing failed:
java.lang.IllegalArgumentException: Owner not found with id: 11. Please ensure the ID is correct and the owner exists in the database.

[INFO]

[ERROR] Tests run: 4, Failures: 0, Errors: 2, Skipped: 0

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] BUILD FAILURE

[INFO] ------------------------------------------------------------------------

[INFO] Total time:  18.323 s

[INFO] Finished at: 2025-10-27T17:42:04-05:00

[INFO] ------------------------------------------------------------------------
```

Comentado [JB7]: The functional tests designed by the researcher allowed for the evaluation of how the `showOwner()` method handles different types of input in the owner query. It was confirmed that when the identifier exists in the database, the system responds correctly with code 200 and the expected view. However, cases where the owner does not exist or the identifier is invalid resulted in runtime errors due to the propagation of an unhandled `IllegalArgumentException`, which generates a code 500 instead of a 4xx status code. Additionally, the use of non-numeric parameters in the path caused a 400 error due to incompatibility with the expected type. These results accurately reflect the actual behavior of the system under test (SUT), which lacks an exception handling mechanism for these cases. These results remain unchanged to preserve the study's neutrality and avoid adjustments that might favor the manual tests performed by the researcher.

3.2 CLASS: PetController METHOD: processCreationForm()

Code generated for this method

File ProcessCreationFormManualTest.java (check in folder)

PATH: "experimental\manual\funcionales\ProcessCreationFormManualTest.java"

Result obtained in the test

[INFO] Results:

[INFO]

[ERROR] Errors:

[ERROR] ProcessCreationFormManualTest.shouldReturnFormWhenTypeIsEmpty:101 » Servlet Request processing failed: org.springframework.dao.DataIntegrityViolationException: could not execute statement [La columna "TYPE_ID" no permite valores nulos (NULL)

NULL not allowed for column "TYPE_ID"; SQL statement:

insert into pets (birth_date,name,type_id,id) values (?,?,?,default) [23502-232]] [insert into pets (birth_date,name,type_id,id) values (?,?,?,default)]; SQL [insert into pets (birth_date,name,type_id,id) values (?,?,?,default)]; constraint [null]

[ERROR] ProcessCreationFormManualTest.shouldReturnServerErrorWhenInvalidType:63 » Servlet Request processing failed: org.springframework.orm.jpa.JpaObjectRetrievalFailureException: Unable to find org.springframework.samples.petclinic.owner.PetType with id -1

[ERROR] ProcessCreationFormManualTest.shouldReturnServerErrorWhenOwnerDoesNotExist:112 » Servlet Request processing failed: java.lang.IllegalArgumentException: Owner not found with id: 30. Please ensure the ID is correct

[ERROR] ProcessCreationFormManualTest.shouldReturnServerErrorWhenTypeDoesNotExist:73 » Servlet Request processing failed: org.springframework.orm.jpa.JpaObjectRetrievalFailureException: Unable to find org.springframework.samples.petclinic.owner.PetType with id 8

[INFO]

[ERROR] Tests run: 10, Failures: 0, Errors: 4, Skipped: 0

[INFO]

[INFO] --

[INFO] BUILD FAILURE

[INFO] --

[INFO] Total time: 18.582 s

[INFO] Finished at: 2025-10-27T17:50:10-05:00

[INFO] --

Comentado [JB8]: The functional tests designed by the researcher for the `processCreationForm()` method allowed for the evaluation of a wide variety of user inputs in the pet registration form. Successful creation scenarios showed that the system responds correctly with redirection and a confirmation message. However, several error cases resulted in runtime failures due to unhandled exceptions, such as `DataIntegrityViolationException`, `JpaObjectRetrievalFailureException`, and `IllegalArgumentException`, leading to 500 Internal Server Error codes instead of validated handling within the form. This occurs when the pet type is invalid, does not exist in the database, or an attempt is made to associate the pet with a non-existent owner. Only in cases where the form data violates Spring's validation rules does the system correctly display the form view with the corresponding errors. The results are presented without altering the behavior of the System Under Test (SUT), respecting the neutrality of the experiment and accurately reflecting the system's weaknesses in the face of unforeseen inputs.

3.3 CLASS: VisitController METHOD: processNewVisitForm()

Code generated for this method

File ProcessNewVisitFormManualTest.java (check in folder)

PATH: “experimental\manual\funcionales\ProcessNewVisitFormManualTest.java"

Result obtained in the test

```
[INFO] Results:

[INFO]

[ERROR] Failures:

[ERROR]   ProcessNewVisitFormManualTest.shouldReturnFormWhenDateIsEmpty:72 View name expected:<pets/createOrUpdateVisitForm> but was:<redirect:/owners/{ownerId}>

[ERROR] Errors:

[ERROR]   ProcessNewVisitFormManualTest.shouldReturnServerErrorWhenOwnerDoesNotExist:51 » Servlet Request processing failed: java.lang.IllegalArgumentException: Owner not found with id: 99. Please ensure the ID is correct

[ERROR]   ProcessNewVisitFormManualTest.shouldReturnServerErrorWhenPetDoesNotExist:60 » Servlet Request processing failed: java.lang.NullPointerException: Cannot invoke "org.springframework.samples.petclinic.owner.Pet.addVisit(org.springframework.samples.petclinic.owner.Visit)" because "pet" is null

[INFO]

[ERROR] Tests run: 7, Failures: 1, Errors: 2, Skipped: 0

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] BUILD FAILURE

[INFO] ------------------------------------------------------------------------

[INFO] Total time:  17.530 s

[INFO] Finished at: 2025-10-27T17:57:07-05:00

[INFO] ------------------------------------------------------------------------
```

Comentado [JB9]: The tests performed by the researcher on this method validated the behavior of the visitor log in various system scenarios. It was confirmed that a successful flow returns a redirect and an informational message, while incomplete entries, such as an invalid date or description, return the creation form as expected according to the validation logic. However, three tests revealed execution failures that produced 500 errors due to the lack of control over internally generated exceptions: an IllegalArgumentException when the owner did not exist, a NullPointerException when attempting to add a visit to a non-existent pet, and a case where an empty date caused an unexpected redirect, indicating inconsistency in field validation. These results reflect real limitations of the System Under Test (SUT) when faced with defective or unforeseen inputs, which were left uncorrected to safeguard the neutrality of the experiment and prevent alterations that could favor the manual tests designed by the researcher.

4 CHAPTER 4 – UNIT TESTS PERFORMED BY THE RESEARCHER

In this case, the procedure followed was like that used in functional testing; however, the verification focused directly on the internal behavior of the method and not on HTTP requests, since these correspond exclusively to the scope of functional testing.

One unique aspect of creating these unit tests is that, before implementing them, I ran test model classes to observe the results obtained by each method in the console. This preliminary execution allowed me to identify the internal flow, return values, potential exceptions, and side effects, which facilitated decision-making regarding the structure and scope of the unit tests to be developed.

4.1 Unit test – Owner class, addPet() method

Code generated for test model

File ManualOwnerModelTest.java (check in folder)

PATH: "experimental\manual\unitarias\model\ManualOwnerModelTest.java"

Result obtained in the test model

Case 1 - New Pet (ID null):

Was it added? true

Number of pets: 1

Case 2 - Existing Pet (ID != null):

Was it added? false

Number of pets: 1

Case 3 - Pet null:

Error thrown: NullPointerException

Case 4 - Duplicate name:

Number of pets: 3

Are there duplicates? 2 with the name Lucky

Case 5 - Bidirectional relationship:

pet1 is in the owner's collection: true

Code generated for this method

File OwnerAddPetUnitManualTest.java (check in folder)

PATH: "experimental\manual\unitarias\OwnerAddPetUnitManualTest.java"

Result obtained in the test

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running org.springframework.samples.petclinic.owner.OwnerAddPetUnitManualTest
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.079 s -- in org.springframework.samples.petclinic.owner.OwnerAddPetUnitManualTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jacoco:0.8.13:report (report) @ spring-petclinic ---
[INFO] Loading execution data file C:\Users\doleh\Downloads\development\spring-petclinic\target\jacoco.exec
[INFO] Analyzed bundle 'petclinic' with 22 classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  9.585 s
[INFO] Finished at: 2025-10-27T18:14:42-05:00
[INFO] ------------------------------------------------------------------------
```

Comentado [JB10]: The unit tests designed by the researcher for the `addPet()` method explicitly evaluated the behaviors previously observed in the model. It was confirmed that a new pet (without an assigned ID) is correctly added to the owner's list and that pets with existing IDs—considered already persisted—are not added back to the internal collection. It was also verified that the method throws an exception when attempting to add a null object, indicating the absence of preventative parameter checking. Furthermore, it was confirmed that the system allows pets with duplicate names and that the relationship with the owner is correctly established when adding a pet. All executed tests were successful and accurately represented the actual behavior of the system without requiring modifications, thus maintaining the impartiality of the study and allowing for the analysis of the functionalities as originally designed.

4.2 Unit test – Owner class, getPet(String name, boolean ignoreNew) method

Code generated for test model

File ManualOwnerGetPetTest.java (check in folder)

PATH: "experimental\manual\unitarias\model\ManualOwnerGetPetTest.java"

Result obtained in the test model

Case 1 - Search by existing name (Lucky):

Result: Lucky

Case 2 - Search by non-existent name (Rocky):

Result: null

Case 3 - Case-insensitive search (lUcKy):

Result: Lucky

Case 4 - Duplicate name (Lucky), should return the first one:

Does it correspond to the first saved name? false

Case 5 - New pet with ignoreNew=true (Bobby):

Result: null

Case 6 - New pet with ignoreNew=false (Bobby):

Result: Bobby

Case 7 - Search by existing ID (10):

Result: null

Case 8 - ID belongs to new pet (null id ignored):

Result: null

Case 9 - Non-existent ID (99):

Result: null

Case 10 - Pet with null name does not cause an error:

Result of null search: null

Case 11 - Empty list (new Owner):

Result of empty list: null

Pontificia Universidad Católica del Ecuador
Seréis mis testigos
ESMERALDAS

Code generated for this method

File OwnerGetPetUnitManualTest.java (check in folder)

PATH: "experimental\manual\unitarias\OwnerGetPetUnitManualTest.java"

Result obtained in the test

```
[INFO] Results:
[INFO]
[ERROR] Failures:
[ERROR]   OwnerGetPetUnitManualTest.shouldIgnoreCaseWhenSearchingByName:59
Expecting actual not to be null
[ERROR]   OwnerGetPetUnitManualTest.shouldReturnLastPetWhenThereAreDuplicates:66
expected: Lucky
 but was: null
[ERROR]   OwnerGetPetUnitManualTest.shouldReturnPetByName:44
Expecting actual not to be null
[INFO]
[ERROR] Tests run: 7, Failures: 3, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  9.961 s
[INFO] Finished at: 2025-10-27T18:56:30-05:00
[INFO] ------------------------------------------------------------------------
```

Comentado [JB11]: Manual unit tests performed on the getPet() method revealed significant discrepancies between the expected behavior and the actual implementation of the System Under Test (SUT). It was observed that when searching by name, the method returns null even when the pet exists in the collection, reflecting a flaw in the retrieval logic. It was also found that the case-insensitive search does not function correctly. Furthermore, the method returns incorrect values when duplicate names exist, delivering a null result instead of the last match detected, as observed during the execution of the test model. Finally, it was verified that the query by identifier does not return the corresponding pet, even when the ID matches the stored object, confirming a functional limitation of the method. All these results demonstrate real defects in the SUT's internal search functionality for the Owner object and were left unchanged to ensure the neutrality of the study and avoid any bias in the evaluation of the AI-generated and manually designed tests.

4.3 Unit test – PetValidator class, validate() method

Code generated for test model

File ManualPetValidatorModelTest.java (check in folder)

PATH: "experimental\manual\unitarias\model\ManualPetValidatorModelTest.java"

Result obtained in the test model

Case 1 - Empty Name:

Error? Field: name, Code: required

Case 2 - Name with spaces:

Error? Field: name, Code: required

Case 3 - Valid Name:

? No errors

Case 4 - New Pet without type:

Error? Field: type, Code: required

Case 5 - Existing Pet without type:

? No errors

Case 6 - Null Date:

Error? Field: birthDate, Code: required

Case 7 - All null:

Error? Field: name, Code: required

Error? Field: type, Code: required

Error? Field: birthDate, Code: required

Case 8 - Null object:

? Error thrown: NullPointerException

Code generated for this method

File PetValidatorUnitManualTest.java (check in folder)

PATH: "experimental\manual\unitarias\PetValidatorUnitManualTest.java"

Result obtained in the test

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running org.springframework.samples.petclinic.experimental.manual.unitarias.model.PetValidatorUnitManualTest
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.347 s -- in org.springframework.samples.petclinic.experimental.manual.unitarias.model.PetValidatorUnitManualTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jacoco:0.8.13:report (report) @ spring-petclinic ---
[INFO] Loading execution data file C:\Users\doleh\Downloads\development\spring-petclinic\target\jacoco.exec
[INFO] Analyzed bundle 'petclinic' with 22 classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  10.157 s
[INFO] Finished at: 2025-10-27T19:10:36-05:00
[INFO] ------------------------------------------------------------------------
```

Comentado [JB12]: The manual unit tests developed for the PetValidator's validate() method thoroughly validated the consistency rules defined for pet data. It was confirmed that the validator correctly rejects empty names or names composed solely of spaces, as well as new pets without a specified type or date of birth, thus ensuring data integrity before persistence. It was also verified that once the pet is persisted (ID assigned), the lack of a specified type is not considered an error, which aligns with the logic implemented in the System Under Test (SUT). Finally, it was confirmed that the method throws an exception when it receives a null object, revealing that no preventative control is applied to invalid inputs. All tests were successful, and their results were retained in the SUT without modification, preserving the neutrality of the experiment and reflecting the validator's actual behavior under different scenarios.

5 METHODS EVALUATED

5.1 Functional methods

5.1.1 SHOWOWNER()

PATH: src\main\java\org\springframework\samples\petclinic\owner\OwnerController.java; 163-171

```
@GetMapping("/owners/{ownerId}")
    public ModelAndView showOwner(@PathVariable("ownerId") int ownerId) {
        ModelAndView mav = new ModelAndView("owners/ownerDetails");
        Optional<Owner> optionalOwner = this.owners.findById(ownerId);
        Owner owner = optionalOwner.orElseThrow(() -> new IllegalArgumentException(
                "Owner not found with id: " + ownerId + ". Please ensure the ID is correct "));
        mav.addObject(owner);
        return mav;
    }
```

5.1.2 PROCESSCREATIONFORM()

PATH: src\main\java\org\springframework\samples\petclinic\owner\PetController.java; 102-122

```
@PostMapping("/pets/new")
    public String processCreationForm(Owner owner, @Valid Pet pet, BindingResult result,
            RedirectAttributes redirectAttributes) {

        if (StringUtils.hasText(pet.getName()) && pet.isNew() && owner.getPet(pet.getName(), true) != null)
            result.rejectValue("name", "duplicate", "already exists");

        LocalDate currentDate = LocalDate.now();
        if (pet.getBirthDate() != null && pet.getBirthDate().isAfter(currentDate)) {
            result.rejectValue("birthDate", "typeMismatch.birthDate");
        }

        if (result.hasErrors()) {
            return VIEWS_PETS_CREATE_OR_UPDATE_FORM;
        }

        owner.addPet(pet);
        this.owners.save(owner);
        redirectAttributes.addFlashAttribute("message", "New Pet has been Added");
        return "redirect:/owners/{ownerId}";
    }
```

5.1.3 PROCESSNEWVISITFORM()

PATH: src\main\java\org\springframework\samples\petclinic\owner\VisitController.java; 87-98

```
@PostMapping("/owners/{ownerId}/pets/{petId}/visits/new")
    public String processNewVisitForm(@ModelAttribute Owner owner, @PathVariable int petId, @Valid Visit visit,
            BindingResult result, RedirectAttributes redirectAttributes) {
        if (result.hasErrors()) {
            return "pets/createOrUpdateVisitForm";
        }

        owner.addVisit(petId, visit);
        this.owners.save(owner);
        redirectAttributes.addFlashAttribute("message", "Your visit has been booked");
        return "redirect:/owners/{ownerId}";
    }
```

5.2 Unit Tests

5.2.1 ADDPET()

PATH: src\main\java\org\springframework\samples\petclinic\owner\Owner.java; 96-100

```java
public void addPet(Pet pet) {
        if (pet.isNew()) {
            getPets().add(pet);
        }
    }
```

5.2.2 GETPET()

PATH: src\main\java\org\springframework\samples\petclinic\owner\Owner.java; 102-144

```java
/**
     * Return the Pet with the given name, or null if none found for this Owner.
     * @param name to test
     * @return the Pet with the given name, or null if no such Pet exists for this Owner
     */
    public Pet getPet(String name) {
        return getPet(name, false);
    }

    /**
     * Return the Pet with the given id, or null if none found for this Owner.
     * @param id to test
     * @return the Pet with the given id, or null if no such Pet exists for this Owner
     */
    public Pet getPet(Integer id) {
        for (Pet pet : getPets()) {
            if (!pet.isNew()) {
                Integer compId = pet.getId();
                if (compId.equals(id)) {
                    return pet;
                }
            }
        }
        return null;
    }

    /**
     * Return the Pet with the given name, or null if none found for this Owner.
     * @param name to test
     * @param ignoreNew whether to ignore new pets (pets that are not saved yet)
     * @return the Pet with the given name, or null if no such Pet exists for this Owner
     */
    public Pet getPet(String name, boolean ignoreNew) {
        for (Pet pet : getPets()) {
            String compName = pet.getName();
            if (compName != null && compName.equalsIgnoreCase(name)) {
                if (!ignoreNew || !pet.isNew()) {
                    return pet;
                }
            }
        }
        return null;
    }
```

5.2.3 VALIDATE()

PATH: src\main\java\org\springframework\samples\petclinic\owner\PetValidator.java; 36-54

```java
@Override
    public void validate(Object obj, Errors errors) {
        Pet pet = (Pet) obj;
        String name = pet.getName();
        // name validation
        if (!StringUtils.hasText(name)) {
            errors.rejectValue("name", REQUIRED, REQUIRED);
        }

        // type validation
        if (pet.isNew() && pet.getType() == null) {
            errors.rejectValue("type", REQUIRED, REQUIRED);
        }

        // birth date validation
        if (pet.getBirthDate() == null) {
            errors.rejectValue("birthDate", REQUIRED, REQUIRED);
        }
    }
```