Cucumber - Glue

In **Cucumber**, the term **"glue"** refers to the **code that connects your feature files (written in Gherkin) to the step definitions (written in Java or other languages)**.

📌 In Simple Terms:

- Feature files contain **scenarios** written in plain English (Given-When-Then).
- Glue code is the **Java code** that defines what should happen when a step like `Given I am on the login page` is executed.
- Glue tells Cucumber where to **look for step definitions**.

🧩 Example:

1. Feature File (login.feature)

```gherkin
gherkin

Feature: Login feature

  Scenario: Successful login
    Given I am on the login page
    When I enter valid credentials
    Then I should be redirected to the homepage
```

2. Step Definition (Java)

```java
java

package stepDefinitions;

import io.cucumber.java.en.*;

public class LoginSteps {

    @Given("I am on the login page")
    public void i_am_on_the_login_page() {
        System.out.println("User is on login page");
    }

    @When("I enter valid credentials")
    public void i_enter_valid_credentials() {
        System.out.println("User enters valid username and password");
    }

    @Then("I should be redirected to the homepage")
    public void i_should_be_redirected_to_the_homepage() {
        System.out.println("User is redirected to homepage");
    }
}
```

3. Test Runner

```
java


package testRunner;

import org.junit.runner.RunWith;
import io.cucumber.junit.*;

@RunWith(Cucumber.class)
@CucumberOptions(
    features = "src/test/resources/features",
    glue = "stepDefinitions"  // 👈 This is the glue code
)
public class TestRunner {
}
```

✅ Key Points:

- The **`glue`** option in **`@CucumberOptions`** tells Cucumber **where to find step definitions**.
- If glue is not specified and step definitions are in a different package, Cucumber won't know where to look, and you'll get **`Undefined step`** errors.

Would you like a mini hands-on project to try this out?

Sure! Let's break down the line:

```
java

glue = {"stepDefinitions"}
```

This line appears inside the `@CucumberOptions` annotation in your **Test Runner** class in a Cucumber + Java project.

🔍 Explanation:

- `glue`: This is a Cucumber option that tells the test runner **where to look for step definitions** (i.e., the Java methods that define what each step does).
- `"stepDefinitions"`: This is the **package name** where your step definition classes are located.

🧠 How it works:

When you run your feature file, Cucumber reads each step like:

```
gherkin

Given I am on the login page
```

To execute it, Cucumber needs to find a matching method in Java like:

```
java

@Given("I am on the login page")
public void i_am_on_the_login_page() {
    // your code here
}
```

Cucumber uses the `glue` path to **search for these methods**. If you don't set the correct glue path, Cucumber won't find the step definitions, and you'll get errors like:

```
arduino

You can implement this step with the following snippet:
```

🧱 Directory Structure Example:

```
bash

src/test/java/
├── stepDefinitions/
│   └── LoginSteps.java
└── testRunner/
    └── TestRunner.java
```

Then your `TestRunner.java` might look like:

```
java


@RunWith(Cucumber.class)
@CucumberOptions(
    features = "src/test/resources/features",
    glue = {"stepDefinitions"}  // 👈 Refers to this folder: src/test/java/stepDefinitions
)
public class TestRunner {
}
```

✅ Summary:

Part	Meaning
`glue`	A Cucumber option to locate step definition files
`"stepDefinitions"`	The package (not the path) where step definitions are written

Let me know if you want to try a working example or need help setting up the folder structure.

Here's a **standard folder structure** for a **Cucumber + Java + JUnit** project showing exactly **where glue code goes**, and how everything fits together:

✅ Standard Maven Folder Structure

```
vbnet

project-root/
├── pom.xml
└── src/
    ├── main/
    │   └── java/
    │       └── (optional: app logic if needed)
    └── test/
        ├── java/
        │   ├── stepDefinitions/          ← ✅ Glue code (step definitions) goes here
        │   │   └── LoginSteps.java
        │   ├── testRunner/               ← ✅ Test runner class goes here
        │   │   └── TestRunner.java
        │   └── utils/                    ← (optional) helper classes, hooks, etc.
        └── resources/
            └── features/                 ← ✅ Feature files (.feature) go here
                └── login.feature
```

🗂️ `stepDefinitions/`

- This package contains all the **Java methods** annotated with `@Given`, `@When`, `@Then`, etc.
- These are the **glue code**—the bridge between your `.feature` file and test automation logic.

🗂️ `testRunner/`

- Contains the `TestRunner.java` file with `@RunWith` and `@CucumberOptions`.
- Specifies where to find features and glue using:

```
java

@CucumberOptions(
    features = "src/test/resources/features",
    glue = {"stepDefinitions"}
)
```

🗂️ `features/`

- Stores `.feature` files written in Gherkin.
- These define scenarios in plain English.

🧩 Example File Contents

1. login.feature

```
gherkin

Feature: Login

  Scenario: Successful login
    Given I am on the login page
    When I enter valid credentials
    Then I should be redirected to the homepage
```

2. LoginSteps.java

```
java

package stepDefinitions;

import io.cucumber.java.en.*;

public class LoginSteps {

    @Given("I am on the login page")
    public void i_am_on_the_login_page() {
        System.out.println("User is on login page");
    }

    @When("I enter valid credentials")
    public void i_enter_valid_credentials() {
        System.out.println("User enters credentials");
    }

    @Then("I should be redirected to the homepage")
    public void i_should_be_redirected_to_the_homepage() {
        System.out.println("User is on homepage");
    }
}
```

3. TestRunner.java

```
java

package testRunner;

import org.junit.runner.RunWith;
import io.cucumber.junit.*;

@RunWith(Cucumber.class)
@CucumberOptions(
    features = "src/test/resources/features",
    glue = {"stepDefinitions"}  // 👈 Looks in this package
)
public class TestRunner {
}
```

🔧 Notes:

- **`glue = {"stepDefinitions"}`** must match the **package name**, not the folder path.
- If you use multiple glue packages (like **`hooks`**, **`utils`**), you can do:

```
java

glue = {"stepDefinitions", "hooks"}
```

Would you like a ready-made Maven project template to try this out locally?