By OnlineInterviewQuestions.com

JBehave Interview Questions and Answers

About JBehave

JBehave is a software-based in java language for Behaviour Driven Development (BDD). It is developed to make practices such as test-driven development(TDD) and Acceptance-driven design(ATD) which are an expansion of Behaviour-driven development(BDD), more accessible for the newcomers as well as experts. It also supports creating custom data types. It is purely a java implementation that works well with java enterprises.

We have listed here the **best JBehave Interview Questions and Answers**. These JBehave Interview Questions are very helpful for the preparation of the Jbehave Interview. You can also download here the **Jbehave Interview Questions PDF**.

Q1. Explain what is JBehave?

JBehave is a software-based in java language for **Behaviour Driven Development** (BDD). It is developed to make practices such as **test-driven development**(TDD) and **Acceptance-driven design**(ATD) which are an expansion of **Behaviour-driven development**(BDD), more accessible for the newcomers as well as experts. It also supports creating custom data types. It is purely a java implementation that works well with java enterprises.

Q2. Enlist the major features of JBehave?

The following are the features of Jbehave:

- It has the feature of Ant integration which gives its stories an advantage to run via an Ant task.
- User stories can be written in JBehave syntax and Gherkin syntax.
- While specifying the number of concurrent threads, users' stories can be executed concurrently.
- The user's stories can be specified as classpath resources or external URL based resources.
- By Maven Integration, user's stories are allowed to run via Maven plugin at any given build phase.

Q3. List different typs of modules in JBehave?

The following are the different types of modules in JBehave are:

- **Game of life**: used to verify multi-line scenarios behavior
- **Gherkin**: it verifies to pass gherkin based features
- **Groovy**: verifies groovy based behavior for writing steps.

- **JRuby**: Verifies JRuby behaviors
- **Performance**: verifies performance behavior
- **Spring Security**: verifies spring-based security behaviors
- **Scala**: verifies scala based behaviors
- **Google**: verifies to retrieve stories from google docs
- **Rest**: verifies to retrieve stories from rest APIs
- **Threads**: verifies multi-threading behaviors

Q4. What are annotations in JBehave?

Jbehave supports the following annotations:

Step annotations

- @Given
- @When
- @Then
- @Alias
- @Aliases
- @Pending

Scenario Annotations

- @BeforeScenario
- @AfterScenario

Story Annotations

- @BeforeStory
- @AfterStory

Stories Annotations

- @BeforeStories
- @Afterstories

Parameter Annotations

- @Named

Configuration Annotations

- @AsparamterCoverter
- @Configure
- @UsingEmbedder
- @UsingSteps
- @UsingGuice

- @UsingNeedle
- @NeedleInjectionsProvider
- @UsingPico
- @UsingSpring

Q5. How to run multiple stories in JBehave?

To run multiple stories in JBehave, you can use the **Ant task** or **Maven task**. But you need to specify your story's location and which configuration should be used for its execution.

There are simple steps for you to follow to run your stories in JBehave.

1. Write a story
2. Map the steps in the story to java code
3. Run jBehave tests
4. Review test results

Q6. How to get scenario name in JBehave?

For this you need to create a new class which extends **org.ibehave.core.reporters.consoleOutput** and then you can modify various methods. After this, you need to create a new instance of abstract class and then you need to add the format of your story builder which you are using in your configuration.

Q7. What are JBehave optional parameters?

JBehave optional parameters are Tabular parameters and parameter injection including a subdivision of ordered parameters, Annotated named parameters, para name named parameters.

Q8. Explain the programmatic lifecycle of jbehave?

JBehave mainly supports two ways to control the lifecycle of a story.

Programmatic lifecycle controls are code-centric. It is best suited for technical instead of functional controls. The @BeforeScenario and @AfterScenario allow the corresponding methods to be executed before and after each scenario. The @AfterStory allows the settings of an optional outcome which specifies whether the method should be executed depending on the outcomes of the scenarios.

Q9. What is meta filtering in jbehave?

A meta filter is represented as a **string** and **supports multiple languages** used based on the **prefix used**. JBehave supports meta tags to give you the benefit of marking some set of tests as p1 tests. It also includes running the tests include/exclude to p1 tests. Upon running stories you can easily see that one story file is excluded by the meta filter.

Now when you open the results folder the marked story marked as p1 has run but the stories got excluded and haven't run.

Q10. What is the use of @skip in jbehave?

It is used to **skip** the kind of story that you don't want to execute in your program. By using the meta filter '**Skip**' you can easily skip the files you don't want.

Q11. List the major keywords used that are used in featured file?

The primary keywords involve Feature, Rule, Example, Given, When, Then, And, But(Steps), Background, scenario outline, and Examples.