



# AD0-E134<sup>Q&As</sup>

Adobe Experience Manager Developer Exam

**Pass Adobe AD0-E134 Exam with 100% Guarantee**

Free Download Real Questions & Answers **PDF** and **VCE** file from:

<https://www.geekcert.com/ad0-e134.html>

100% Passing Guarantee  
100% Money Back Assurance

Following Questions and Answers are all new published by Adobe  
Official Exam Center

- ⚙️ **Instant Download** After Purchase
- ⚙️ **100% Money Back** Guarantee
- ⚙️ **365 Days** Free Update
- ⚙️ **800,000+** Satisfied Customers





### QUESTION 1

An AEM application development team is assigned a task to create an Event-Driven Data Layer implementation for an Analytics solution. Which Adobe recommended best practice should the developer choose?

- A. Use Adobe Experience Platform's data layer to integrate with AEM.
- B. Create a custom data layer and add each component template, and its properties to the data layer
- C. Use Adobe Client Data Layer and integrate with Core components.
- D. Create an Adobe Cloud Service configuration to use third-party tool's data layer.

Correct Answer: C

Adobe Client Data Layer is a JavaScript library that provides a standardized way to collect, structure, and manage data on a web page. It can be used to implement an event-driven data layer for analytics solutions. It integrates with Core components and allows authors to configure data layer properties for each component. It also supports custom events and data sources. References: <https://experienceleague.adobe.com/docs/experience-manager-core-components/using/developing/data-layer.html?lang=en><https://github.com/adobe/adobe-client-data-layer>

### QUESTION 2

A custom component has one dialog field:

-> Title

-fieldLabel = Title

-sling:resourceType = granite/ui/components/coral/foundation/form/textfield

-name = ./title

The developer needs to implement a Sling Model to perform a business logic on the authored value. The developer writes the following HTL snippet.

```
<sly data-sly-use.display="com.adobe.aem.guides.certification.core.models.HelloWorldModelImpl">  
<h1>${display.messageText}</h1>  
</sly>
```

Which two implementations will support this HTL snippet? (Choose two.)



A.

`@Model(adaptables = Resource.class, defaultInjectionStrategy = DefaultInjectionStrategy.OPTIONAL)``public class HelloWorldModelImpl {``@ScriptVariable``private String authoredVal;``private String messageText;``@PostConstruct``public void init() {` `if (StringUtils.isNotBlank(authoredVal)) {` `setMessageText(StringUtils.join("Welcome", StringUtils.SPACE, authoredVal));` `}``}``public void setMessageText(String messageText) {` `this.messageText = messageText;``}``public String getMessageText() {` `return messageText;``}``}`



☐ B.

```
public void init() {
    if (StringUtils.isNotBlank(title)) {
        setMessageText(StringUtils.join("Welcome", StringUtils.SPACE, title));
    }
}

public void setMessageText(String messageText) {
    this.messageText = messageText;
}

public String getMessageText() {
    return messageText;
}

}

@Model(adaptables = SlingHttpServletRequest.class, defaultInjectionStrategy = DefaultInjectionStrategy.OPTIONAL)
public class HelloWorldModelImpl {
    @Inject
    @Via("resource")
    private String title;
    private String messageText;
```

☐ C.

```
@PostConstruct
public void init() {
    if (StringUtils.isNotBlank(title)) {
        setMessageText(StringUtils.join("Welcome", StringUtils.SPACE, title));
    }
}

public void setMessageText(String messageText) {
    this.messageText = messageText;
}

public String getMessageText() {
}

}

@Model(adaptables = Resource.class, defaultInjectionStrategy = DefaultInjectionStrategy.OPTIONAL)
public class HelloWorldModelImpl {
    @ValueMapValue
    @Named("title")
    private String authoredVal;
    private String messageText;
```

☐ D.

```
@PostConstruct
public void init() {
    if (StringUtils.isNotBlank(title)) {
        setMessageText(StringUtils.join("Welcome", StringUtils.SPACE, title));
    }
}

public void setMessageText(String messageText) {
    this.messageText = messageText;
}

public String getMessageText() {
    return messageText;
}

}
```

A. Option A

B. Option B

C. Option C

D. Option D

Correct Answer: BD



Explanation: Option B and Option D are two implementations that will support the HTL snippet. Option B uses the @Model annotation with the adaptables parameter set to Resource.class. This allows the Sling Model to adapt from a resource object and access its properties using the ValueMap interface. Option B also uses the @Inject annotation with the name parameter set to ".text" to inject the value of the text property into the text field. Option D uses the @Model annotation with the defaultInjectionStrategy parameter set to OPTIONAL. This allows the Sling Model to use optional injection for all fields and avoid null pointer exceptions if a property is missing. Option D also uses the @Inject annotation without any parameters to inject the value of the text property into the text field, using the field name as the default property name. References:

<https://sling.apache.org/documentation/bundles/models.html><https://experienceleague.adobe.com/docs/experience-manager-htl/using-htl/htl-block-statements.html?lang=en#use>

---

### QUESTION 3

On package install content that is already present in the repos must not be overwritten and if not present in the repos it must not be removed.

Which import mode should the developer use?

- A. update
- B. replace
- C. merge

Correct Answer: C

---

### QUESTION 4

A developer is on an AEM application that is being used to calculate an employee's salary. The calculation is done in an OSGi service called CalculationService. This service class has a dependency on one other service, called the EmployeeService.

How should the developer make sure that the critical code in the CalculationService has a high unit test coverage?

- A. Use a mock framework in the unit test to inject the CalculationService
- B. Instantiate the EmployeeService in the unit test and pass it to the constructor of the CalculationService
- C. Use a mock framework in the unit test to inject the EmployeeService
- D. Use the feature flag in the unit test to disable the calls to the EmployeeService

Correct Answer: C

Explanation: A mock framework is a tool that allows creating mock objects that simulate the behavior of real objects in a controlled way. A mock framework can be used in a unit test to inject the EmployeeService dependency into the CalculationService and verify its interactions. This way, the unit test can focus on testing the logic of the CalculationService without relying on the actual implementation of the EmployeeService. References:

<https://sling.apache.org/documentation/development/sling-testing-tools.html><https://wcm.io/testing/aem-mock/usage.html>

---



### QUESTION 5

A developer needs to create a dynamic participant step where the participant is selected automatically at run time.

The developer decides to develop an OSGi service, which needs to implement the `com.day.cq.workflow.exec.ParticipantStepChooser` interface.

Which method should the developer implement from the `com.day.cq.workflow.exec.ParticipantStepChooser` interface?

- A. `String getParticipant(WorkItem workItem, WorkflowSession workflowSession, MetaDataMap metaDataMap)`
- B. `void getParticipant(WorkItem workItem, WorkflowSession workflowSession, MetaDataMap metaDataMap)`
- C. `String getDynamicParticipant(WorkItem workItem, WorkflowSession workflowSession, MetaDataMap metaDataMap)`
- D. `void getDynamicParticipant(WorkItem workItem, WorkflowSession workflowSession, MetaDataMap metaDataMap)`

Correct Answer: A

<https://medium.com/@toimrank/aem-dynamic-participant-step-1cd58e15f28>

[Latest AD0-E134 Dumps](#)

[AD0-E134 PDF Dumps](#)

[AD0-E134 Exam Questions](#)