AD0-E134 Q&As

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

QUESTION 1

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.htmlhttps://experienceleague.adob e.com/docs/experience-manager-htl/using-htl/htl-block-statements.html?lang=en#use

QUESTION 2

What two types of testing are available OOB in AEM Cloud Manager Pipeline? (Select Two.)

A. Code Quality testing

B. Performance testing

C. UI testing

D. Penetration testing

E. Integration testing

Correct Answer: AC

Explanation: Code Quality testing and UI testing are two types of testing that are available OOB in AEM Cloud Manager Pipeline. Code Quality testing checks the code quality of the project using SonarQubeand reports any issues or

vulnerabilities. UI testing checks the functionality and usability of the project using Selenium WebDriver and reports any errors or failures.

References: https://experienceleague.adobe.com/docs/experience-manager- cloud-service/implementing/testing/testing-overview.html?lang=en#testing-types https://experienceleague.adobe.com/docs/experience-manager-cloud- service/

implementing/testing/code-quality-testing.html?lang=enhttps://experienceleague.adobe.com/docs/experience-manager-cloud- service/implementing/testing/ui-testing.html?lang=en

QUESTION 3

An AEM application requires LDAP Service integration to synchronize users/groups. Which two OSGi configuration are required for LDAP integration in AEM? (Select Two.)

A. Apache Jackrabbit Oak AuthorizableActionProvider

B. Apache Jackrabbit Oak Solr server provider

C. Apache Jackrabbit Oak CUG Configuration

D. Apache Jackrabbit Oak External Login Module

E. Apache Jackrabbit Oak Default Sync Handler

Correct Answer: DE

Explanation: The Apache Jackrabbit Oak External Login Module and Apache Jackrabbit Oak Default Sync Handler are the two OSGi configurations that are required for LDAP integration in AEM. The External Login Module defines how AEM connects to the LDAP server and authenticates users against it. The Default Sync Handler defines how AEM synchronizes users and groups from the LDAP server to the repository.
References:https://experienceleague.adobe.com/docs/experience-manager- 65/administering/security/ldap-config.html?lang=en#ldap-integration

QUESTION 4

A developer needs to create sling models for two fields name and occupations. The dialog has two fields, name - a single value field, and occupations - a multi value field. The following code is included in sling models inherited from interface com.adobe.aem.guides.wknd.core.models.Byline

```
package com.adobe.aem.guides.wknd.core.models.impl;
.....
public class BylineImpl implements Byline {
    .......
    @Override
    public List<String> getOccupations() {
        if (occupations != null) {
            Collections.sort(occupations);
            return new ArrayList<String>(occupations);
        } else {
            return Collections.emptyList();
        }
    }
....
}
```

A.
```
<div data-sly-use.byline="com.adobe.aem.guides.wknd.core.models.Byline"
         data-sly-use.placeholderTemplate="core/wcm/components/commons/v1/templates.html"
         data-sly-test.hasContent="${!byline.empty}"
     class="cmp-byline">

     <h2 class="cmp-byline__name">${byline.name}</h2>
     <p class="cmp-byline__occupations">${byline.occupations @ join=', '}</p>
</div>
```

B.
```
<div data-sly-use.byline="com.adobe.aem.guides.wknd.core.models.Byline.impl"
    data-sly-use.placeholderTemplate="core/wcm/components/commons/v1/templates.html"
    data-sly-test.hasContent="${!byline.empty}"
    class="cmp-byline">

   <h2 class="cmp-byline__name">${byline.name}</h2>
   <p class="cmp-byline__occupations">${byline.occupations @ join=', '}</p>
</div>
```

C.
```
<div data-sly-use.byline="com.adobe.aem.guides.wknd.core.models.Byline"
         data-sly-use.placeholderTemplate="core/wcm/components/commons/v1/templates.html"
         data-sly-test.hasContent="${!byline.empty}"
     class="cmp-byline">

     <h2 class="cmp-byline__name">${byline.name}</h2>
     <p class="cmp-byline__occupations">${byline.occupations }</p>
</div>
```

D.
```
<div data-sly-use.byline="com.adobe.aem.guides.wknd.core.models.Byline"
         data-sly-use.placeholderTemplate="core/wcm/components/commons/v1/templates.html"
         data-sly-test.hasContent="${!byline.empty}"
     class="cmp-byline">

     <h2 class="cmp-byline__name">${byline.name @ join=', '}</h2>
     <p class="cmp-byline__occupations">${byline.occupations @ join=', '}</p>
</div>
```

A. Option A

B. Option B

C. Option C

D. Option D

Correct Answer: C

Explanation: Option C is the correct implementation for the Sling Model. Option C 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 C also uses the @Inject annotation with the name parameter set to "./name" and "./occupations" to inject the values of the name and occupations properties into the name and occupations fields. Option C also uses the @Named annotation with the value parameter set to "byline" to specify the name of the Sling Model that can be used in HTL scripts. References: https://sling.apache.org/documentation/bundles/models.htmlhttps://experienceleague.adob e.com/docs/experience-manager-htl/using-htl/htl-block-statements.html?lang=en#use

QUESTION 5

Which configuration/section should be used to resolve the domain name by dispatcher?

A. Configuration in vhosts file

B. Configuration in filters.any

C. Configuration in httpd.conf

D. Configuration in DNS

Correct Answer: D

Explanation: The configuration in DNS (Domain Name System) should be used to resolve the domain name by dispatcher. The DNS resolves the domain names to the IP address of the web server that hosts the dispatcher. The dispatcher then matches the incoming request URL with the cached files or the AEM publish instances. References:https://experienceleague.adobe.com/docs/experience-manager- dispatcher/using/configuring/dispatcher-domains.html?lang=en#client-requests