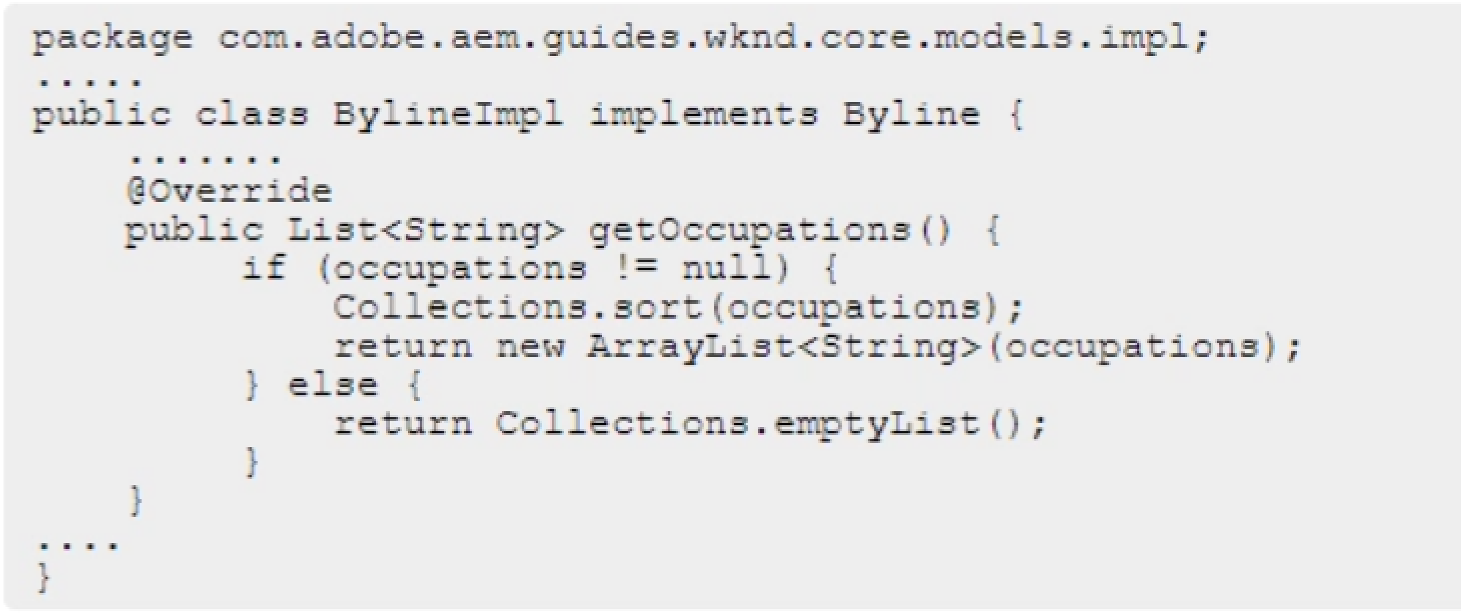
Question 1

Question Type: **MultipleChoice**

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>
```

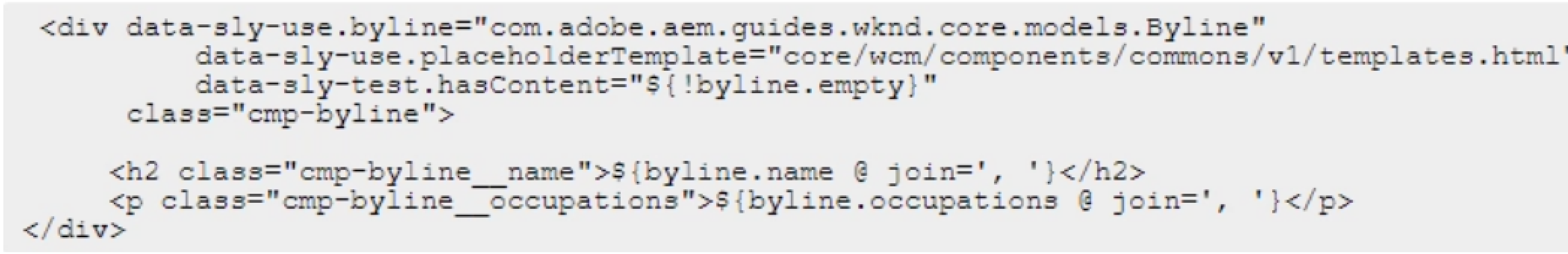
B)

```
<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>
```

Options:

A- Option A

B- Option B

C- Option C

D- Option D

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. Reference: 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 2

Question Type: MultipleChoice

A developer needs to create a new Title component. The requirements are:

1. The layout must be the same as the Title core component

2. The text property must have the page title as prefix (e.g., Page Title -)

3. The component must be reusable

Which approach is recommended?

Options:

A- 1. Create a Proxy Component of Title core component
2. Create a Custom Sling Model that overrides the default behavior
3. Customize the component template

B- 1. Create a custom component from scratch
2. Create a Custom Sling Model for the component that follows the requirement
3. Create a Model Exporter

C- 1. Create a Proxy Component from Title core component
2. Create a Custom Sling Model that overrides the default behavior

Answer:

A

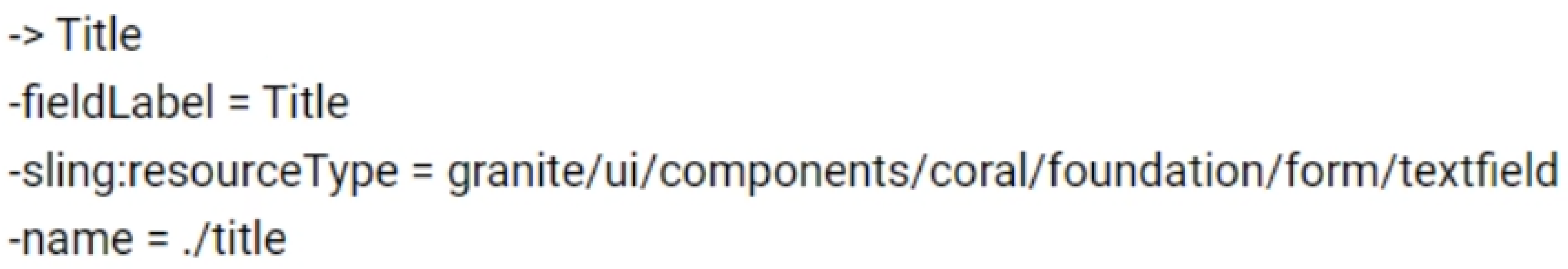
Explanation:

A proxy component is a site-specific component that inherits from a core component and allows customization of the component name, group, dialog, and behavior. A proxy component can refer to any version of the core component by changing the sling:resourceSuperType property. A custom sling model can be used to implement the logic for adding the page title as prefix to the text property. A component template can be used to define the layout of the component.

Question 3

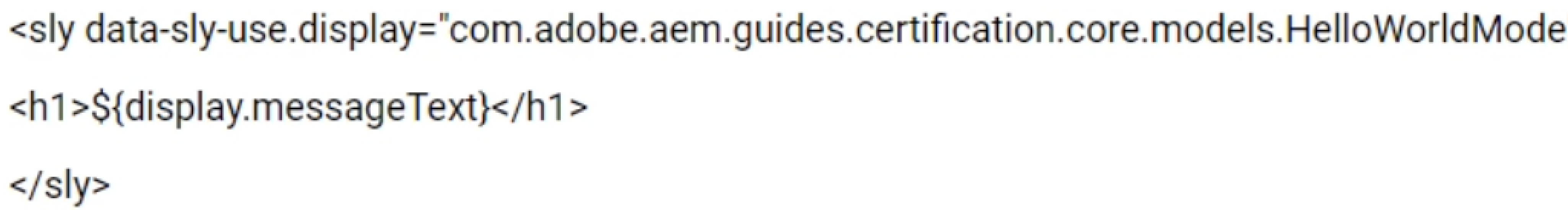
Question Type: **MultipleChoice**

A custom component has one dialog field:



```
-> Title
-fieldLabel = Title
-sling:resourceType = granite/ui/components/coral/foundation/form/textfield
-name = ./title
```

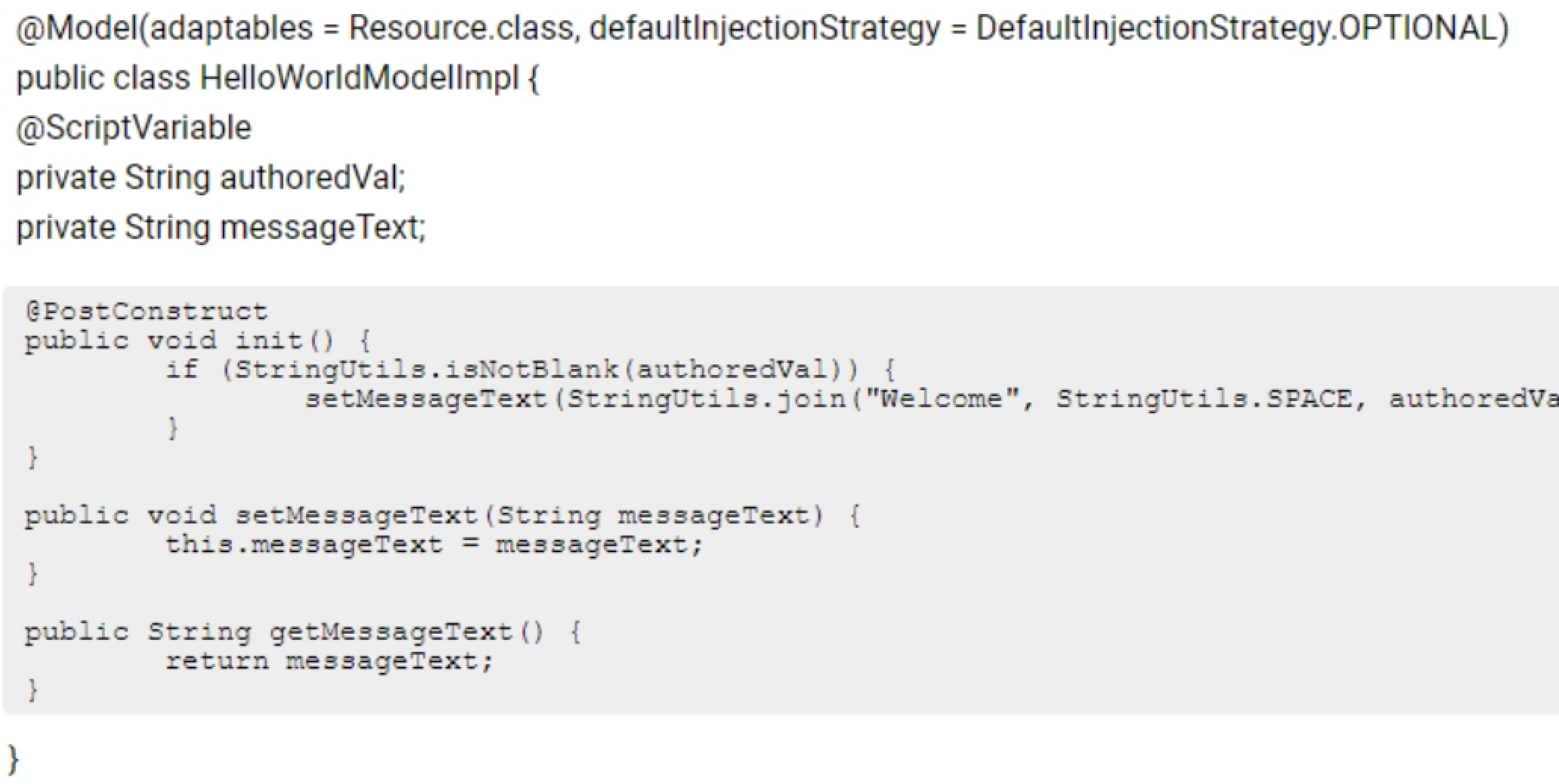
The developer needs to implement a Sling Model to perform a business logic on the authored value. The developer writes the following HTL snippet.



```
<sly data-sly-use.display="com.adobe.aem.guides.certification.core.models.HelloWorldMode

<h1>${display.messageText}</h1>

</sly>
```

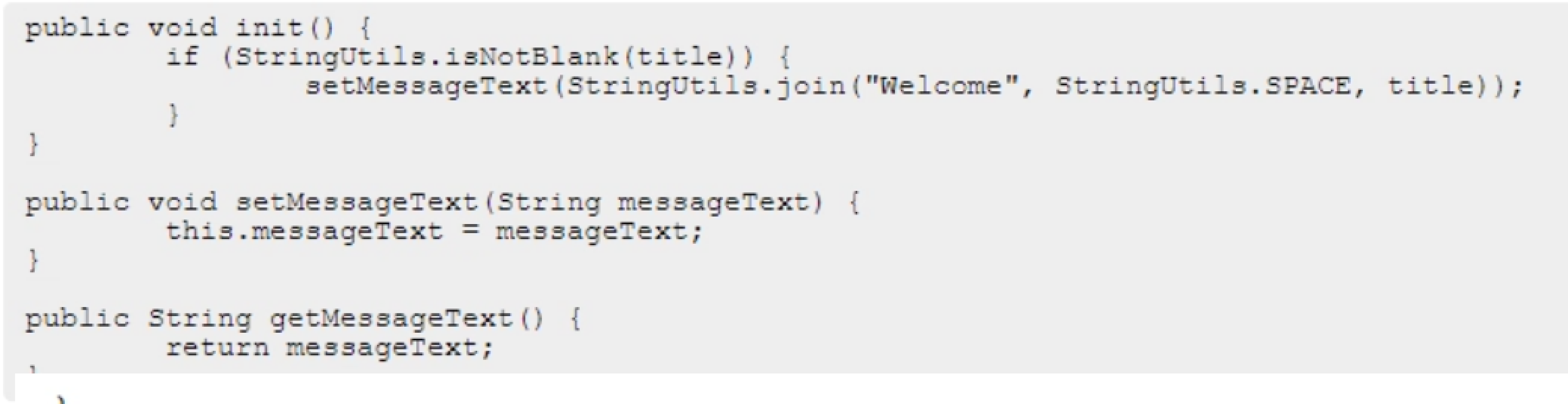
Which two implementations will support this HTL snippet? (Choose two.)

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, authoredVa
        }
}

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

public String getMessageText() {
        return messageText;
}

}
```

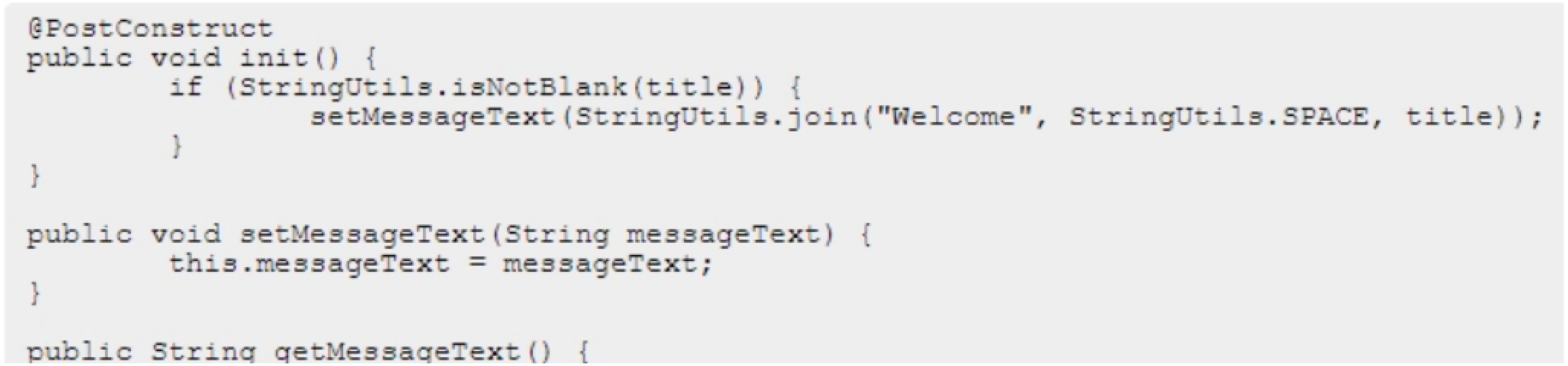
B)



```
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 = DefaultInjectionStrateg
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;
}
```

Options:

A- Option A

B- Option B

C- Option C

D- Option D

Answer:

B, D

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. Reference: 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 4

Question Type: MultipleChoice

An AEM server is overloaded with too many concurrently running workflows. The developer decides to reduce the number of concurrent workflows.

What should be configured to reduce the number of concurrent workflows?

Options:

A- The number of threads in Scheduler

B- The number of threads in Apache Felix Jetty Http Service

C- Launchers for each workflow

D- Maximum Parallel Jobs in OSGI console

Answer:

D

Explanation:

Maximum Parallel Jobs is a configuration property that controls how many workflows can run concurrently on an AEM instance. Reducing this value can help to avoid overloading the server with too many workflows.

Question 5

Question Type: MultipleChoice

A client is having issues with some query results:

* Many of the client's industry terms have the same meaning, and users do not always search the exact wording

* Many users search by typing in short phrases instead of exact keywords, ex:// "cats and dogs"

What index analyzers should the AEM developer recommend?

Options:

A- 1. Add a Mapping filter to the current indexes
2. Add a Stop filter to the current indexes

B- 1. Tokenize the current indexes with a Keyword tokenizer
2. Add a Mapping filter to the current indexes

C- 1. Add a Synonym filter to the current indexes
2. Add a Stop filter to the current indexes

D- 1. Add a Synonym filter to the current indexes
2. Add a LowerCase filter to the current indexes

Answer:

D

Explanation:

A Synonym filter can help to map different terms that have the same meaning, such as "cat" and "feline". A LowerCase filter can help to normalize the case of the terms, so that "cats and dogs" and "Cats and Dogs" are treated the same.

Question 6

Question Type: MultipleChoice

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

Options:

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

Answer:

D, E

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. Reference: https://experienceleague.adobe.com/docs/experience-manager-65/administering/security/ldap-config.html?lang=en#ldap-integration

Question 7

Question Type: MultipleChoice

An AEM application wants to set up multi-tenancy using Adobe-recommended best practices and bind multiple configurations to it. Which of the following options is recommended?

Options:

A- import org.apache.felix.scr.annotations.Component; @Component(label = 'My configuration', metatype = true, factory= true)

B- import org.osgi.service.component.annotations.Component; @Component(service = ConfigurationFactory.class)

C- import org.osgi.service.metatype.annotations.AttributeDefinition;
import org.osgi.service.metatype.annotations.ObjectClassDefinition;
@ObjectClassDefinition(name = 'My configuration')

D- @Component(service = ConfigurationFactory.class)
@Designate(ocd = ConfigurationFactoryImpl.Config.class, factory=true)

Answer:

D

Explanation:

The @Component(service = ConfigurationFactory.class) @Designate(ocd = ConfigurationFactoryImpl.Config.class,factory=true) option is recommended for creating a multi-tenancy configuration and binding multiple configurations to it. This option uses the OSGi R6 annotations to define a component that provides a service of type ConfigurationFactory and designates a class that contains the configuration properties. The factory=true attribute indicates that multiple configurations can be created for this component. Reference: https://experienceleague.adobe.com/docs/experience-manager-65/deploying/configuring/osgi-configuration-settings.html?lang=en#creating-factory-configurations

Question 8

Question Type: MultipleChoice

A developer needs to create a runmode-specific OSGi configuration for an AEM as a Cloud Service implementation. In which location should the OSGi configuration be created?

Options:

A- core project, (/core/.../config <runmode>) folder

B- ui.config project, (/config/.../config.<runmode>) folder

C- all project, (/all/.../config.<runmode>) folder

D- ui.apps project (/apps/.../config.<runmode>) folder

Answer:

B

Explanation:

The ui.config project, (/config/.../config.<runmode>) folder is the location where the OSGi configuration should be created for a runmode-specific configuration for an AEM as a Cloud Service implementation. The ui.config project contains OSGi configurations that are deployed to /apps in the repository. The config.<runmode> folder specifies the runmode for which the configuration is applicable, such as author or publish. Reference: https://experienceleague.adobe.com/docs/experience-manager-cloud-service/implementing/deploying/configuring-osgi.html?lang=en#project-structure