DUMPSBOSS.

Sitecore XM Cloud Developer Certification Exam

Sitecore Sitecore-XM-Cloud-Developer

Version Demo

Total Demo Questions: 7

Total Premium Questions: 72

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

QUESTION NO: 1

A developer can reference an external code base that includes components so they can be used in the Components builder interface. Which of the following are possible results of adding React components to the Components builder? Select all that apply.

A. The registered component is scoped to only one environment.

B. A content author can enter values for the defined input parameters.

C. The registered component is part of the Components library.

D. The registered component can be previewed in the Component builder.

ANSWER: B C D

Explanation:

When a React component is registered from an external code base, the component definition can expose input parameters for authoring. This enables a content author to provide values for those defined inputs through the XM Cloud Pages editing experience, allowing the component's rendered output and behavior to be configured without changing its source code. The registered component is also made available through the Components library, where authors can locate and add it to a page. This is the key connection between developer-created React code and the reusable page-building components available to authors.

The Components builder also supports previewing a registered component while it is being configured and managed. Preview capability helps developers validate the visual result of the component and its input configuration before authors use it in page composition. Together, component registration, input definitions, library availability, and preview provide the intended developer-to-author workflow for custom components in XM Cloud. Sitecore's developer documentation for XM Cloud provides the broader implementation context, while its documentation portal contains the Components builder guidance and related authoring workflows. See Sitecore XM Cloud developer documentation and Sitecore XM Cloud user documentation.

QUESTION NO: 2

A content author is unable to edit a company webpage in Sitecore. Where is the best place to check if the user has Write access to this content item?

A. Security Editor

B. Role Manager

C. Access Viewer

D. Administrator Tools

ANSWER: C

Explanation:

Access Viewer is the best place to verify whether a content author effectively has Write access to a particular content item. It provides an item-focused view of the access rights that apply to a selected user or role, including rights inherited through Sitecore's security hierarchy and rights granted or denied directly on the item. This is especially useful when troubleshooting because an author's effective permission can be affected by membership in multiple roles as well as inheritance from parent items.

In Access Viewer, select the relevant item and user or role, then inspect the effective Write permission. For editing localized content, the author must also have the applicable language rights for the item's language, so the selected language context should be considered during the check. Sitecore documents Access Viewer as a tool for viewing the access rights assigned to users and roles for items in the content tree. See the Sitecore Access Viewer documentation and the Sitecore security administration documentation for details on evaluating and managing item security.

QUESTION NO: 3

A developer needs to configure a rendering in order to use dynamic placeholders. Which of the following steps is required? Select all that apply.

A. Include the IDynamicPlaceholder base template in the Rendering Parameters template.

B. Link the placeholder settings item to the rendering item.

C. Define the placeholder key using an asterisk (*) wildcard in the placeholder settings item.

D. In the component TSX file, set a unique placeholder key value that has not yet been defined.

ANSWER: A B C

Explanation:

Dynamic placeholders allow a rendering to contain placeholders whose keys are made unique for each rendering instance, so authors can add components independently within repeated or nested component structures. To enable this behavior, the rendering parameters template must inherit from the `IDynamicPlaceholder` base template. This supplies the rendering-level configuration required by Sitecore to recognize the rendering as supporting dynamic placeholders. The placeholder settings item must also be linked to the rendering item, associating the allowed placeholder configuration with that component. Finally, the placeholder settings key must use the asterisk wildcard (`*`) to represent the dynamic portion of the generated placeholder key. The wildcard enables Sitecore to match the unique runtime keys produced for individual instances while applying the same placeholder settings, including allowed controls and authoring restrictions. Together, these settings make the rendering eligible for dynamic-placeholder operation and ensure its generated placeholder keys resolve to the appropriate settings item. See the Sitecore JSS/XM Cloud documentation at Sitecore XM Cloud Developer Documentation and the broader Sitecore Developer Portal.

QUESTION NO: 4

A solution uses Sitecore Content Serialization to deploy implementation items through XM Cloud. Content authors continuously update page content and promotional data sources in the production environment. Which items are appropriate to treat as developer-managed and include in serialization? Select three.

A. Data templates

B. Rendering definitions

C. Published page content authored in production

D. Placeholder settings

E. Promotional data source items maintained by authors

ANSWER: A B D

Explanation:

Data templates, **rendering definitions**, and **placeholder settings** are implementation artifacts that define the solution's content model and presentation configuration. They are normally maintained in source control and deployed consistently as part of the solution.

Pages and author-managed promotional data sources are editorial content in this scenario. Serializing them as developer-managed items risks overwriting author changes during deployment. The ownership model should distinguish implementation items from content owned and maintained by authors.

QUESTION NO: 5

A developer is working on managing environments within the XM Cloud Deploy app. They have created a new environment for the project and linked it to a specific repository branch. However, they realize that they need to change the linked repository branch due to new developments in the project. Which steps should they follow to achieve this?

A. Delete the current environment and create a new one with the desired repository branch.

B. Unlink the current repository branch and then relink the desired branch to the environment through the project's "Options" menu.

C. Link the desired branch to the environment directly from the repository settings. The XM Cloud Deploy app will automatically update the linked branch.

D. Go to the project page, click the environment, choose "Options," and then "Edit environment details." In the dialog, select the desired branch from the "Link to branch" drop-down menu and save the changes.

ANSWER: D

Explanation:

Go to the project page, click the environment, choose "Options," and then "Edit environment details." In the dialog, select the desired branch from the "Link to branch" drop-down menu and save the changes. This is the supported XM Cloud Deploy workflow because the repository-branch association is an editable property of an existing environment. Opening the environment first ensures the change applies to the intended deployment target within the project. The *Edit environment details* dialog provides the *Link to branch* control, where the developer can choose the repository branch that should subsequently be associated with that environment. Selecting *Save* persists the updated environment configuration. This lets teams redirect an environment to the appropriate development, feature, release, or main branch while retaining the environment and its existing configuration rather than recreating it. Sitecore documents environment administration, including editing environment details and linking an environment to a branch, in its XM Cloud developer documentation. See the official Manage an environment guide and the XM Cloud developer documentation for the relevant Deploy app procedures.

QUESTION NO: 6

If a developer wants to limit the serialization of items under a Navigation item to just the item itself and then one step below, what property should the developer add to the includes to indicate this limitation?

A. Scope

B. Nothing—this is the default.

C. Limit

D. Name

ANSWER: A

Explanation:

Scope is the property used in a Sitecore serialization include to define how far serialization extends from the configured item path. An include identifies the starting item with its `path`, while `scope` determines whether serialization covers only that item, the item and its direct children, or its complete descendant hierarchy. To include a Navigation item and exactly one level below it, the developer configures the include's `scope` with the value that represents the item and its children (commonly `ItemAndChildren`, depending on the serialization configuration format and version). This keeps the Navigation item and its immediate child items under source control without serializing deeper levels. Defining scope explicitly is useful because it makes the intended serialization boundary clear and prevents an unbounded navigation subtree from being included accidentally. Sitecore's YAML serialization documentation describes how item include definitions use paths and scopes to control serialized item coverage. See the Sitecore YAML serialization format documentation and the Sitecore Serialization project for configuration concepts and implementation details.

QUESTION NO: 7

A developer needs to create a site for a company and must define the data structures in Sitecore to create items and content. What should the developer use to define the data structures?

A. Templates

B. Rendering parameters

C. Renderings

D. Components data sources

ANSWER: A

Explanation:

Templates are the Sitecore mechanism used to model and enforce the structure of content items. A data template defines the fields that an item contains, organizes those fields into sections for authoring, and specifies each field's type, such as single-line text, rich text, image, link, date, or a reference to another item. Sitecore associates every content item with a template, so the template establishes the item's schema and the editing experience presented to content authors.

Templates also support inheritance through base templates. This lets a development team define shared fields and standard behavior once, then reuse them across multiple content types. For example, common metadata, navigation properties, and SEO fields can be inherited by page, article, product, or location templates while each content type adds its own specific fields. In an XM Cloud solution, this content-modeling capability is fundamental because items and component data-source items need a consistent, reusable schema before authors can enter content or components can consume it.

Sitecore's documentation describes templates as the definitions that control item fields and item structure. See Sitecore templates documentation and XM Cloud data templates documentation.