Free Questions for S90.09 by certsdeals

Shared by Hubbard on 24-05-2024

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

Services A, B, and C are non-agnostic task services. Service A and Service B use the same shared state database to defer their state data at runtime. An assessment of these three services reveals that each contains some agnostic logic, but because it is bundled together with the non-agnostic logic, the agnostic logic cannot be made available for reuse. The assessment also determines that because Service A and Service B and the shared state database are each located in physically separate environments, the remote communication required for Service A and Service B to interact with the shared state database is causing an unreasonable decrease in runtime performance.

How can the application of the Orchestration pattern improve this architecture?

Options:

A- The application of the Orchestration pattern will result in an environment whereby the State Repository and Service Data Replication patterns are naturally applied, allowing the shared state database to be replicated for Services A and B so that each task service can have its own dedicated state database. The Process Centralization pattern can also be applied to Services A and B, so that their logic is physically centralized, turning them into orchestrated task services.

B- The application of the Orchestration pattern will result in an environment whereby the Process Abstraction and Process Centralization patterns are naturally applied to Services A, B, and C, resulting in a clean separation of non-agnostic task services from newly designed agnostic services with reuse potential. Also, the State Repository pattern can be applied by the availability of a central state database that can be shared by Services A and B . This database can be made available as a local part of the environment so that Services A and B can avoid remote communication.

C- The application of the Orchestration pattern will result in an environment whereby the Compensating Service Transaction is naturally applied, resulting in the opportunity to create sophisticated exception logic that can be used to compensate for the performance problems caused by Services A and B having to remotely access the state database. The Process Abstraction and Service Broker patterns are also naturally applied, enabling the separation of nonagnostic logic and agnostic logic while providing common transformation functions required to overcome any disparity in the service contracts that will need to be created for the new agnostic services.

D- None of the above.

Answer:

B

Question 2

Question Type: **MultipleChoice**

Services A, B, and C are non-agnostic task services. Service A and Service B use the same shared state database to defer their state data at runtime. An assessment of these three services reveals that each contains some agnostic logic, but because it is bundled together with the non-agnostic logic, the agnostic logic cannot be made available for reuse. The assessment also determines that because Service A and Service B and the shared state database are each located in physically separate environments, the remote communication required for Service A and Service B to interact with the shared state database is causing an unreasonable decrease in runtime performance.

You are asked to redesign this architecture in order to increase the opportunity for agnostic service logic to be reused and in order to decrease the runtime processing demands so that performance can be improved. What steps can be taken to achieve these goals?

Options:

A- The Enterprise Service Bus pattern can be applied to establish an environment whereby the Process Abstraction and Process Centralization patterns are naturally applied, resulting in a clean separation of non-agnostic task services from newly designed agnostic services that are further shaped into reusable services by the application of the Service Reusability principle.

B- The Process Centralization pattern can be applied, resulting in a redesign effort where agnostic logic is removed from the three task services so that they only encapsulate non-agnostic logic. The agnostic logic is then moved to one or more new agnostic services that are shaped into reusable services by the application of the Service Reusability principle. The Process Abstraction pattern is then applied to the redesigned task services Service A and Service B, so that their logic is physically centralized, turning them into orchestrated task services.

C- The Process Abstraction pattern can be applied, resulting in a redesign effort where agnostic logic is removed from the three task services so that they only encapsulate non-agnostic logic. The agnostic logic is then moved to one or more new agnostic services that are shaped into reusable services by the application of the Service Reusability principle. The Orchestration pattern can be further applied to establish an environment whereby the Process Centralization pattern is naturally applied to Services A and B and the State Repository pattern in naturally applied to further help avoid remote communication by providing a local and centralized state database that can be shared by both services.

D- None of the above.

Answer:

C

Question 3

Question Type: MultipleChoice

Service Consumer A sends a message to Service A (1), which then forwards the message to Service B (2). Service B forwards the message to Service C (3), which finally forwards the message to Service D (4). Services A, B, and C each contain logic that reads the content of the message and, based on this content, determines which service to forward the message to. As a result, what is shown in the Figure is one of several possible runtime scenarios.

Currently, this service composition architecture is performing adequately, despite the number of services that can be involved in the transmission of one message. However, you are told that new logic is being added to Service A that will require it to compose one other service in order to retrieve new data at runtime that Service A will need access to in order to determine where to forward the message to. The involvement of the additional service will make the service composition too large and slow. What steps can be taken to improve the service composition architecture while still accommodating the new requirements and avoiding an increase in the amount of service composition members?

Options:

A- The Rules Centralization pattern can be applied to establish a centralized service that contains routing-related business rules. This new Rules service would replace Service B and could be accessed by Service A and Service C in order for Service A and Service C to determine where to forward a message to at runtime. The Service Composability principle can be further applied to ensure that all

remaining services are designed as effective service composition participants.

B- The Asynchronous Queuing pattern can be applied together with the Rules Centralization pattern to establish a Rules service that encapsulates a messaging queue. This new Rules service would replace Service B and could be accessed by Service A and Service C in order for Service A and Service C to determine where to forward a message to at runtime. The Service Composability principle can be further applied to ensure that all remaining services are designed as effective service composition participants.

C- The Intermediate Routing pattern can be applied together with the Service Agent pattern by removing Service B or Service C from the service composition and replacing it with a service agent capable of intercepting and forwarding the message at runtime based on pre-defined routing logic. The Service Composability principle can be further applied to ensure that all remaining services are designed as effective service composition participants.

D- None of the above.

Answer:

C

Question 4

Question Type: MultipleChoice

Service Consumer A sends a message to Service A (1), which then forwards the message to Service B (2). Service B forwards the message to Service C (3), which finally forwards the message to Service D (4). Services A, B, and C each contain logic that reads the content of the message and, based on this content, determines which service to forward the message to. As a result, what is shown in

the Figure is one of several possible runtime scenarios.

You are told that the current service composition architecture is having performance problems because of two specific reasons. First, too many services need to be explicitly invoked in order for the message to arrive at its destination. Secondly, because each of the intermediary services is required to read the entire message contents in order to determine where to forward the message to, it is taking too long for the overall task to complete. What steps can be taken to solve these problems without sacrificing any of the functionality that currently exists?

Options:

A- The Intermediate Routing pattern can be applied together with the Service Agent pattern in order to establish a set of service agents capable of intercepting and forwarding the message based on pre-defined routing logic. To avoid the need for service agents to read the entire message contents, the Messaging Metadata pattern can be applied so that content relevant to the routing logic is placed in the header of a message. This way, only the message header content needs to be read by the service agents.

B- The Intermediate Routing pattern can be applied together with the Service Agent pattern in order to establish a set of service agents capable of intercepting and forwarding the message based on pre-defined routing logic. To avoid the need for service agents to read the entire message contents, the Rules Centralization pattern can be applied so that content relevant to the routing logic is isolated into a separate Rules service. This way, service agents are only required to access the Rules service in order to determine where to forward messages to. The Standardized Service Contract principle will need to be applied to ensure that the new Rules service and the new service agents provide service contracts that are compliant to existing design standards.

C- The Intermediate Routing pattern can be applied together with the Service Agent pattern in order to establish a set of service agents capable of intercepting and forwarding the message based on pre-defined routing logic. The Service Discoverability principle can be applied to improve the communications quality of message contents, which will reduce the time required by service agents to read the message contents at runtime.

D- None of the above.

Answer:

A

Question 5

Question Type: MultipleChoice

Service A is an entity service that provides a Get capability that returns a data value that is frequently changed. Service Consumer A invokes Service A in order to request this data value (1). For Service A to carry out this request, it must invoke Service B (2), a utility service that interacts (3.4) with the database in which the data value is stored, Regardless of whether the data value changed. Service B returns the latest value to Service A (5), and Service A returns the latest value to Service Consumer A (6). The data value is changed when the legacy client program updates the database (7). When this change happens is not predictable. Note also that Service A and Service B are not always available at the same time. Any time the data value changes. Service Consumer A needs to receive it as soon as possible. Therefore, Service Consumer A initiates the message exchange shown in the Figure several times a day. When it receives the same data value as before, the response from Service A is ignored. When Service A provides an updated data value, Service Consumer A can process it to carry out its task.

Because Service A and Service B are not always available at the same times, messages are getting lost and several invocation attempts by Service Consumer A fail. What steps can be taken to solve this problem?

Options:

A- The Asynchronous Queuing pattern can be applied so that messaging queues are established between Service A and Service B and between Service Consumer A and Service A . This way, messages are never lost due to the unavailability of Service A or Service B .

B- The Asynchronous Queuing pattern can be applied so that a messaging queue is established between Service A and Service B . This way, messages are never lost due to the unavailability of Service A or Service B . The Service Agent pattern can be further applied to establish a service agent that makes a log entry and issues a notification when re-transmission attempts by the messaging queue exceeds a pre-determined quantity.

C- The Asynchronous Queuing pattern can be applied so that a messaging queue is established between Service Consumer A and Service A . This way, messages are never lost due to the unavailability of Service A or Service B .The Service Agent pattern can be further applied to establish a service agent that makes a log entry each time a runtime exception occurs.

D- None of the above.

Answer:

A

Question 6

Question Type: **MultipleChoice**

Service A is an entity service that provides a Get capability that returns a data value that is frequently changed. Service Consumer A invokes Service A in order to request this data value (1). For Service A to carry out this request, it must invoke Service B (2), a utility service that interacts (3.4) with the database in which the data value is stored. Regardless of whether the data value changed, Service B returns the latest value to Service A (5), and Service A returns the latest value to Service Consumer A (6). The data value is changed when the legacy client program updates the database (7) When this change happens is not predictable. Note also that Service A and Service B are not always available at the same time. Any time the data value changes. Service Consumer A needs to receive it as soon as possible. Therefore, Service Consumer A initiates the message exchange shown in the Figure several times a day. When it receives the same data value as before, the response from Service A is ignored. When Service A provides an updated data value, Service Consumer A can process it to carry out its task.

The current service composition architecture is using up too many resources due to the repeated invocation of Service A by Service Consumer A and the resulting message exchanges that occur with each invocation. What steps can be taken to solve this problem?

Options:

A- The Event-Driven Messaging pattern can be applied by establishing a subscriber-publisher relationship between Service A and Service B . This way, every time the data value is updated, an event is triggered and Service B, acting as the publisher, can notify Service A, which acts as the subscriber. The Asynchronous Queuing pattern can be applied between Service A and Service B so that the event notification message sent out by Service B will be received by Service A, even when Service A is unavailable.

B- The Event-Driven Messaging pattern can be applied by establishing a subscriber-publisher relationship between Service Consumer A and Service A . This way, every time the data value is updated, an event is triggered and Service A, acting as the publisher, can notify Service Consumer A, which acts as the subscriber. The Asynchronous Queuing pattern can be applied between Service Consumer A and Service A so that the event notification message sent out by Service A will be received by Service Consumer A, even when Service Consumer A is unavailable.

C- The Asynchronous Queuing pattern can be applied so that messaging queues are established between Service A and Service B and between Service Consumer A and Service A . This way, messages are never lost due to the unavailability of Service A or Service B .

D- None of the above.

Answer:

D

Question 7

Question Type: **MultipleChoice**

Service A is an entity service that provides a set of generic and reusable service capabilities. In order to carry out the functionality of any one of its service capabilities, Service A is required to compose Service B (1) and Service C (2) and Service A is required to access Database A (3), Database B (4), and Database C (5). These three databases are shared by other applications within the IT enterprise. All of service capabilities provided by Service A are synchronous, which means that for each request a service consumer makes. Service A is required to issue a response message after all of the processing has completed. Depending on the nature of the service consumer request, Service A may be required to hold data it receives in memory until its underlying processing completes. This includes data it may receive from either Service A or Service B or from any of the three shared databases. Service A is one of many entity services that reside in a highly normalized service inventory. Because Service A provides agnostic logic, it is heavily reused and is currently part of many service compositions.

You are told that Service A has recently become unstable and unreliable. The problem has been traced to two issues with the current service architecture. First, Service B, which is also an entity service, is being increasingly reused and has itself become unstable and unreliable. When Service B fails, the failure is carried over to Service A . Secondly, shared Database B has a complex data model. Some of the queries issued by Service A to shared Database B can take a very long time to complete. What steps can be taken to solve these problems without compromising the normalization of the service inventory?

Options:

A- The Redundant Implementation pattern can be applied to Service A, thereby making duplicate deployments of the service available. This way, when one implementation of Service A is too busy, another implementation can be accessed by service consumers instead. The Service Data Replication pattern can be applied to establish a dedicated database that contains an exact copy of the data from shared Database B that is required by Service A .

B- The Redundant Implementation pattern can be applied to Service B, thereby making duplicate deployments of the service available. This way, when one implementation of Service B is too busy, another implementation can be accessed by Service A instead. The Service Data Replication pattern can be applied to establish a dedicated database that contains an exact copy of the data from shared Database B that is required by Service A .

C- The Redundant Implementation pattern can be applied to Service B, thereby making duplicate deployments of the service available. This way, when one implementation of Service B is too busy, another implementation can be accessed by Service A instead. The Service Data Replication pattern can be applied to establish a dedicated database that contains a copy of the data from shared Database B that is required by Service A . The replicated database is designed with an optimized data model in order to improve query execution performance.

D- None of the above.

Answer:

C

Question 8

Question Type: MultipleChoice

Service A is an entity service that provides a set of generic and reusable service capabilities. In order to carry out the functionality of any one of its service capabilities, Service A is required to compose Service B (1) and Service C (2) and Service A is required to access Database A (3), Database B (4), and Database C (5). These three databases are shared by other applications within the IT enterprise. All of service capabilities provided by Service A are synchronous, which means that for each request a service consumer makes. Service A is required to issue a response message after all of the processing has completed. Depending on the nature of the service consumer request, Service A may be required to hold data it receives in memory until its underlying processing completes. This includes data it may receive from either Service A or Service B or from any of the three shared databases. Service A is one of many entity services that reside in a highly normalized service inventory. Because Service A provides agnostic logic, it is heavily reused and is currently part of many service compositions.

You are told that Service A has recently become unstable and unreliable and several of the service consumers that access it have had to raise runtime exceptions due to these problems. What steps can be taken to solve these problems without compromising the normalization of the service inventory?

Options:

A- The Service Autonomy principle can be applied to increase the physical isolation of Service A and to reduce dependencies Service A has on external resources. In support of this, the Service Data Replication pattern can be applied in order to establish a dedicated database that contains replicated data from shared Databases A, B, and C . Furthermore, the Redundant Implementation pattern can be applied so that the logic Service A requires from Services B and C can be redundantly placed inside of Service A . This way, Service A avoids having to separately compose Services B and C

B- The Service Statelessness principle can be applied with the help of the State Repository pattern in order to establish a state database that Service A can use to defer state data it may be required to hold for extended periods. The Service Autonomy principle can also be applied in order to increase the physical isolation of Service A and to reduce dependencies Service A has on external resources. In support of this, the Service Data Replication pattern can be applied in order to establish a dedicated database that contains replicated data from shared Databases A, B, and C .

C- The Service Loose Coupling and Standardized Service Contract principles can be applied by introducing a separate utility service that provides centralized data access to the Databases A, B, and C, and exposes a standardized service contract that can be used by Service A . This will prevent Service A from direct dependencies on the shared databases in case any of them are replaced in the future. By following this approach, the Legacy Wrapper pattern is effectively applied via the introduction of the new utility service.

D- None of the above.

Answer:

B

Question 9

Question Type: **MultipleChoice**

Our service inventory contains the following three services that provide invoice-related data access capabilities: Invoice, InvProc, and Proclnv. These services were created at different times by different project teams and were not required to comply to any design standards. Therefore each of these services has a different data model for representing invoice data. Currently each of these three services has one service consumer: Service Consumer A accesses the Invoice service(1). Service Consumer B (2) accesses the InvProc service, and Service Consumer C (3) accesses the Proclnv service. Each service consumer invokes a data access capability of an invoice-related service, requiring that service to interact with the shared accounting database that is used by all invoice-related services (4, 5, 6). Additionally, Service Consumer D was designed to access invoice data from the shared accounting database directly (7). (Within the context of this architecture. Service Consumer D is labeled as a service consumer because it is accessing a resource that is related to the illustrated service architectures.)

A project team recently proclaimed that it has successfully applied the Contract Centralization pattern to the service inventory in which the Invoice service, InvProc service, and ProcInv service reside. Upon reviewing the previously described architecture you have doubts that this is true. After voicing your doubts to a manager, you are asked to provide specific evidence as to why the Contract Centralization is not currently fully applied. Which of the following statements provides this evidence?

Options:

A- The Contract Centralization pattern is not fully applied to the Invoice, InvProc, and ProcInv services because they are being accessed by different service consumers and because they have redundant logic that introduces denormalization into the service inventory.

B- The Contract Centralization pattern is not fully applied because Service Consumer D is accessing the shared accounting database directly.

C- The Contract Centralization pattern is not fully applied because none of the invoice-related services are carrying out data access logic via a centralized and standardized invoice service. This is primarily because the Standardized Service Contract principle was not consistently applied during the delivery processes of the individual services.

D- None of the above.

Answer:

B