

Quiz 6: Service-Oriented Architecture II

Due Oct 24, 2016 at 11:59pm

Points 100

Questions 3

Available Oct 19, 2016 at 8am - Oct 24, 2016 at 11:59pm 6 days

Time Limit 60 Minutes

This quiz was locked Oct 24, 2016 at 11:59pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	60 minutes	90 out of 100

Score for this quiz: **90** out of 100

Submitted Oct 24, 2016 at 9:43pm

This attempt took 60 minutes.

Question 1

30 / 40 pts

Match the description with the design pattern below.

You Answered

Maintain a global repository of software assets for the enterprise.

Logic centralization

Correct Answer

Enterprise inventory

Correct!

Maintain repositories of domain-specific software assets in organizational subunits.

Domain inventory

Correct!

Services should not overlap in the functionality that they provide to clients.

Service normalization

You Answered

Control of reusable software assets should be centralized, with a service custodian responsible for coordinating updates and modifications to the assets.

Enterprise inventory

Correct Answer

Logic centralization

Correct!

Services should be organized into levels, so that lower-level services provide useful abstractions to higher-level services.

Service layering



Correct!

An SOA should include an enterprise-wide collection of services that provide cross-cutting functionality (e.g. logging, transaction coordination, authentication) for services in all domains.

Utility layer



Correct!

Identify useful and reusable business abstractions for basic services.

Entity abstraction



Correct!

Identify useful and reusable business abstractions for long-lived business processes.

Process abstraction



Correct!

Allow multiple communication paths from clients to a service.

Multi-channel endpoint



Correct!

Services should be made as simple and basic as possible, to enable loose coupling and service reusability.

Service decomposition



Correct!

A basic service should be designed so that it does not matter what application it is executed in.

Agnostic context



Correct!

An operation of a basic service should be designed so that it does not matter what application it is executed in.

Agnostic capability



Correct!

Higher-level application logic should be encapsulated as services, to enable composition with other services.

Non-agnostic context



Correct!

Business rules that enforce enterprise policies should be centralized.

Rules centralization



Correct!

The logic for validating data should be taken out of services and centralized.

Validation abstraction ▼

You Answered

Computationally expensive operations should be off-loaded to a separate host to avoid degrading the performance of a service.

Correct Answer

Distributed capability

You Answered

A client stub should be used to invoke an operation that has been off-loaded to another host.

Correct Answer

Proxy capability

Correct!

A service should be replicated across several hosts, so that the service remains available if some of the hosts fail or become overloaded.

Redundant implementation ▼

Correct!

Parts of a backing data store should be cached at a service, to avoid latency in retrieving data from the backing store, and so that the service is decoupled from heavy loads on the store.

Service data replication ▼

You Answered

A separate state server can provide an API to allow a service to save some or all of its state there.

Distributed capability ▼

Correct Answer

Stateful service

You Answered

Part of a service state may be saved on a client, and sent with a client request to the service.

Correct Answer

State messaging

Correct!

Parts of a service's state may be stored elsewhere from the service in the architecture.

Partial state deferral ▼

Correct!

A service's state may be cached on a separate state server while it is idle.

State repository ▼

Correct!

A task or process service may be redirected from one form of client interface to another.

Channel switching ▼

Question 2

30 / 30 pts

For the example below, is it safe replace the service `foo` operation with the service operation `bar`, or to replace the service operation `bar` with the service operation `foo`? In other words, which of the following assignments is safe?

```
foo=bar;    // Is it safe?  
bar=foo;    // Is it safe?
```

For each case, if it is not safe, explain what will go wrong. If it is safe, explain why it is safe.

```
class ContactInfo { String phone; String email; }  
class ExtContactInfo extends ContactInfo { String url; }
```

```
String foo (ContactInfo c) {  
    return "<contact>" +  
           "<phone>" + c.phone + "</phone>" +  
           "<email>" + c.email + "</email>" +  
           "</contact>";  
}  
  
String bar (ExtContactInfo c) {  
    return "<contact>" +  
           "<phone>" + c.phone + "</phone>" +  
           "<email>" + c.email + "</email>" +  
           "<url>" + c.url + "</url>" +  
           "</contact>";  
}
```

Your Answer:

1. `foo = bar` --> unsafe operation -> In this operation, when `bar` is assigned to `foo`, `bar` has a domain value of `url` that `foo` can not possess. So this assignment is unsafe assignment.

2. `bar=foo` --> safe operation -> `foo` is a subtype of `bar`. So `bar` possesses all the domain types of values which `foo` can possess. So this operation is a safe operation.

Question 3

30 / 30 pts

Consider this O'CAML code:

```
let x : List.t = List.insert 3 List.empty
...
let y : int = List.head x
```

1. For opaque types, where in the execution of this code does the list type cross the abstraction boundary from exposed to opaque?
2. For opaque types, where in the execution of this code does the list type cross the abstraction boundary from opaque to exposed?
3. For cryptographic sealing, where in the execution of this code does the list value get encrypted as ciphertext?
4. For cryptographic sealing, where in the execution of this code does the list value get decrypted back to plaintext?

Your Answer:

1. The result of the insert operation crosses an abstraction boundary from the transparent internal type to the opaque representation type `List.t` when it is returned.
2. The argument to the head operation crosses the abstraction boundary from the opaque to the exposed when it is received by the operation.
3. The result of 'insert' operation is where the list get encrypted as ciphertext.
4. the result of 'head' operation is where the list values get decrypted back to plain text.

Quiz Score: **90** out of 100