EVS

1 Intro

- Main Topics tuwis
  - Remoting und Distribution Patterns
  - Design von Verteilten Systemen
  - Unterschiede bei der Planung und Entwicklung von Verteilten Systemen zu nicht verteilten Systemen
  - Modelle für Verteilte Systeme/Informationssysteme
  - Middleware-Design
  - Concurrency Patterns
  - Distribution Strategies
  - Patterns für Service Oriented Computing

2 Remoting Patterns

2.1 Basic Remoting Patterns

2.1.1 App Layer

Client
Remote Object

2.1.2 Invocation Layer

Client Proxy
- implements Remote Object Interface
- facades Requestor

Requestor
- marshal request

Invoker
- demarshal request
- perform invocation on targeted remote object

2.1.3 Messaging Layer

Server Request Handler
- deals with communication issues
- manages connections, threads, ...

Client Request Handler
- connection handling, timeouts, ...

2.1.4 More

Marshaller
Interface Description
- WSDL

Remoting Error
- distinguish distribution & application errors

2.2 Identification Patterns

2.2.1 Object ID

invoker context unique id

2.2.2 Absolute Object Reference

- identifies invoker and remote object

2.2.3 Lookup

- associate references with properties (names)

2.3 Lifecycle Petterns

2.3.1 Static Instance

- independent of client's state & lifecycle
- activation and lookup-registration typically at application startup

2.3.2 Per-Request Instance

- Scalability, Synchronization & Performance
- Servant instanciated by Invoker for each request

2.3.3 Client-Dependent Instance

- extend client application logic
- lifetime controlled by client - create, destroy
- state may be private

2.3.4 Lazy Acquisition

- manage static instances
- resource waste at startup, startup time

2.3.5 Pooling

- instanciating and destroying causes overhead
- pooled servant: take from pool, initialize, handle request, back to pool

2.3.6 Leasing

- releases resources no longer needed
- client-dependent instances expire (lifecycle manager)
- client may renew lease by operation call & explicitly

2.3.7 Passivation

- passivate stateful remote objects

2.4 Extension Patterns

2.4.1 Invocation Context

- plug-in add-on services without changing operation signatures
- bundle context information
- invocation interceptors add & consume information

2.4.2 Invocation Interceptor

- transparently integrates add-on services
- for transactions, logging, security
- chained hooks before/after request/response

2.4.3 Protocol Plug-In

- multiple/specialized/optimized communication protocols
- extend client/server request handlers

2.5 Extended Infrastructure Patterns

2.5.1 Lifecycle Manager

- handle lifecycle of remote objects
- implements lifecycle strategy based on configuration, resources...
- triggers lifecycle operations (instanciate, initialize, destroy)

2.5.2 Configuration Group

- config per group of remote objects
- more flexible - not only per server / per object
- for QoS, interception, lifecycle mgmt, protocol support, ...

2.5.3 Local Object

- behave like remote objects, but cannot be accessed remotely
- allow to access middleware configuration and status

2.5.4 QoS Observer

- observe QoS parameters like message sizes, invocation counts, execution times, ...
- hooks in middleware allow for registering QoS Observers
- observer may react on QoS data

2.5.5 Location Forwarder

- remote object not at receiving server (moved, load balancing, ...)
- forward invocation to remote object on other server application
- maps object id to absolute object references

2.6 Asynchronous Invocation Patterns

2.6.1 Fire and Forget

- client simply notifies remote object of an event
- no ack, return value or exeption handling
- request sent async

2.6.2 Sync with Server

- extends Fire and Forget reliability
- client request handler waits for ack
- invocation call is still performed async

2.6.3 Poll Object

- result expected but not needed immediately
- requestor performs async invocation
- invoker stores result in poll object
- client queries or blocks on poll object to get the result

2.6.4 Result Callback

- result expected and client needs to react immediately
- client passes Callback Object which is executed by Invoker on result

3 Prüfung

3.1 CharakteristikaUnterschiede

3.1.1 Lifecycle PatternsCharakteristikaUnterschiede

- Static Instance
- Per-request Instance
- Client-dependent Instance
- Lazy Aquisition
- Pooling
- Leasing
- Passivation

3.1.2 Asynchrony PatternsCharakteristikaUnterschiede

- Poll Object
- Result Callback
- Fire and Forget
- Sync with server

3.2 SkizzeTechnologieBeispiel

3.2.1 Invocation Interceptor Pattern

- Skizze
- Technologie Beispiel

3.2.2 Protocol-Plugin Pattern

- Skizze
- Technologie Beispiel

3.2.3 Invocation Context Pattern

- Skizze
- Technologie Beispiel

3.2.4 Server-Request-Handler Pattern

- Skizze
- Technologie Beispiel

3.2.5 Poll Object Pattern

- Technologie Beispiel
- Skizze

3.3 FolieBeziehungen

3.3.1 Invocation Asynchrony Patterns

- Abbildung > Beziehungen benennen
- Kurzbeschreibung
- Folie 168

3.3.2 Lifecycle Management Patterns

- Kurzbeschreibung
- Abbildung > Beziehungen benennen
- Folie 98

3.4 Einzel

- 09-06-16
  - detail
    - 1 Beschreiben Sie die in der Vorlesung vorgestellten Lifecycle Patterns und heben Sie die Charakteristika und Unterschiede hervor: Static Instance, Per-request Instance, Client-dependent Instance, Lazy Aquisition, Pooling, Leasing, Passivation.
    - 2 Erklären Sie das Pattern Invocation Interceptor mit Skizze und einem (schematischen/beschreibenden) Technologie Beispiel.
    - 3 Erklären Sie das Pattern Protocol-Plugin mit Skizze und einem (schematischen/beschreibenden) Technologie Beispiel.
    - 4 Beschreiben Sie die angegebenen Patterns (in einem Satz) und bezeichnen Sie die Beziehung zwischen den Patterns in der Abbildung
  - Lifecycle PatternsCharakteristikaUnterschiede
    - Static Instance
    - Per-request Instance
    - Client-dependent Instance
    - Lazy Aquisition
    - Pooling

        - Leasing
        - Passivation
    - Invocation Interceptor Pattern
        - Skizze
        - Technologie Beispiel
    - Protocol-Plugin Pattern
        - Skizze
        - Technologie Beispiel
    - Invocation Asynchrony Patterns
        - Abbildung > Beziehungen benennen
        - Kurzbeschreibung
        - Folie 168
- 09-07-06
    - detail
        - 1 Beschreiben Sie die in der Vorlesung vorgestellten Asynchrony Patterns und heben Sie die Charakteristika und Unterschiede hervor: Poll Object, Result Callback, Fire and Forget, Sync with server.
        - 2 Erklären Sie das Pattern Invocation Context mit Skizze und einem (schematischen/beschreibenden) Technologie Beispiel.
        - 3 Erklären Sie das Pattern Server-Request-Handler mit Skizze und einem (schematischen/beschreibenden) Technologie Beispiel.
        - 4 Beschreiben Sie die angegebenen Patterns (in einem Satz) und bezeichnen Sie die Beziehung zwischen den Patterns in der Abbildung
    - Asynchrony PatternsCharakteristikaUnterschiede
        - Poll Object
        - Result Callback
        - Fire and Forget
        - Sync with server
    - Invocation Context Pattern
        - Skizze
        - Technologie Beispiel
    - Server-Request-Handler Pattern
        - Skizze
        - Technologie Beispiel
    - Lifecycle Management Patterns
        - Kurzbeschreibung
        - Abbildung > Beziehungen benennen
        - Folie 98
- 09-09-15
    - detail
        - 1 Beschreiben Sie die in der Vorlesung vorgestellten Lifecycle Patterns und heben Sie die Charakteristika und Unterschiede hervor: Static Instance, Per-request Instance, Client-dependent Instance, Lazy Aquisition, Pooling, Leasing, Passivation.
        - 2 Erklären Sie das Pattern Invocation Interceptor mit Skizze und einem (schematischen/beschreibenden) Technologie Beispiel.
        - 3 Erklären Sie das Pattern Protocol-Plugin mit Skizze und einem (schematischen/beschreibenden) Technologie Beispiel.
        - 4 Beschreiben Sie die angegebenen Patterns (in einem Satz) und bezeichnen Sie die Beziehung zwischen den Patterns in der

    - Abbildung
- Lifecycle PatternsCharakteristikaUnterschiede
    - siehe 09-06-16
- Poll Object Pattern
    - Technologie Beispiel
    - Skizze
- Invocation Context Pattern
    - siehe 09-07-06
- Asynchrony PatternsCharakteristikaUnterschiede
    - siehe 09-07-06