An Approach to Composition of EJB Components Using C2 style

Dept. of Software·Contents Technology Research
Computer & Software Laboratory
Electronics and Telecommunications Research Institute
You-Hee Choi (yhchoi@etri.re.kr)
Organization

- Introduction
- The C2 architectural style
- Our approach
- Case Study
- Conclusion
In CBSD,

- The notion of building a system by writing code
  - has been replaced with building a system by assembling existing software components into multiple applications

The component model, Enterprise JavaBeans (EJB) follows the "Write Once, Run Anywhere" philosophy of the Java programming language

- But it is difficult to compose components provided by third parties without recompilation or a source code modification in the absence of mediators.
  - Due to the need of a direct method invocation from the standpoint of a client for communicating with other components.

The component composition approach is needed

- To compose various EJB components provided by third-parties and reconfigure flexibly an architecture through the plug-and-play technique.
Introduction (2/2)

- In order to compose EJB components through the plug-and-play technique,
  - It is necessary to generate a mediator which handles mismatches of interfaces and interactions between components
    - Software architecture concept

- For flexible reconfiguration when components are being composed,
  - A communication method by an indirect message passing rather than a direct method invocation is required
    - The C2 architectural style
The C2 architectural style

- Message-based layered architectural style

- Two types of building blocks: components and connectors
- Connectors act as message routing devices.
- Components and connectors both have a defined top and bottom.
Our approach

Component composition approach for EJB

- **Purpose**
  - To create a composite EJB for a C2 architecture which is composed of EJB components provided by third parties through the plug-and-play technique
  - To compose EJB components without modification through the plug-and-play technique based on the C2 style,
    - Direct method invocations of EJB components should be replaced with indirect message passing
      » Generate an EJB wrapper class by a wrapping mechanism, which has the logic of message passing instead of method invocations
EJB wrapper for EJB composition

The role of EJB wrappers

- To communicate with components
  - By the logic of message passing instead of the logic of method invocation
- To invoke methods of EJB components according to received messages

The logic of message passing

- Enables to send a message to the target component which is not explicitly specified, and to convert the message to the message the target component requires.
EJB wrapper for EJB composition

- The representative form of the message handling logic

```
if (message_received)
{
    //get parameters
    ..............
    //invoke methods of EJB
    ..............
    //create requests/notifications
    ..............
    //send messages
    ..............
}
```
Generating a composite EJB component

- The relationship of the C2 architecture which is composed of various EJB components and a composite EJB.
Generating a composite EJB component

- A new C2 component which links the C2 architecture with the composite EJB is needed
  - The role of initializing the C2 architecture to send messages, creating and sending requests, and receiving notifications

```java
//remote method
protected Object remoteMethod1() {
    ........
    Request new_r = new Request ("request1");
    ........
    send (new_r);
    return return_value;
}
protected void handle(Message n) {
    // handle message
}
```
Case Study

A simple example using the component-assembly tool called COBALT (COmponent-Based Application deVelopment Tool) Assembler that has been developed for our approach

- The tool
  - Enables component reusers to assemble EJB components through the plug-and-play technique
  - Generates EJB wrappers and a composite EJB automatically based on component specifications described through a specification editor

- The example system
  - Is a simple Internet shopping mall system
Case Study

- The architecture that is composed of 8 EJB components
The message flow for registering an order:

1. Register an order information
2. Register a payment information
3. Registered the payment information
4. Registered the order information
Case Study

```java
message_name = r.getName();

if (message_name.equalsIgnoreCase("registerBuyerOrder\etrie.shoppingmall.

try
{
    OrderInfo par_OrderInfo_oInfo =\etrie.shoppingmall.order.model.OrderInfo
    PaymentInfo par_PaymentInfo_pInfo =\etrie.shoppingmall.payment.model.PaymentInfo
    boolean par_boolean_registerOrderResult =\etrie.shoppiingmall.payment.model
    OrderSession.registerOrderResult = _OrderSession.getResult();

    if(par_boolean_registerOrderResult)
    {
        Message new_0 = new Message ("Request", "registerBuyerOrder\etrie.
        addParameter ("pInfo", par_PaymentInfo_pInfo);
        send(new_0);
    }
}

catch (Exception e)
{
    e.printStackTrace();
    throw e;
}
```
Case Study

- Generating a composite EJB
Case Study
Conclusions

- **An approach to composition of EJB components based on the C2 style**
  - By generating a composite EJB
    - Allows component reusers to build a new function by composing prebuilt EJB components
  - By generating an EJB wrapper
    - Various EJB components can be composed without recompilation or source code modification

- **Our research work includes**
  - Redefinition of C2 ADL (Architecture Description Language) for EJB
  - The development of EJB component-assembly tool called COBALt (COnponent-Based Application deveLopment Tool) assembler that is the visual composition tool to support our approach.