유연성있는 분산 어플리케이션 개발을 위한 동적 서비스 관리 프레임 워크

이용환*, 염귀덕*, 안형근*, 민덕기*, 장진호*
*건국대학교 컴퓨터 정보통신공학과
(yhlee, kdyeom, h96ahg, dkmin)@konkuk.ac.kr
jhjang@etri.re.kr

Dynamic Service Configuration Framework For Flexible Distributed Applications

Young-Han Lee* Ki-Deuk Yeom*, Heongun Ahn*, Dugki Min*, Jinho Jang*
*Dept. of Computer Science and Engineering, Konkuk University
*ETRI

요 약

유연성 있고 확장성 있는 분산 어플리케이션을 작성하기 위해서는 다음과 같은 요건을 만족하는 서비스 개발 및 관리를 위한 프레임워크이 필요하다. 첫번째 서비스들은 외부적인 어떤 정책이나 어플리케이션 구동 시 다양한 제한 등의 요인으로 인하여 선택적으로 서비스 사용이 가능해야 한다. 둘째로 서비스에 대해서 초기화, 구동, 임시정지, 정지 등과 같은 제어를 중앙 집중형태로 관리 콘솔에서 관리 할 수 있어야 한다. 세 번째로 서비스를 제어할 때 각 서비스들 사이의 의존관계를 반영할 수 있어야 한다. 네 번째로 이러한 분산 어플리케이션의 서비스에 대한 환경 설정 부분에 대한 변경을 실시간으로 탐지를 해서 이러한 변경과 관계가 있는 서비스에게 통지를 할 수 있어야 한다. 다섯번째로 한 어플리케이션 안에서 구동 되고 있는 의존관계가 있는 각 서비스들 사이의 이벤트 기반의 통신을 메커니즘이 필요하다. 여섯번째로 환경 설정 변경에 대해서 분산 상황에서 다른 어플리케이션과 동기화를 맞출 수 있어야 한다. 본 논문은 이러한 유연하고 확장성 있는 어플리케이션을 개발 하는데 필요한 자바 기반의 동적 서비스 관리 프레임워크에 대한 설계와 구현을 제시한다.

1. 서 론

Timing Service나 분배 필터링을 위한 이벤트 서비스와 같이 어플리케이션을 개발 시 다양한 종류의 서비스가 필요하다[5][6]. 기존에는 이러한 서비스를 구현 시점에 하나의 어플리케이션이나 운영환경에 정적으로 바인딩 식의 정적 서비스 관리 기법을 사용하였다. 하지만 이러한 정적 관리 방법은 차후에 서비스 구현이나 환경 설정 부분에 대한 변경이 발생하게 될 경우 다시 컴파일하고 링크를 해야 하므로 유연성이 없고 확장성이 없는 분산 어플리케이션을 만들어 내는 근본 원인이 된다[11]. 최근의 분산 어프리케이션들은 요구사항의 변화가 빈번하고, 구동시점에 다양한 타입의 서비스들과 선택적 연동[3][4]을 하여야 하며, 구동되고 있는 서비스들을 중앙 집중 형식으로 제어하는 것을 필요로 한다. 이러한 요구사항을 위해서 GoF Pattern Format[1]의 다양한 종류의 패턴을 사용한 Service 설정을 위한 패턴[2]을 사용하고 있다. 또한 분산환경에서 환경 설정에 대한 변경을 실시간으로 탐지해서 동적으로 환경 설정을 할 수 있어야 하며 각 서비스들 간에 이벤트 기반으로 통신을 할 수 있는 메커니즘이 필요하다[10]. 본 논문에서는 이러한 요구 사항을 만족하는 자바 기반의 객체 지향 서비스 관리 프레임워크을 제시한다. 이 프래임워크는 유연성 있고 확장성 있는 분산 어프리케이션을 개발하는 기본 프래임워크를 제시한다.

2. 동적인 서비스 관리 프래임워크

본 논문에서는 서론부분에서 설명한 서비스 관리 패턴을 이용한 동적 서비스 관리 프레임워크를 제시한다. 이 프래임워크는 응용 서비스의 정적 형상의 변화나 서비스 버전의 증가 및 서비스 운영 중 발생하는 서비스 변화 이벤트에 따른 관련 서비스의 영향 등을 런타임시에 자동으로 해결할 수 있는 관리 프래임 워크이다.

2.1 아키텍처

그림 1은 본 논문에서 제시한 유연성 있고 확장성 있는 분산 어

플리케이션 제작을 위한 동적 서비스 관리 프래임워크의 구조도이다. 위의 아키텍처는 크게 다음과 같은 세 개의 컴포넌트로 나누어진다.

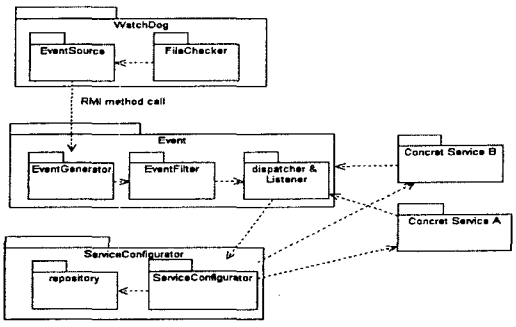

그림 1 Architecture of Dynamic Service Configuration Framework

첫째로, WatchDog 컴포넌트는 자신이 관리해야 할 환경 설정 파일에 대한 정보를 가지고 실시간으로 이 파일에 대해서 변동이 가해졌는지를 주기적으로 검사해서 실제로 이러한 변동 이벤트가 발생했을 때 이벤트 서비스 쪽으로 이벤트 발생한다. 이벤트 발생 요청을 보낼 때에는 해당하는 이벤트의 종류, 환경설정 파일 등과 같은 정보를 넘겨 준다. 둘째로, Event Service는 WatchDog과 같은 다른 Event Source로부터 요청을 받아서 해당하는 Event을 생성한 다음 정책에 따라 필터링 한 후에 Event Dispatcher을 통해서 특정 종류의 이벤트에 관심 있는 Event Listener에게 이러한 이벤트를 넘겨주는 역할을 수행한다. 실제로 이는 하나의 어플리케이션 안에서 서비스들 사이의 통신 메커니즘을 제공하는 하나의 Event Coordinator로서의 역할을 수행한다. 마지막으로, Service Configurator는 서비스에 설정에 관련된 다양한 이벤트에 대한 처리를 수행한다. 또한 분산 어플리케이션 안에서 구동 되고 있는 다양한 종류의 서비스를 중앙 집중형태로 관리하고 있다. 이러한 역할을 수행하는 것이 바로 Service Configurator이며 Repository는 실제 구동 되고 있는 서비스에 대한 모든 리스트를 가지고 있다. Concrete Service는 실제로 시스템이나 어플리케이션 안에서 사용되고 있는 다양한 종류의 서비스를 말한다. 이러한 서비스들은 Service라는 인터페이스가 제공하는 후크 메쏘드 들을 구현하고 있다. 또한 Service Configurator에 의해서 제어를 받는다.

2.2 Design and Implementation

여기서는, WatchDog과 Event Service에 대한 설계 부분은 지면 관계상 생략하고, 핵심 부분인 Service Configurator 컴포넌트의 설계에 대하여 좀 더 자세히 알아본다.. Service Configurator 컴포넌트가 수행하는 역할은 현재 어플리케이션 안에서 구동되고있는 서비스에 대해서 제어 즉 서비스 초기화, 서비스 일시 중지, 서비스 재 시작, 서비스 중지 등과 같은 제어를 수행한다. 다름 그림 2은 이러한 서비스 관리 컴포넌트의 클래스도이다.

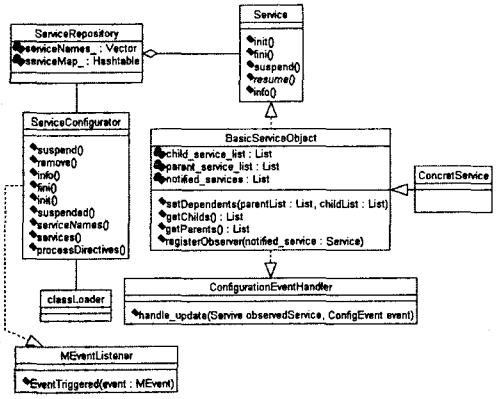

그림 2 Class Diagram of Service Configurator

ClassLoader는 실제로 해당하는 클래스를 찾아서 메모리를 loading하는 역할을 수행하며 Service는 인터페이스 안에는 각 서비스가 초기화, 일시정지, 재가동, 중지 될 때 호출되는 후크 메소드를 가지고 있다. ConfigurationEventHandler는 두 서비스 사이에 의존 관계가 있을 때 한 서비스에 어떤 변동이 생길 때 이때 처리 되야 하는 Listener에 해당하는 부분이다. BasicServiceObject는 ConfigurationEventHandler, Service에 대한 Interface을 구현하고 있으며 이 객체 안에는 이 서비스가 의존하는 서비스 객체에 대한 Collection을 가지고 있다. MevnetListener는 실제로 Event가 발생했을 때 처리해야 할 코드가 들어가는 Listener이다. ConcretService는 실제 구현 서비스이다. 실제로 하나의 서비스가 로딩될 때 init 후크함수를 호출 함으로써 초기화 파라메타를 넘겨 주고 로딩이 성공적으로 수행되었다면 Service Repository안에 보관한다.

그림 3는 서비스 관리 컴포넌트 안에서 실제 서비스에 대한 제어(init, suspend, finish, resume, info) 중에서 init과 finish에 관련된 제어 부분 대한 동적인 과정을 보여 주는 시퀀스도이다. 그림 4는 환경 설정 파일 정보를 바탕으로 해서 기동 된 서비스가 환경설정 파일이 변동 되었을 때 전체적으로 WatchDog과 이벤트 서비스 컴포넌트와 서비스 관리자 컴포넌트 사이에서

어떤 행위를 취하는 지를 보여주는 시퀀스 다이어 그램이다. 만일 관리자가 관리 콘솔 상에서 서비스 환경 설정을 변동했다며 이는 WatchDog에 의해서 실시간으로 탐지가 되며 이벤트 생성기나 Dispatcher에 의해서 ServiceConfigurator에 전해 지면 Repository을 통해서 해당하는 서비스를 참조한 후에 해당하는 서비스에게 명령을 내린다. 또한 서비스 간의 의존성 문제가 있을 경우 Event Generator, Event Dispatcher등을 통해서 서비스 들 사이에 통신을 위한 이벤트 기반의 통신 메커니즘을 제공하고 있다. 실제로 한 서비스가 어떤 다른 Event 의존 관계를 가지고 있는지는 각 서비스 객체가 가지고 있다.

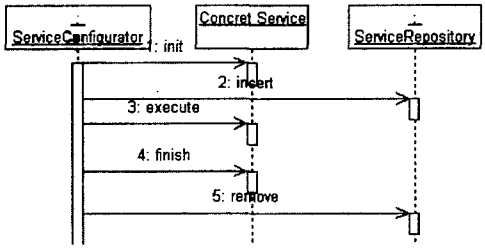

그림 3 Sequnce Diagram for Dynamic Service Control

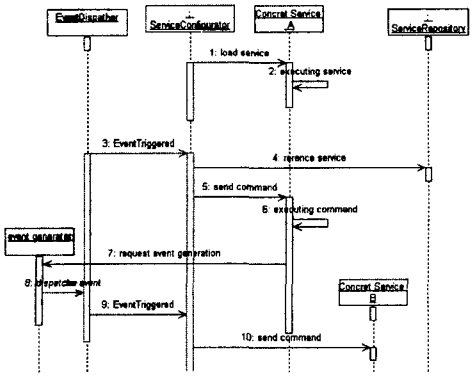

그림 4 Sequence Diagram for Communication Among Services

3.결론

본 논문에서 제공한 프래임워크을 사용하면 WatchDog을 사용해서 실시간으로 환경설정 파일에 대한 변경을 탐지할 수 있다. 또한 Event Generator, Event Filter, Event Dispatcher를 통해서 한 시스템이나 어플리케이션 상에서 서비스들 사이의 커뮤니케이션을 위한 커뮤니케이션 채널을 사용할 수 있다. 그리고 서비스 관리 패턴을 사용 함으로서 어플리케이션 상에서 가동되는 서비스에 대해서 중앙 집중형태로 관리할 수 있으며 또한 서비스 자체의 모듈화와 재 사용성을 좋게 한다. 또한 한 어플리케이션의 성능향상이나 서비스 자체의 동기화 모델을 실제 서비스의 기능과 분리해서 독자적으로 구현할 수 있다는 확장성과 유연성이 뛰어난 서비스 구현이 가능한 프래임웍이다.

4.참고문헌

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns : Elements of Reusable Object-Oriented software. Reading, MA: Addison-Wesley, 1995

[2] Prashan Jain, Dougles C. Schmidt, "Service Configurator pattern for dynamic configuration of Service", Proceeding of the Third USENIX Conference on Object-oriented Technologies and Systems, 1997

[3] R.G. Schmidt and T. Suda, " Active Object: an Object Behavioral Pattern for Concurrent Programming," in pattern language of program Design(J. O. Coplien, J. Vlissides, and N. Kerth, eds), Reading, MA : Addison-Wesley, 1996

[4] D. C. Schemidt, "Reactor: An Object Behavioral Pattern for Concurrent Event De-multiplexing and Event Handler dispatching," in Pattern Languages of Program Design(J. O. Coplien and D.C. Schmidit, etc), Reading, MA: Addison-Wesley, 1995

[5] M. Malowidzki, "Event Filtering: Advanced Event Filtering Approach for CORBA-Based Management System," IEEE, 2000

[6] M. TOMONO, " Event Notification: An Event Notification Framework based on Java and CORBA", IEEE, 2000

[7] Fayed, M.E., Schemidt, D.C., *Object-oriented Application* Framework, Communications of the ACM, vol.40, No.10, pp.32-38, Oct. 1997

[8] Mansouri-Samani, M., Sloman M., A configurable Service for distributed Systems, Proc. of the third International Conference on configurable distributed systems, pp.210-217, May.1996

[10] J. Oueichek and X. Rousset de Pina, "Dynamic Configuration Management in the Guide Object-oriented Distributed System", Proc. of 3rd IEEE Intl. Conf. On Configurable Distributed Systems Annapolis, pp. 28-35,May 1996.

[11] J. Magee and J. Kramer, "Dynamic Structure in Software Architectures", SIGSOFT 96, ACM Software Engineering Notes, Vol 21 No. 6, Nov. 1996.