J2EE 커넥터 아키텍처에서의 리소스 어댑터 배치 방법

°채정화* 김송주* 유철중* 장옥배*
전북대학교 컴퓨터과학과
jhchae@cs.chonbuk.ac.kr
a0050637@cbnu.chonbuk.ac.kr
{cjyoo, okjang}@moak.chonbuk.ac.kr

On Deployment of Resource Adapter in J2EE Connector Architecture

°Jung-Hwa Chae* Song-Ju Kim* Cheol-Jung Yoo* Ok-Bae Chang*
Dept. of Computer Science, Chonbuk National University

요 약

J2EE(Java™ 2 Platform, Enterprise Edition)의 클라이언트 애플리케이션, 웹 컴포넌트, Enterprise JavaBeans 컴포넌트와 같은 엔터프라이즈 애플리케이션은 개발된 후 컨테이너에 배치되어 실행된다. 이러한 엔터프라이즈 애플리케이션(enterprise application)은 다양한 EIS(Enterprise Information System)에 접근하여 관련된 기능과 데이터를 사용할 필요가 있다. J2EE 커넥터 아키텍처(Connector Architecture)는 J2EE 플랫폼을 다양한 EIS와 연결하기 위한 표준을 정의하는 API이며, 각 EIS 벤더는 이러한 커넥터 아키텍처의 설계서를 따르는 EIS의 시스템 레벨 소프트웨어 드라이버인 표준 리소스 어댑터(Resource Adaptor)를 제공한다. 커넥터 아키텍처를 구현한 리소스 어댑터를 사용함으로써 특정 벤더의 J2EE 플랫폼에 종속되지 않도록 EIS를 통합할 수 있다. 본 논문에서는 이러한 J2EE 커넥터 아키텍처를 구현한 리소스 어댑터를 애플리케이션 서버에 배치하기 위하여 배치기가 수행해야 하는 배치 단계를 크게 네 단계로 구분하고, 배치구성 활동을 다섯 가지로 분류 및 정의한다.

1. 서론

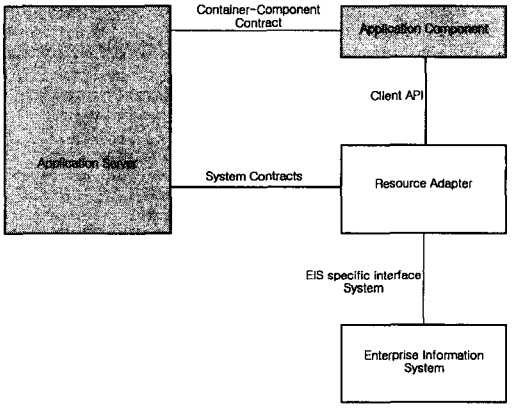
J2EE(Java™ 2 Platform, Enterprise Edition)는 클라이언트 애플리케이션, 웹 컴포넌트, Enterprise JavaBeans 컴포넌트 등의 엔터프라이즈 애플리케이션을 위한 컨테이너를 제공한다[1]. 컨테이너는 애플리케이션 컴포넌트가 배치(Deployment)되고 실행될 수 있도록 하며, 애플리케이션 서버의 일련의 통합 서비스를 제공한다. 이러한 엔터프라이즈 애플리케이션(enterprise application)은 EIS(Enterprise Information System)에 접근하여 관련된 기능과 데이터를 사용할 필요가 있다.

J2EE 커넥터 아키텍처(Connector Architecture)는 J2EE 플랫폼을 다양한 EIS와 연결하기 위한 표준을 정의하는 API이다. EIS 벤더는 이러한 커넥터 아키텍처의 설계서를 따르는 각 EIS의 시스템 레벨 소프트웨어 드라이버인 표준 리소스 어댑터(Resource Adaptor)를 제공한다[2]. EIS 벤더가 제공하는 리소스 어댑터는 애플리케이션 서버에 플러그인 되어 EIS와 애플리케이션 서버 및 엔터프라이즈 애플리케이션의 연결(connection)을 지원한다. 해당 리소스 어댑터는 애플리케이션 서버에 배치되어 실행되어야 하는데, 이를 담당하는 배치 책임자(Deployer)는 특정 EJB 서버나 컨테이너를 포함하는 운영환경에 리소스 어댑터를 배치하기 위하여 배치 툴을 사용한다. 애플리케이션 개발 및 EIS 통합을 하는데 있어서 애플리케이션 컴포넌트 제공자는 이러한 작업을 단순화할 수 있는 툴을 사용함으로써 노력을 감소시켜준다. 본 논문에서는 J2EE 커넥터 아키텍처를 구현한 리소스 어댑터를 애플리케이션 서버에 배치하기 위한 과정을 단계별로 분류 및 정의한다. 본 논문의 구성은 다음과 같다. 2장에서는 J2EE 커넥터 아키텍처의 개괄적인 내용을 살펴보고, 3장에서는 J2EE 커넥터 아키텍처를 지원하는 리소스 어댑터의 배치과정으로서 배치기가 수행하는 패키징 및 개략적인 배치 단계를 정의하며, 4장에서는 배치구성 활동을 단계별로 세부적으로 나누어 살펴본다. 5장에서는 결론 및 향후 연구 방향에 대해서 언급한다.

2. 관련연구

클라이언트 애플리케이션, 웹 컴포넌트, Enterprise JavaBeans 컴포넌트 등과 같은 엔터프라이즈 애플리케이션은 ERP 시스템, 메인프레임 트랜잭션 처리(TP) 시스템, 관계형 데이터베이스 등과 같은 EIS에 접근하여 관련된 기능과 데이터를 사용할 필요가 있다. 애플리케이션 서버는 컨테이너를 확장하여 이종의 EIS와의 연결을 허용한다. EIS 벤더는 J2EE 커넥터 아키텍처를 따르는 리소스 어댑터를 제공하며, 이것은 애플리케이션 서버에 플러그인 되어 EIS와 애플리케이션 서버 및 엔터프라이즈 애플리케이션의 연결을 지원한다. 커넥터 아키텍처는 J2EE 플랫폼을 이기종의 EIS와 연결하기 위한

표준 API로써 EIS와의 연결을 위한 클라이언트 API인 일반 클라이언트 인터페이스(Common Client Interface) 및 시스템 레벨의 표준 시스템 컨트렉트(System Contracts)를 정의한다. EIS 벤더는 커넥터 아키텍처의 설계서에 맞게 구현함으로써 각 EIS에 해당되는 시스템 레벨 소프트웨어 드라이버인 표준 리소스 어댑터를 제공한다[그림 1]. 리소스 어댑터는 해당 EIS에 특화(specific)된 것으로 애플리케이션 서버나 애플리케이션 클라이언트가 각 EIS에 연결하기 위하여 사용된다. 하나의 애플리케이션 서버에는 다수의 리소스 어댑터가 조립될 수 있으며, 이러한 속성으로 인하여 애플리케이션 서버에 배치된 애플리케이션 컴포넌트는 해당 EIS에 접근할 수 있게 된다.



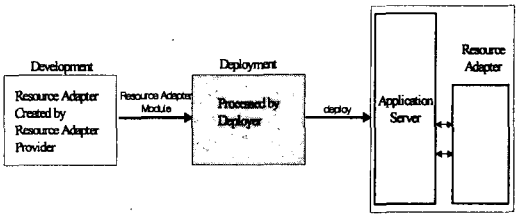
[그림 1] 커넥터 아키텍처

애플리케이션 서버와 EIS는 트랜잭션, 보안 및 커넥션 관리와 같은 모든 시스템 레벨의 메커니즘을 애플리케이션 컴포넌트로부터 투명하게 유지하도록 상호 협력한다. 결국, 애플리케이션 컴포넌트 제공자는 EIS 통합과 관련된 시스템 레벨의 사항에는 신경쓰지 않고 비즈니스 로직 및 프레젠테이션 로직의 개발에만 전념할 수 있게 된다. 이것은 다수의 EIS와의 연결에서 요구되는 확장가능하고(scalable) 안전(secure)하며 트랜잭션을 처리할 수 있는 엔터프라이즈 애플리케이션을 개발할 수 있도록 한다[2].

3. 패키징과 배치방법

리소스 어댑터 제공자는 필요한 자바 인터페이스 및 클래스를 리소스 어댑터에 구현해야 하며, 리소스 어댑터를 개발하는데 있어서 대상 EIS에 대한 원시 라이브러리를 사용해야 한다. 리소스 어댑터 모듈은 자바 인터페이스 및 클래스, 원시 라이브러리, 도움말(help) 파일, 도큐먼테이션, 기타 리소스들, 배치 디스크립터(Deployment Descriptor)로 구성되며 이들은 RAR(Resource Adapter Archive) 형태로 패키징된다. XML DTD와 일관성있게 작성된 배치 디스크립터는 리소스 어댑터 제공자와 리소스 어댑터 배치 책임자간의 약정(contracts)을 나타낸다[3]. 리소스 어댑터를 EJB 서버와 컨테이너를 포함하는 대상 운영환경에 배치하기 위해서는 명시적인 정보를 제공하는 이러한 배치 디스크립터에 정의된 속성을 근거로 해야한다[4].

리소스 어댑터 모듈은 독립된(stand-alone) 형태로 애플리케이션 서버에 직접 배치되거나 또는 한 개 이상의 J2EE 모듈로 구성된 J2EE 애플리케이션과 함께 배치된다.



[그림 2] 리소스 어댑터의 패키징 및 라이프 사이클

[그림 2]는 개발된 리소스 어댑터가 배치 책임자에 의해 일련의 배치 과정을 통해 애플리케이션 서버에 배치되는 과정을 나타내는 것으로 세부적인 배치 과정은 일반적으로 다음과 같다.

운영 환경에 리소스 어댑터를 적절하게 배치하기 위해서 배치기는 먼저 리소스 어댑터 모듈인 .rar 파일로부터 배치 디스크립터 파일인 ra.xml을 읽어서 배치 디스크립터에 명시된 배치구성 요구사항을 반영한다. 배치기는 원시 라이브러리 및 자바 클래스와 같은 모든 리소스 어댑터 모듈을 읽은 후, 애플리케이션 서버에 이러한 부분들을 인스톨한다. 그런 다음 배치 디스크립터에 명시된 요구사항을 근거로 대상 운영 환경에 적합하도록 각 부분에 대한 배치구성 활동을 수행한다[2]. 이러한 배치구성 활동의 각 단계는 4장에서 구체적으로 정의하고 설명한다.

4. 배치구성 단계

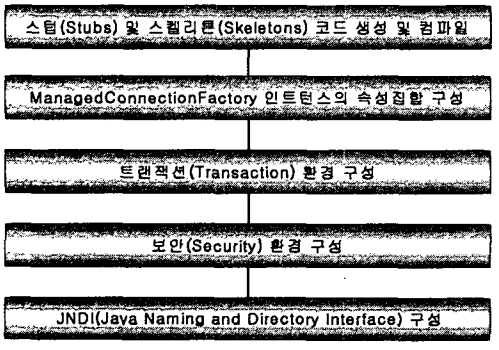
4.1 엔터프라이즈 빈의 배치구성 단계

리소스 어댑터 모듈은 독립된 형태로 애플리케이션 서버에 직접 배치되거나 또는 한 개 이상의 J2EE 모듈로 구성된 J2EE 애플리케이션과 함께 배치되는데, 여러 개의 J2EE 애플리케이션이 하나의 리소스 어댑터 모듈을 공유하는 경우에 독립적인 형태로 리소스 어댑터 모듈이 배치되며, 단일 J2EE 애플리케이션 내에서 여러 개의 컴포넌트들에 의해 하나의 리소스 어댑터 모듈이 필요한 경우에는 J2EE 애플리케이션 모듈과 함께 배치된다. J2EE 애플리케이션과 함께 배치되는 리소스 어댑터 모듈의 배치과정은 J2EE 애플리케이션인 엔터프라이즈 빈의 배치과정과 일치하며 배치 단계는 1) 엔터프라이즈 빈에 대한 스텁(stubs) 및 스켈리튼(skeletons) 코드 생성 및 컴파일 2) 보안(security) 환경 구성 3) 트랜잭션(transaction) 환경 구성 4) JNDI 구성, 5) 컨테이너-관리

(container-managed persistence) 엔터프라이즈 빈에 대한 데이터베이스 테이블 작성의 단계 등 다섯 단계를 거친다[3].

4.2 리소스 어댑터의 배치구성 단계

리소스 어댑터를 배치하기 위해서 배치기는 먼저 리소스 어댑터 모듈로부터 배치 디스크립터 파일을 읽어서 배치 디스크립터에 명시된 배치구성 요구사항을 반영한다. 다음으로 원시 라이브러리 및 자바 클래스와 같은 모든 리소스 어댑터 모듈을 읽은 후, 애플리케이션 서버에 이러한 부분들을 인스톨한다. 그런 다음 배치 디스크립터에 명시된 요구사항을 근거로 대상 운영환경에 적합하도록 각 부분에 대한 배치구성 활동을 수행한다. 본 논문에서는 독립형 리소스 어댑터 모듈을 배치하기 위한 배치구성 활동을 위의 4.1의 엔터프라이즈 애플리케이션의 배치 방법과 구분하여 [그림 3]과 같이 다섯 단계로 분류 및 정의한다.



[그림 3] 리소스 어댑터의 배치구성 활동

각 단계의 세부 사항은 다음과 같다.

첫째, 각 리소스 어댑터에 대한 스텁 및 스켈리톤 코드를 생성하고 컴파일한다. 애플리케이션 컴포넌트와 각 EIS를 연결하기 위해 필요한 스텁과 스켈리톤 코드를 생성하고 컴파일한다.

둘째, 각 ManagedConnectionFactory 인스턴스에 한 개 이상의 속성 집합(property set)을 설정한다. javax.resource.spi.ManagedConnectionFactory는 EIS와의 실질적인 연결을 하기 위한 인터페이스로서 이것은 리소스 어댑터에서 구현된다. 다양한 EIS 인스턴스에 연결하기 위하여 각 ManagedConnectionFactory 인스턴스에 한 개 이상의 속성 집합을 생성한다. 배치 디스크립터에 명시된 config-property-name, config-property-type, config-property-value 및 필드의 명세(description) 등을 바탕으로 각 환경 설정 필드에 대한 유효한 값을 지정한다. 이외에도 필요한 경우에는 서버 명, 포트 번호, 사용자 이름, 암호, 커넥션 URL 등과 같은 표준 속성(Standard Properties)을 참고한다. 각 속성 집합은 특정 EIS 인스턴스와의 연결을 하기 위한 특정 배치 값이다. 리소스 어댑터가 동일 EIS에 여러 개의 인스턴스와의 연결에 사용될 수 있기 때문에 하나의 리소스 어댑터에 복수개의 속성 집합이 있을 수 있다. 즉 배치기는 배치 디스크립터를 기초로 성에 일련의 속성 집합을 추가하거나 삭제할 수 있음을 의미한다. 또한 배치단계에서 운영 환경으로부터 리소스 어댑터를 추가하거나 삭제할 수 있어야 한다.

셋째, 트랜잭션 관리에 대한 애플리케이션 서버 메커니즘을 구성한다. 리소스 어댑터에 지원되는 트랜잭션 레벨을 근거로 하여 트랜잭션 관리를 위한 애플리케이션 서버 메커니즘을 구성한다. 지원되는 트랜잭션 레벨은 no_transaction, local_transaction, xa_transaction 중의 하나이며 이것은 배치 디스크립터에 명시되어 있다.

넷째, 운영 환경에서의 보안을 구성한다. 배치 디스크립터의 인증 메커니즘과 대응되는 허가 인터페이스를 나타내는 auth-mechanism 속성을 참고로 하여 리소스 어댑터가 특정 인증 메커니즘과 허가 인터페이스를 지원하는지의 여부와 리소스 어댑터에 명시된 보안 요구사항을 근거로 하여 EJB 서버와 컨테이너를 포함하는 대상 운영환경에서의 보안을 구성한다.

다섯째, JNDI를 구성한다. ManagedConnectionFactory 인스턴스를 이용하여 ConnectionFactory의 인스턴스를 생성한다. 그런 다음 JNDI 네임 스페이스(name space)에 ConnectionFactory 인스턴스에 대한 참조를 등록한다.

5. 결론

J2EE는 커넥터를 통하여 기존의 정보 시스템(EIS)과 통합하도록 확장된다. 커넥터는 EIS를 J2EE와 연결하는 벤더에 특화된 브릿지이다. 커넥터 아키텍처를 구현한 리소스 어댑터를 사용함으로써 특정 벤더의 J2EE 플랫폼에 종속되지 않도록 EIS를 통합할 수 있다. 이러한 리소스 어댑터는 각 EIS 벤더가 제공하며 이것은 애플리케이션 컴포넌트와 마찬가지로 서버에 배치되어 실행된다. 본 논문에서는 J2EE 커넥터 아키텍처를 구현한 리소스 어댑터를 애플리케이션 서버에 배치하기 위한 배치 단계를 정의하였다. J2EE 애플리케이션과 함께 배치되는 리소스 어댑터 모듈의 배치과정은 J2EE 애플리케이션의 배치과정과 일치하므로 간단히 살펴본 후, 독립형 리소스 어댑터 모듈을 배치하기 위해 배치기가 수행해야 하는 배치 단계를 크게 네 단계로 구분하였고, 배치구성 활동을 다섯 단계로 분류 및 정의하였다. 향후 연구 과제로는 리소스 어댑터의 배치 과정을 개략적으로 정의한 본 논문의 내용을 보다 구체적으로 연구하여 이러한 커넥터 아키텍처를 지원하는 리소스 어댑터를 애플리케이션 서버에 배치할 수 있도록 하는 배치기를 설계 및 구현하는 것이다.

참고문헌

[1] Sun Microsystems, Inc., "Java™ 2 Platform Enterprise Edition Specification," v1.2, 1999.

[2] Sun Microsystems, Inc., "J2EE Connector Architecture Specification," v1.0, June 1, 2000.

[3] Sun Microsystems, Inc., "Designing Enterprise Applications with the Java 2 Platform, Enterprise edition," v1.2, 1999.

[4] Sun Microsystems, Inc., "Enterprise JavaBeans™ Specification," v1.1, 1999.