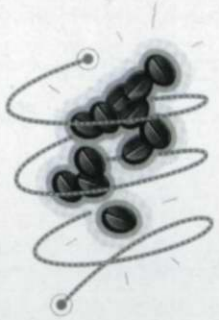
생성 및 사용 쉬운 컴포넌트 아키텍처

수년전부터 Java 프로그래밍 언어가 각광을 받고 있다. 객체지향 프로그래밍을 가능하게 하는 애플리케이션 개발 언어로서 Java가 차지하는 영역은 매우 크다. Java용 컴포넌트 아키텍처인 JavaBeans에 대해 살펴봤다. 〈편집자〉



JavaBeans : Java용 컴포넌트 아키텍처

지난 수년동안 재사용 가능한 소프트웨어 컴포넌트를 결합하여 새로운 애플리케이션을 구축하는 방법이 그 생산성을 입증하면서 고객 애플피케이션을 개발하는 일반적인 행태로 자리잡고 있다.

마이크로소프트의 비주얼 베이직과 같은 초기 제품은 VBX 컴포넌트와 폼 기반의 애플리케이션 어셈블리 과정을 포함하고 있어 광범위한 애플리케이션을 구축하는데 유용하다는 사실이 입증됐다.

비주얼 베이직에 이어 강력한 데이터 액세스 컴포넌트와 객체지향적인 컴포넌트 확장기능을 추가함으로써 기본적인 컴포넌트 어셈블리 애플리케이션 개발모델을 크게 보강한 볼랜드의 델파이 등이 선보였다.

JavaBeans는 이러한 컴포넌트 소프트웨어 어셈블리 패러다임을 새로운 수준으로 끌어 올렸다. JavaBeans는 역동적인 Java 컴포넌트를 생성하고 사용하기 위한 아키텍처이자 플랫폼 중립적인 API로 JavaBeans는 기존의 혁신적인 제품들에 의해 절립된 컴포넌트 어셈블리 개발모델의 강점을 토대로 구축되어 그 성능을 더욱 확장시키고 있다.

애플리케이션 개발자들은 고객 애플리케이션을 조합할 수 있을 것이다. 여기서는 JavaBeans를 개략적으로 살펴보고 그의 기능에 대해 알아보았다.

JavaBeans를 통해 이식가능한 Java 플랫폼의 기능확장 방법

소프트웨어 컴포넌트 모델을 구성하는 핵심 요소

- JavaBeans 기능들의 강조
- JavaBeans 준비 방법

- JavaBeans를 통한 Java 플랫폼 확장 방법

Java의 장점 활용하기

Java는 환벽하게 이식가능한 인터넷과 기업 인트라넷 애플릿과 애플리케이션을 구축하기 위한 업계 표준 플랫폼으로 확고히 자리잡고 있다. Java 플랫폼은 이러한 유형의 애플리케이션을 개발하는 사람들에게 여러 가지의 이점을 제공한다.

완벽하게 이식가능한 플랫폼 : 브라우저 영역에서 언어와 라이브러리, 가상기기의 파급력을 과시하고 있는 Java 플랫폼은 조만간 운영체계 영역에서도 급속히 파급될 것이다. 따라서 개발자들은 애플리케이션 기능은 한번만 작성하면 다양한 운영체계와 하드웨어에서 그 애플리케이션을 전개할 수 있다.

강력하고 컴팩트한 환경 : Java 플랫폼은 개발자들이 객체지향 언어의 강점을 충분히 활용하면서도 다른 언어나 프로그래밍 모델 환경에서 초래되는 덩치 큰 객체의 생성과 등록과정의 복잡함과 번거로움을 없애준다. 간결한 런타임은 Java가 점점 확산되고 있는 클라이언트나 서버급 PC 혹은 웍스테이션 뿐만 아니라 PDA의 내장 시스템 칩에 통합될 수 있다.

네트웍 인식 : Java 플랫폼은 초기부터 네트웍 인식을 주 목적으로 했기 때문에 TCP/IP가 기본적으로 지원된다. 클라이언트측 데이터에 의한 애플릿 간섭으로부터 완전히 보호하는 보호 매커니즘이 내장되어 있다. Java 플랫폼은 결국 번거로운 설치나 등록과정을 거치지 않고 클라이언트 환경으로 쉽게 다운로드할 수 있는 자가기술 클래스로 애플릿을 설계하는 것을 목표로 한다.

이런 강점을 토대로 구축된 JavaBeans는 Java 플랫폼을 더욱 확장시킬 수 있다.

Java 플랫폼의 확장

Java 애플릿은 간단한 정적 컴포넌트 모델을 제공한다. 애플릿은 웹페이지에 둘 수 있지만 애플릿은 그 웹 페이지나 그 페이지내에 있는 다른 Java 애플릿과는 상호작용할 수 없다.

JavaBeans는 좀더 풍부하고 역동적인 상호작용을 지원함으로써 Java 플랫폼을 보강한다. JavaBeans는 개발자들이 독립적인 컴포넌트들을 정의해 다양한 브라우저 또는 비브라우저 환경(예 : 핫Java, 넷스케이프 네비게이터, 인터넷 익스플로러, 비주얼 베이직 폼스, 클라리스 웍스)에서 새로운 애플리케이션을 작성할 때 다양하게 조합해 사용할 수 있다.

JavaBeans의 컴포넌트는 GUI 위젯이 될 수도 있고 눈으로 볼 수 없는 기능이나 서비스가 될 수도 있으며 애플릿이나 좀더 규모가 큰 애플리케이션일 수도 있다. 이러한 각각의 컴포넌트들은 서로 다른 개발자들에 의해 제각각 개발될 수 있다. JavaBeans 컴포넌트들은 동일한 애플리케이션의 일부가 될 필요는 없다. 대신 서로 역동적으로 상호작용한다.

예를 들어 온라인 뱅킹 서비스를 제공하는 웹 사이트를 가정해 보자. 이 웹 사이트는 사용자들에게 그래프 형식을 빌어 역대 이자율에 대한 정보를 제공할 수 있다. 이 웹 페이지가 이자율 데이터 검색 컴포넌트와 그래프 작성 컴포넌트의 2가지 JavaBeans 컴포넌트를 포함하고 있을 것이다. 이자율 검색 컴포넌트는 이자율에 대한 데이터를 찾아내어 그래프로 만들기 위해 그래프 작성 컴포넌트와 역동적으로 상호작용하는 과정을 거치게 된다.

그리고 여기에 또다른 웹 사이트 개발자가 고객들이 역대 계정 수치 정보를 검색할 수 있는 새로운 컴포넌트를 추가할 수 있을 것이다. 그렇게 되면 계정 수치 데이터는 기존의 그래프 작성 컴포넌트를 재사용해 그래프로 만들어 질 수 있다. 즉, 개발자들은 이러한 3가지 컴포넌트를 하나로 결합한 애플리케이션을 새로 만들 필요없이 계정 수치 검색 컴포넌트만 추가하면 되는 것이다.

JavaBeans는 Java 플랫폼의 일부이므로 새로운 컴포넌트들은 필요한 경우 기존의 그래프 작성 컴포넌트와 역동적으로 작용할 수 있다. 이때 그래프 작성 및 이자율 검색 컴포넌트는 상업적인 소프트웨어 개발자들이 제공할 수 있고 계정 수치 검색 컴포넌트는 특정 은행에 맞춰 개별적으로 구축할 수 있다.

JavaBeans는 이와 같이 역동성과 유연성 및 재활용 가능성을 추가함으로써 Java 플랫폼을 크게 향상시켜 준다.

JavaBeans의 파워

Java의 설계 철학과 강점을 그대로 따르고 있는 JavaBeans는 구축과 사용이 쉬우며 네트웍 전개에 적합하도록 컴팩트하며 완전히 이식가능하다.

JavaBeans는 앞으로 툴 개발업체들이 좀더 강력하고 생산성이 높은 Java 어셈

하나의 데스크탑 컴포넌트 모델을 이기종으로 복잡하게 연결된 네트웍 컴퓨팅 환경의 요구사항들을 모두 충족시킬 수 있도록 확장하려면 많은 문제점이 야기되며 어느정도 절충해야 한다.

블리 툴을 생산할 수 있는 토대를 마련할 것이다. 즉 JavaBeans는 애플리케이션 개발자들이 인터넷 및 인트라넷 애플리케이션에 한차원 높은 상호작용성과 더욱 풍부한 기능을 보강할 수 있도록 힘을 불어넣을 것이다.

결국 개발자들은 각종 인기있는 환경내에서 JavaBeans를 재사용함으로써 유용성을 극대화할 수 있게 된다. JavaBeans는 또한 핫Java, 넷스케이프 네비게이터, 볼랜드 제이빌더, 시맨텍 비주얼 카페, 아이비엠 Java용 비주얼 에이지, 마이크로소프트 인터넷 익스플로러, 비주얼 베이직, 비주얼 J++, 워드 및 클라리스웍스 등의 각종 컴포넌트와 어셈블리 툴 및 컨테이너를 재사용하게 된다.

JavaBeans는 Java 플랫폼 전체에 걸쳐 이식할 수 있고 다양한 개발툴이나 컨테이너를 재사용할 수 있기 때문에 JavaBeans를 상업적으로 개발하는 사람들은 좀더 많은 고객층을 확보할 수 있게 된다.

컴포넌트 모델 개요

컴포넌트 모델이란 어떤 애플리케이션을 만들 때 역동적으로 결합시킬 수 있는 소프트웨어 컴포넌트를 개발자들이 규정할 수 있게 하는 아키텍처이자 일련의 API라고 할 수 있다. 컴포넌트 모델은 컴포넌트와 컨테이너라른 2가지의 주요 요소로 구성돼 있다.

컴포넌트는 크기나 기능면에서 다양하여 버튼과 같은 작은 GUI 위젯에서부터 평판 뷰어 같은 애플릿 크기, 나아가 텍스트 레이아웃 애플리케이션의 HTML, 브라우저와 같이 규모가 큰 애플리케이션 등이 있다. 컴포넌트는 버튼과 같이 시각적으로 표시될 수도 있고 데이터 공급 모니터링 컴포넌트와 같이 시각적으로 보이지 않는 것도 있다.

컨테이너는 어셈블리나 관련 컴포넌트를 보유하는데 사용된다. 즉, 컨테이너는 각 컴포넌트들을 정렬하거나 서로 상호작용할 수 있도록 하는 컨텍스트를 제공한다. 따라서 컨테이너는 폼, 페이지, 프레임 혹은 셀이라고 불리기도 한다. 컨테이너 또한 컴포넌트가 될 수 있다. 즉, 하나의 컨테이너가 또다른 컨테이너의 내부 컴포넌트로 사용될 수 있다.

컴포넌트 모델 서비스

컴포넌트 모델은 컴포넌트 인터페이스 발현 및 발견, 컴포넌트 특성, 이벤트 처리, 지속성, 애플리케이션 구축기 지원, 컴포넌트 패키징 등의 주요 서비스를 제공한다. 이 서비스들의 특성은 다음과 같다.

- 런타임 컴포넌트 인터페이스 발현 및 발견

이는 각 컴포넌트가 다른 컴포넌트나 애플리케이션 스크립트로부터 호출이나 이벤트 통보를 받았을 때 역동적으로 활성화될 수 있도록 자체 인터페이스를 발현하는 매커니즘이다. 앞서 예로 든 온라인 뱅킹의 경우 이 매커니즘을 적용하면 계정수치 컴포넌트가 그래프 작성 컴포넌트에게 데이터를 그래프로 그리도록 요구하는 것이다.

그래프 작성 컴포넌트가 자신의 인터페이스를 컴포넌트 환경에 발현시켰기 때문에 이 컴포넌트는 계정 수치 컴포넌트와 동일한 애플리케이션의 일부가 되지 않아도 된다. 2가지 컴포넌트가 따로따로 구축되었더라도 컴포넌트 환경에서 제공하는 서비스들을 사용해 역동적으로 상호작용할 수 있게 된다.

- 컴포넌트 자산

컴포넌트의 특성은 그 컴포넌트의 현 상황을 직접적으로 반영하거나 영향을 주는 공공연한 속성이라고 할 수 있다. 예를 들어 누르기 버튼의 전면 색깔이나 실시간 주식제공 컴포넌트의 주식 시몰 등이 그 컴포넌트의 특성이다. 컴포넌트 모델은 이러한 특성을 컴포넌트 환경에 발현시킴으로써 표준 메커니즘을 통해 컴포넌트의 현 상황에 질문하거나 그 상황을 수정할 수 있게 한다.

- 이벤트 처리

이벤트 처리는 컴포넌트가 이벤트를 유

발하거나 방송하는 메커니즘으로 통보될 필요가 있거나 통보를 원하는 해당 컴포넌트에 그 이벤트를 전달하는 것이다. 통보를 받은 컴포넌트는 그에 상응해 특정 기능을 수행한다.

예를 들어 온라인 뱅킹 애플리케이션 개발자가 사용자들이 막대 그래프나 선 그래프 중에서 선택할 수 있는 버튼을 제공한다고 하면 이벤트 처리 시스템은 사용자가 그 버튼 컴포넌트를 클릭했을 때 그래프 작성 컴포넌트에 이를 통보한 것이다.

마우스 클릭과 같은 시스템 이벤트 외에도 컴포넌트는 자체 이벤트를 규정할 수 있다. 예를 들어 주식티켓 정보나 주식 가격 변동 상황 및 기계 파워 레벨의 변동과 같은 생생한 데이터를 모니터하는 컴포넌트가 있다고 하자. 이 변경된 데이터 이벤트는 또다른 컴포넌트들에 의해 처리되어 경고음을 내거나 시각 디스플레이가 바뀌거나 새로운 프로세스를 시작하게 된다.

- 지속성

지속성은 컴포넌트의 현상태를 영구적으로 보존되는 장소에 저장하는 메커니즘을 말한다. 컴포넌트 상태은 컨테이너의 턴텍스트하에서 다른 컴포넌트와 관계를 맺으며 저장된다. 예를 들어 사용자가 자신의 계정 정보와 특정 그림 유형이 들어있는 온라인 뱅킹 웹 페이지를 저장하려면 지속성 매커니즘이 이를 지원해 준다.

- 애플리케이션 구축기 지원

애플리케이션 구축기 지원 인터페이스는 각 컴포넌트들이 자신의 특성과 행동을 애플리케이션 구축기 개발툴에 발현시키도록 한다. 이 인터페이스를 사용해 툴들은 임의의 컴포넌트 특성과 행동(이벤트, 메쏘드)을 알아낼 수 있다.

이 툴들은 툴 팔레트, 검사기, 편집기와 같은 메커니즘을 제공하여 애플리케이션 개발자들이 애플리케이션을 개발할 때 다양한 컴포넌트를 조합할 수 있도록 지원한다. 애플리케이션 개발자들은 이러한 메커니즘을 통해 컴포넌트간의 관계를 설정하는 것은 물론 각 컴포넌트의 상황과 모습을 수정할 수 있다.

앞서의 온라인 뱅킹 사례의 버튼과 그래프 작성 컴포넌트를 예로 들어보자. 최종 사용자가 막대 그래프에서 선 그래프로 바꾸는 선택버튼을 누르는 경우를 생각해 보자. 개발자들은 이 애플리케이션을 구축할 때 컴포넌트 특성 편집기를 이용해 버튼의 크기, 색상, 레이블과 같은 겉모양을 정하고 막대 그래프의 기본적인 형태를 규정할 것이다. 또한 개발자들은 다른 애플리케이션 개발 툴 메커니즘을 사용하여 버튼의 클릭 이벤트와 그래픽 작성 컴포넌트의 그래픽 형식 특성간 관계를 규정할 수 있다.

- 컴포넌트 패키징

JavaBeans의 컴포넌트들은 네트웍을 통해 전개되고 분배될 수 있기 때문에 컴포넌트를 구성하는 자원을 물리적으로 묶는 기능을 제공해야 한다. 그래야 컴포넌트의 분배 및 전개관리가 쉬워진다.

컴포넌트 모델은 Java 아취브 파일 포맷을 사용해 기술을 제공함으로써 사운드, 이미지, 도움말 파일, 국지화 메시지 카탈로그 등 임의의 컴포넌트 자원과 클래스 파일을 물리적으로 하나의 엔티티로 만들어 분배한다.

분산 컴퓨팅

이러한 주요 서비스외에도 컴포넌트 모델은 분산 컴퓨팅 환경에서 컴포넌트를 사용할 수 있는 전략을 제공한다. 예를 들어 공장 기기들의 상태를 모니터하는 컴포넌트는 그 기기에 부착되어 있는 서버상에서 실행할 수 있다. 만약 데이터값이 변한다면 그 서버 컴포넌트는 네트웍을 통해 데스크탑에서 실행되는 또다른 소프트웨어 컴포넌트에 전달될 이벤트를 발생시킬 것이다. 그러면 그 데스크탑 컴포넌트는 메시지를 게재하거나 그래프의 형태를 바꾸는 등으로 적절하게 반응을 보일 것이다.

한 컴퓨터내에서 상호작용하는 소프트웨어 컴포넌트와 네트웍상에서 상호작용하는 컴포넌트간에는 분명히 큰 차이가 있다. 개발자들과 분산 컴퓨팅 하부구조는 네트웍 속도 지연에 대해 고려해야 할 뿐만 아니라 컴포넌트나 기기가 장애를 일으킬 경우 적절한 복구와 재동기화 매커니즘을 제공해야 한다.

하나의 데스크탑 컴포넌트 모델을 이기종으로 복잡하게 연결된 네트웍 컴퓨팅 환경의 요구사항들을 모두 충족시킬 수 있도록 확장하려면 많은 문제점이 야기되며 어느정도 절충해야 한다. 그러한 데스크탑 컴포넌트 모델은 API가 매우 복잡해지거나 실행환경의 덩치가 너무 커질 수 있고 혹은 분산 컴퓨팅 환경 기능이저하되는 등의 부작용이 생길 것이다.

결론적으로 모든 분산 컴퓨팅 기능을 제공하는 컴포넌트 모델은 CORBA와 같은 기존의 강력한 분산 컴퓨팅 기수을 잘 활용할 것이다. 이렇게 함으로써 단일기기 컴포넌트 모델은 컴팩트하면서도 분산 애플리케이션이 요구하는 모든 기능에 충

분히 액세스할 수 있을 것이다.

JavaBeans API의 하이라이트

JavaBeans는 역동적인 Java 컴포넌트와 컨테이너 기능을 구축하고 사용하기 위한 아키텍처이나 플랫폼 중립적인 API로 컴포넌트 인터페이스 발현 및 발견이나 컴포넌트 특성과 같은 주요 컴포넌트 모델 서비스를 제공한다.

JavaBeans 컴포넌트 모델 서비스는 마이크로소프트의 액티브X, CI랩의 오픈닥, 넷스케이프의 라이브 컨넥트 등의 특정 컴포넌트 모델을 연결함으로써 구현할 수 있다. 또 JavaBeans는 앞으로 Java소프트의 내장가능한 JavaOS에서도 실행될 것으로 보인다. JavaBeans API를 다양한 컴포넌트 모델들에 연결하는 역할은 라이브러리들이 담당하게 된다.

따라서 개발자들은 완벽하게 인식가능한 JavaBeans API만 사용해도 Java로 컴포넌트를 완벽하게 구축할 수 있다. 또 개발자들은 이식가능한 Java 코드내의 특정 컴포넌트 모델 호출이나 이식 불가능한 플랫폼 등에 구애를 받지 않아도 될 것이다. JavaBeans API를 포함하고 있는 Java 플랫폼은 컴포넌트 개발자들이 Java 언어로 컴포넌트의 행동과 기능을 모두 구현할 수 있도록 한다.

JavaBeans 컴포넌트는 또한 한번 작성한 애플리케이션을 모든 플랫폼에서 사용할 수 있도록 하는 Java의 특징을 새로운 차원으로 끌어 올린다. 즉 JavaBeans는 넷스케이프(Java스크립트와 라이브컨넥트를 사용하는), 핫Java와 기타 Java 컨테이너 및 마이크로소프트 컨테이너(익스플로러, 비주얼 베이직, 윈도우 셀, 워드 등), 오픈닥 컨테이너, 파워빌더나 델파이, 액티브X 컨테이너 등을 비롯한 각종 컨테이너에 최대한 결합된다.

JavaBeans 서비스는 곧 Java 플랫폼의 일부가 될 것이다. 이렇게 되면 개발자들은 JavaBeans를 사용한 애플릿이나 애플리케이션을 실행시키기 위해서 라이브러리를 추가로 분배하지 않아도 된다. 또한 JavaBeans는 컨테이너의 외부에서 역동적으로 의사소통하는 독립적인 Java 애플릿으로도 사용될 것이다.

JavaBeans 플랫폼과 아키텍처 중립적인 API, 그리고 Java 컴포넌트의 모든 기능이 완벽하게 인식된다는 사실을 고려해 볼 때 JavaBeans의 재활용성은 매우 크다. JavaBeans는 다른 컴포넌트 모델과는 달리 특정 플랫폼이나 컨테이너, 컴포넌트 모델에 종속되지 않는다. 결론적으로 JavaBeans를 상업적인 용도로 개발하는 사람들은 좀더 광범위한 시장을 확보할 수 있게 될 것이다.

컴포넌트의 소비자들 또한 JavaBeans 컴포넌트의 구입 후 최대한으로 활용할 수 있을 뿐만 아니라 사용법 습득에 걸리는 시간도 줄일 수 있게 될 것이다. JavaBeans를 인터넷이나 인트라넷 혹은 배타적인 클라이언트 서버 애플리케이션 등에 광범위하게 사용하게 된다.

JavaBeans는 다른 컴포넌트 모델과는 달리 특정 플랫폼이나 컨테이너, 컴포넌트 모델에 종속되지 않는다.
결론적으로 JavaBeans를 상업적인 용도로 개발하는 사람들은 좀더 광범위한 시장을 확보할 수 있게 될 것이다.

설계 목표

JavaBeans API의 주요 설계 목표와 구현 방법은 다음과 같다.

- JavaBeans는 컴팩트하며 생성과 사용이 쉽다

JavaBeans는 매우 컴팩트하며 생성이 쉽다. 특히 간단한 컴포넌트는 더욱 쉬울 것이다. 물론 Java로 복잡한 컴포넌트를 구축할 수도 있다. Java 플랫폼의 모든 기능을 완벽하게 이용함으로써 컴팩트한 상태를 그대로 유지할 수 있다.

기존의 컴포넌트 모델 API는 복잡하고 덩치가 큰 애플리케이션급의 컴포넌트에서 더 가벼운 위젯 크기의 컴포넌트로 축

소되고 있다. 반면 JavaBeans API는 단순하고 가벼운 위젯이 애플릿에서 좀더 많은 기능을 갖춘 애플리케이션으로 확대할 수 있도록 설계되었다.

따라서 JavaBeans API는 이 작은 위젯이나 애플릿에 복잡함이나 무게를 과도하게 더하지 않을 것이다. JavaBeans에서는 이렇게 가볍고 크기도 작은 컴포넌트가 주로 사용되기 때문에 JavaBeans의 컴팩트 설계는 컴포넌트 개발자들이나 소비자 모두에게 합리적이다.

- JavaBeans는 완벽하게 이식가능하다

JavaBeans는 플랫폼 중립적인 JavaBeans API를 통해 완벽하게 이식되며 표준 Java 플랫폼의 일부가 될 라이브러리들을 연결해 준다. 따라서 개발자들은 Java 애플릿에 호환되지 않는 코드나 특정 플랫폼에만 존재하는 라이브러리들을 포함시키지 않아도 된다.

-JavaBeans는 Java 플랫폼의 인터넷 기능을 최대로 활용한다

JavaBeans는 Java 플랫폼에 이미 구축되어 있는 클래스 발견 매커니즘을 최대한 이용한다. 이 매커니즘은 Java의 독특한 검사 및 반영기술을 사용한 것으로 Java는 인터페이스 발현이나 발견을 지원하기 위해 규모가 큰 등록 매커니즘을 런타임에 추가로 제공하지 않아도 된다.

또한 가볍고 이해하기 쉬운 매커니즘을 채택한 Java의 전체적인 설계 목표와 잘 부합되기 때문에 JavaBeans는 개발자의 프로그래밍 추가작업을 덜어준다. 일례로 AWT 컴포넌트는 자동적으로 JavaBeans가 된다.

JavaBeans의 라이브러리는 간단한 컴포넌트를 위해 컴포넌트의 기본 행동을 제공한다. 예를 들어 자동 유지 기능은 JavaBeans의 직렬화 기능에 의해 처리된다. JavaBeans는 또한 컴포넌트의 취득과 설정 메쏘드를 검사해 특성 편집기의 자동 생성 기능을 제공한다. 물론 좀더 복잡한 컴포넌트를 개발할 때나 좀더 풍부한 기능을 갖춘 컴포넌트 편집기를 개발할 때는 이러한 기본 행동은 무시할 수도 있다. 아울러 JavaBeans 컴포넌트는 새로운 AWT 데스크탑 통합 기능도 활용할 수 있다.

- JavaBeans는 강력한 분산 컴퓨팅 매커니즘을 활용한다

JavaBeans 컴포넌트 모델 API와 그 구현체들은 단일 가상 기기내에서 컴포넌트를 상호작용에 초점을 맞추고 있다. Java 개발자들은 JavaBeans API를 복잡하게 만들거나 무거운 분산 컴퓨팅 매커니즘을 Java 플랫폼에 추가하지 않고도 기존의 분산 컴퓨팅 방법중에서 선택할 수 있다.

예를 들어 Java의 RMI(Rmote Method Invocation)를 사용하거나 원격 객체 액세스용 CORBA IDL 인터페이스를 사용하거나 아니면 기타 분산 컴퓨팅 매커니즘에 의해 Java 애플리케이션에 분산 컴포넌트 상호작용 기능을 추가할 수 있다. 개발자들은 또한 자신이 원하는 이식성이나 성능 혹은 기존 시스템과의 통합 요구 조건에 가장 잘 맞는 매커니즘을 선택할 수 있다.

- 유연한 구축시간 컴포넌트 편집기

JavaBeans는 컴포넌트 개발자들이 자신의 컴포넌트에 알맞은 구축시간 특성 시트와 검사기 및 편집기를 다양하게 작성할 수 있도록 한다. 또한 컴포넌트 사용자들이 컴포넌트가 제공하는 기능으로부터 최대한의 결과물을 얻어낼 수 있도록 가장 생산성이 높은 방법을 제공한다.

예를 들어 데이터베이스 연결 컴포넌트 공급자는 개발자들이 구축시 사용할 수 있도록 더 길고 복잡한 특성 시트를 제공할 수도 있다. 아니면 다양한 특성들을 탭 섹션으로 조직하길 원할 수도 있고 또 컴포넌트 사용자가 구축시 테이블 결합을 시각적으로 지정하도록 할 수도 있다.

JavaBeans 애플리케이션 구축 툴 API는 컴포넌트 개발자들이 자신의 컴포넌트 형식에 가장 잘 맞는 특성 편집기를 생성할 수 있도록 지원한다.

구현 과정

Java소프트는 현재 설계리뷰 과정의 일부로서 실행되는 JavaBeans 컴포넌트 프로토 타입을 발표했다. 동사는 협력업체와의 공동 노력하에 액티브X나 오픈닥, 라이브 컨넥트와 같은 기존의 컴포넌트 모델에 대한 최상의 연결을 추구하고 있다.

따라서 Java소프트의 협력업체들은 JavaBeans를 비롯하여 Java 플랫폼의 모든 면모를 완벽하게 구현할 수 있는 능력을 갖추고 있어야 한다. 이런 방식을 통해 Java를 라이센스받은 업체들은 최상의 애플리케이션 사용 환경을 보장하게 된다.

현재 몇몇 컴포넌트 구축 업체들과 Java 개발자 툴 생산업체들이 JavaBeans API의 조기 검토과정에 적극적으로 참여하고 있다.