J2EE™용 Rational® XDE™ 모델 구조 가이드라인

Rational Software 백서
TP 154, 05/03

목차

1. 소개

이 문서는 Rational XDE , Java Platform Edition 에서 RUP 모델 아티팩트를 나타내고 구성하는 방법에 대한 권장사항을 제공합니다. 물론 이러한 RUP 아티팩트를 XDE 에서 모델링하는지 여부는 프로젝트에 따라 결정됩니다. 이 문서에서는 결정사항에 대한 영향을 고려하여, XDE 가 자동화 지원을 제공하는 모델과 해당 지원을 제공하지 않는 모델에 대해 설명합니다.

모든 XDE 모델은 XDE 프로젝트 내부에 존재하므로 XDE 프로젝트 구조 섹션은 작성되어야 하는 XDE 프로젝트와 이 프로젝트에서 작성되어야 하는 XDE 모델 파일에 대한 권장사항을 제공합니다.

RUP 및 XDE 모두 "모델"이라는 용어를 사용하며 RUP 모델과 XDE 모델 간의 맵핑이 항상 일 대 일 관계는 아닙니다. RUP 모델 대 XDE 모델 맵핑 섹션에서 RUP 모델에서 XDE 모델로의 맵핑에 대해 설명합니다.

그 다음으로 해당 섹션에서는 해당 XDE 모델 파일에서 각 RUP 모델 아티팩트의 구조에 대해 설명합니다.

2. 범위

이 문서는 연관된 RUP 아티팩트의 컨텐츠 개발 프로세스가 아닌 권장 XDE 모델 파일 구조의 설명에 초점을 맞춥니다. 이 문서는 또한 설명한 XDE 모델을 포함하는 XDE 프로젝트를 정의하는 세부 방법에 대해 설명하지 않습니다. RUP 아티팩트 컨텐츠의 정의, 개발 및 모델링 방법에 대한 정보는 RUP 를 참조하십시오. 프로젝트에 대한 자세한 정보는 IDE 문서를 참조하십시오.

이 문서는 전체 예제에 대해 설명하지는 않으며 대신 특정 내용을 강조하는 선택된 예제를 사용합니다. 그러나 모든 예제는 서로 일관성이 있으며 실제 XDE 모델을 참조한 예제입니다.

이 문서 버전은 태그 라이브러리 개발에 대해서는 다루지 않습니다.

이 문서에서 설명하는 프로젝트 및 모델 구조는 권장사항일 뿐이며 여러 가지 올바른 구조로 바꿀 수 있습니다.

3. XDE 프로젝트 구조

이 문서의 초점은 XDE 모델의 구조화 방법입니다. 그러나 모든 XDE 모델은 XDE 프로젝트 내부에 존재하므로 권장 모델 구조가 존재하는 프로젝트 구조에 대한 간략한 소개를 제공해야 합니다.

여러 명이 개발하는 J2EE 엔터프라이즈 응용프로그램의 경우, 다음 XDE 프로젝트 및 모델을 작성하는 것이 바람직합니다.
참고: XDE "프로젝트 작성" 마법사를 사용하는 경우, 프로젝트가 작성될 때 많은 모델이 자동으로 작성됩니다. 실제로, WSS AD XDE 를 사용하여 *엔터프라이즈 응용프로그램 모델링 프로젝트를 작성하는 경우, 많은 모델을 포함하여* 이 여러 프로젝트 구조의 대부분이 자동으로 작성됩니다. XDE 는 또한 모델 컨텐츠를 시작하는 모델 템플리트를 제공합니다.

XDE 프로젝트	설명	XDE 모델 "<권장 모델 이름>" (<XDE 파일 유형: 모델 템플리트>)
응용프로그램 프로젝트(XDE 기본 모델링 프로젝트)	응용프로그램 프로젝트는 전체 응용프로그램을 나타냅니다. 전체 응용프로그램에 대해 설명하는 XDE 모델 파일이 포함됩니다.	– "유스 케이스 모델"(Rational XDE: 유스 케이스 모델) – "분석 모델"(Rational XDE: 분석 모델) – "전체 디자인 모델"(Rational XDE: 디자인 모델) – "전체 구현 모델"(Rational XDE: 공백 모델) – "EAR 배치 모델"(Java: EAR 배치 모델)
데이터 모델링 프로젝트(XDE 데이터 모델링 프로젝트)	데이터 모델링 프로젝트에는 응용프로그램 데이터를 모델링하는 데 필요한 자원과, 데이터 모델과 데이터베이스의 라운드트립 엔지니어가 포함됩니다.	– "논리 데이터 모델"(데이터: 논리 데이터 모델) – "실제 데이터 모델"(데이터: *벤더 특정 실제 데이터 모델 파일*) [1] – "도메인 모델"(데이터: *벤더 특정 도메인 모델 파일*)
EJB 프로젝트(XDE EJB 모델링 프로젝트)	EJB 프로젝트에는 EJB 를 구현하는 데 필요한 자원이 포함됩니다. 포함된 요소는 EJB 모듈(.EJB-JAR 파일)로 패키지화되어 배치됩니다. 개별 EJB 또는 EJB 세트에 별도의 EJB 프로젝트를 정의할 수 있으며 각 EJB 프로젝트에는 Java 코드 모델이 하나만 포함될 수 있습니다. 권장사항은 생성될 각 EJB-JAR 마다 EJB 프로젝트를 작성하는 것입니다. 별도의 프로젝트가 정의되는 경우, 프로젝트 이름에 해당 컨텐츠를 반영해야 합니다. [2]	– "EJB 코드 모델"(Java: EJB 코드 모델) – "EJB 배치 모델"(Java: EJB 배치 모델)
웹 프로젝트(XDE 웹 모델링 프로젝트)	웹 프로젝트는 응용프로그램의 웹 자원을 나타냅니다. 포함된 요소는 웹 아카이브 파일(WAR 파일)로 패키지화되어 배치됩니다. 프리젠테이션 로직의 특정 영역에 별도의 웹 프로젝트가 정의될 수 있습니다. 권장사항은 생성해야 하는 각 WAR 마다 웹 프로젝트를 작성하는 것입니다. 별도의 프로젝트가	– "Java 코드 모델"(Java: Java 1.3/1.4 코드 모델) – "JSP 태그 라이브러리 모델"(웹: JSP 태그 라이브러리 모델)[4] – "가상 디렉토리 모델"(웹: 가상 디렉토리 모델)[5] – "웹 배치 모델"(웹: 웹 배치 모델)

	정의되는 경우, 프로젝트 이름에 해당 컨텐츠를 반영해야 합니다. [3]	

[1] Rational XDE 는 여러 데이터베이스 벤더에 대한 실제 데이터베이스 지원을 제공합니다. XDE 에서 지원하는 각 데이터베이스 벤더마다 벤더 특정 템플리트가 존재합니다.

[2] WSAD 용 XDE 를 사용하여 추가 EJB(모델링) 프로젝트를 작성하는 경우, 마법사가 EAR 을 호스트할 응용프로그램 프로젝트를 요청합니다. 이 때 위의 동일한 응용프로그램(모델링) 프로젝트를 재사용해야 합니다.

[3] WSAD 용 XDE 를 사용하여 추가 웹(모델링) 프로젝트를 작성하는 경우, 마법사가 EAR 을 호스트할 응용프로그램 프로젝트를 요청합니다. 이 때 위의 동일한 응용프로그램(모델링) 프로젝트를 재사용해야 합니다.

[4] 프로젝트마다 여러 태그 라이브러리 모델이 존재할 수 있습니다. 실제로는 각 .tld 파일마다 별도의 모델이 필요합니다. 이 문서 버전은 태그 라이브러리 개발에 대해서는 다루지 않습니다.

[5] XDE 웹 프로젝트당 여러 가상 디렉토리 모델이 존재할 수 있습니다.

해당 프로젝트 및 모델 조직 예제가 그림 1에 표시됩니다(고유 모델 이름 유의).

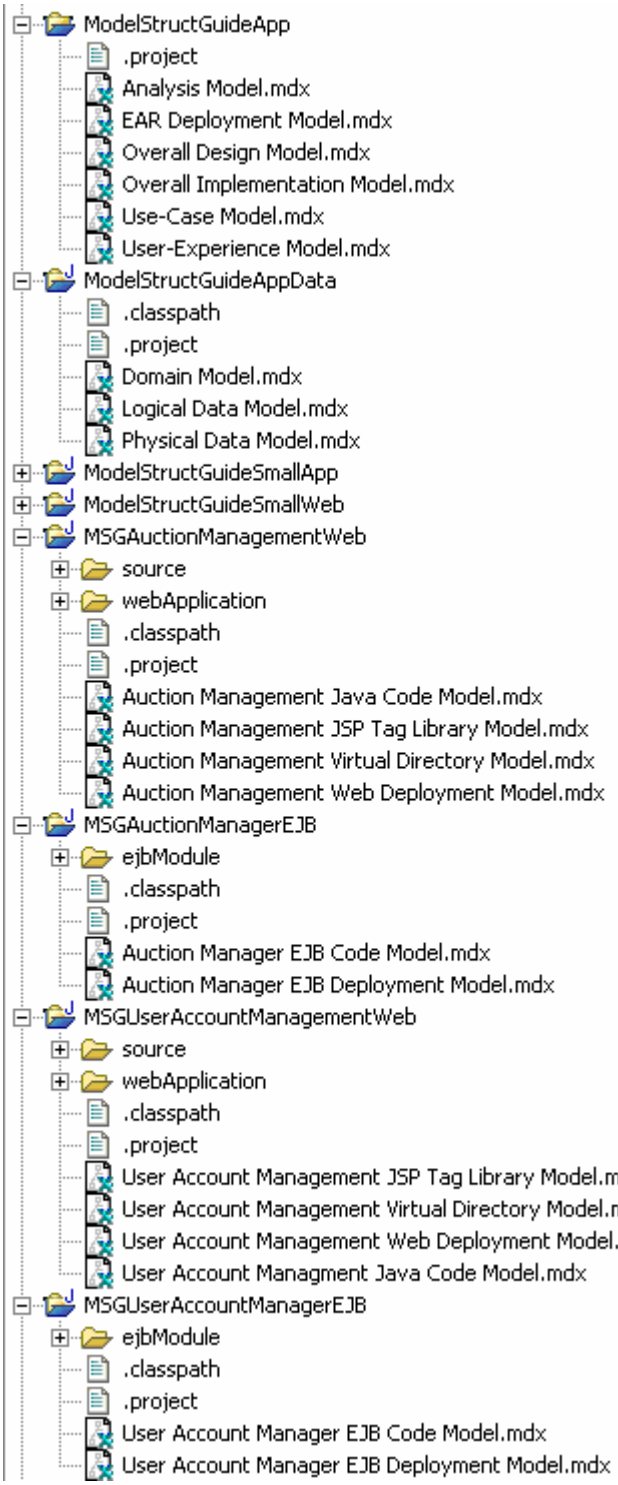

그림 1: XDE 프로젝트 및 모델 조직 예제

또는 응용프로그램의 크기가 작아 한 명의 작업자가 개발하려는 경우, 위의 프로젝트 구조를 두 개의 프로젝트로 단순화할 수 있습니다. 한 프로젝트에는 응용프로그램 전체 요소와 웹 이외의 요소가 포함되며 다른 프로젝트에는 웹 요소가 포함됩니다. 프로젝트 수뿐만 아니라 모델 수도 줄일 수 있습니다.

예를 들어, 한 명의 개발자가 담당하는 소규모 프로젝트의 경우, 다음과 같이 단순화할 수 있습니다.

- 별도의 분석 모델이 유지보수되지 않습니다. 분석 및 디자인은 모두 XDE 라운드트립 모델에서 수행됩니다.
- "전체 디자인 모델" 및 "전체 구현 모델"이 유지보수되지 않습니다. 프로젝트 규모가 작아 XDE 라운드트립 모델을 직접 보고 개요를 파악할 수 있습니다. 또한 유스 케이스 실현(realization)은 EJB 코드 모델에서 유지보수되며 가상 디렉토리 모델의 요소에 대한 참조가 포함됩니다.
- 별도의 논리 데이터 모델이 유지보수되지 않습니다. "실제 데이터 모델"에 실제 데이터 스키마가 직접 개발됩니다.

이러한 "소규모 프로젝트 구조"에 대한 요약 설명은 다음 표를 참조하십시오.

XDE 프로젝트	설명	XDE 모델 "<권장 모델 이름>" (<XDE 파일 유형: 모델 템플리트>)
응용프로그램 프로젝트(XDE EJB 모델링 프로젝트)	응용프로그램 프로젝트는 응용프로그램에서 웹 이외의 측면을 나타냅니다. 이 프로젝트에는 전체 응용프로그램에 대해 설명하는 모델, 데이터 모델 및 EJB 특정 모델이 포함됩니다.	– "유스 케이스 모델"(Rational XDE: 유스 케이스 모델) – "실제 데이터 모델"(데이터: *벤더 특정 실제 데이터 모델 파일*) – "EJB 코드 모델"(Java: EJB 코드 모델) – "EJB 배치 모델"(Java: EJB 배치 모델) – "EAR 배치 모델"(Java: EAR 배치 모델)
웹 프로젝트(XDE 웹 모델링 프로젝트)	웹 프로젝트는 응용프로그램의 웹 자원을 나타냅니다. 포함된 요소는 웹 아카이브 파일(WAR 파일)로 패키지화되어 배치됩니다.	– "Java 코드 모델"(Java: Java 1.3/1.4 코드 모델) – "JSP 태그 라이브러리 모델"(웹: JSP 태그 라이브러리 모델)[6] – "가상 디렉토리 모델"(웹: 가상 디렉토리 모델)[7] – "웹 배치 모델"(웹: 웹 배치 모델)

[6] 프로젝트마다 여러 태그 라이브러리 모델이 존재할 수 있습니다. 실제로는 각 .tld 파일마다 별도의 모델이 필요합니다. 이 문서 버전은 태그 라이브러리 개발에 대해서는 다루지 않습니다.

[7] XDE 웹 프로젝트당 여러 가상 디렉토리 모델이 존재할 수 있습니다.

소규모 프로젝트 및 모델 조직 예제는 그림 2에 표시됩니다.

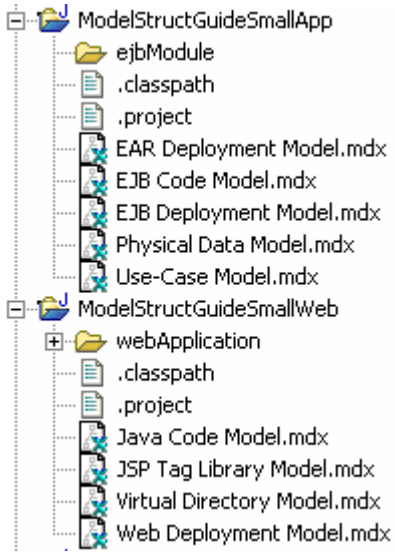

그림 2: 소규모 XDE 프로젝트 및 모델 조직 예제

프로젝트 및 개별 모델 파일 수는 아키텍처와 관련이 있는 선택사항이므로 프로젝트마다 다르게 선택할 수 있습니다. 그러나 정의되는 프로젝트 수와 관계 없이, XDE Java 코드 모델 파일은 프로젝트당 하나만 존재할 수 있습니다. 프로젝트 및 프로젝트에 포함될 수 있는 XDE 모델 파일에 대한 자세한 정보는 XDE 문서를 참조하십시오.

또한 *XDE 모델 이름은 모든 XDE 프로젝트에서 반드시 고유해야 합니다.* 이는 특히 XDE 모델 간 참조를 해석할 때 중요합니다. 모델 간 참조 및 참조 해석에 대한 자세한 정보는 XDE 문서를 참조하십시오.

이 문서 예제의 경우, 그림 1에 표시된 프로젝트 및 모델 구조가 사용됩니다. 여러 EJB 및 웹 프로젝트가 정의되었습니다. 여러 EJB 및 웹 프로젝트를 사용하는 근거는 구현 서브시스템 섹션을 참조하십시오.

4. RUP 모델 대 XDE 모델 맵핑

RUP 모델 아티팩트를 XDE 로 나타내는 방법을 설명하기 전에 "RUP 모델"과 "XDE 모델"의 차이를 명확히 해야 합니다. 이 모델은 서로 다른 모델이며 RUP 모델에서 연관된 XDE 모델로의 맵핑이 항상 일 대 일 맵핑은 아닙니다(유사한 경우라도 일 대 일 맵핑은 아닙니다). "모델"이라는 용어는 RUP 와 XDE 에서 모두 사용되므로 처음에는 동일한 개념으로 가정했습니다. 그러나 RUP 의 모델은 프로세스 관심사항을 분리(분석 대 디자인 대 구현 등)하며,XDE 의 모델은 개발 관심사항을 분리(프로그래밍 언어 패키징 구조 대 가상 디렉토리 구조를 설명하는 코드 모델 분리와, 여러 프로그래밍 언어 및 개발 환경 등에 대한 코드 모델 분리)합니다. 이 백서에서는 이러한 혼동을 줄이기 위해 "모델"이라는 용어를 "RUP" 또는 "XDE"와 함께 명시적으로 규정합니다.

다음 표는 RUP 모델에서 XDE 모델로의 맵핑을 요약 설명합니다. XDE 모델은 XDE 프로젝트 구조 섹션에서 설명하는 모델입니다. 각 XDE 모델의 구조는 이 백서의 후반부 섹션에서 설명합니다.

RUP 모델	<XDE 프로젝트>: < XDE 모델 이름>
유스 케이스 모델	응용프로그램 프로젝트: 유스 케이스 모델
분석 모델	응용프로그램 프로젝트: 분석 모델
디자인 모델	응용프로그램 프로젝트: 전체 디자인 모델 [각 XDE 라운드트립 모델 파일의 디자인 클래스(아래 참조)]
데이터 모델	XDE 데이터 모델: – 데이터 모델링 프로젝트: 논리 데이터 모델 – 데이터 모델링 프로젝트: *벤더 특정* 실제 데이터 모델 – 데이터 모델링 프로젝트: *벤더 특정* 도메인 모델
구현 모델	응용프로그램 프로젝트: 전체 구현 모델 XDE 라운드트립 모델[8] – EJB 프로젝트: EJB 코드 모델 – 웹 프로젝트: Java 코드 모델 – 웹 프로젝트: JSP 태그 라이브러리 모델 – 웹 프로젝트: 가상 디렉토리 모델
배치 모델	XDE 배치 모델[9] – 응용프로그램 프로젝트: EAR 배치 모델"[10] – EJB 프로젝트: EJB 배치 모델 – 웹 프로젝트: 웹 배치 모델

[8] 보다 쉬운 설명을 위해, 이 문서에서는 "XDE 라운드트립 모델"을 사용하여 이 XDE 모델을 나타냅니다.

[9] 보다 쉬운 설명을 위해, 이 문서에서는 "XDE 배치 모델"을 사용하여 이 XDE 모델을 나타냅니다.

[10] EAR 배치 모델은 개별 XDE 배치 모델과 "연결됩니다. 이 모델에는 배치 노드 및 노드 연결에 대해 설명하는 다이어그램이 포함됩니다. 또한 개별 배치 모델에 정의된 개별 아카이브 파일을 배치 노드에 맵핑하는 다이어그램이 포함됩니다.

5. 유스 케이스 모델

권장되는 "유스 케이스 모델" 구조는 그림 3에 표시됩니다.

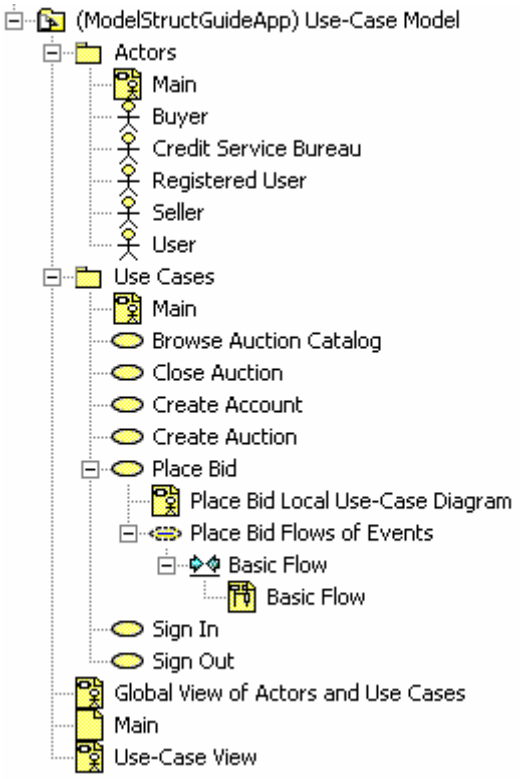

그림 3: "유스 케이스 모델" 구조

"유스 케이스 모델"은 "액터" 패키지와 "유스 케이스" 패키지로 파티션됩니다.

액터 및 **유스 케이스**를 포함하는 **유스 케이스 모델** 다이어그램 이외에 추가 다이어그램을 사용하여 **유스 케이스**의 다양한 측면을 명확히 할 수 있습니다. **유스 케이스 모델**의 **유스 케이스** 모델 요소 "아래"에 다음 보충 모델 요소가 포함될 수 있습니다. 그림 3:

- "입찰 참여 로컬 유스 케이스 다이어그램" 다이어그램에는 "입찰 참여" **유스 케이스**와 해당 **유스 케이스**에 참여하는 **액터**가 포함됩니다.
- "입찰 참여 이벤트 플로우" 협업 인스턴스에는 유스 케이스 설명에서 설명하는 이벤트 플로우(예: **액터**와 **유스 케이스** 간 상호작용)를 그래픽으로 설명하는 상호작용 인스턴스가 포함됩니다. 유스 케이스 협업 인스턴스와 **유스 케이스 실현(realization)**(분석 모델 섹션 및 디자인 유스 케이스 실현 섹션 참조)을 혼동해서는 안됩니다. "유스 케이스 모델"의 협업 인스턴스는 "블랙 박스"이므로 응용프로그램 내 요소 상호작용에 대해 설명하지 않습니다.
- "입찰 참여 이벤트 플로우" 활동 그래프에는 유스 케이스 설명에서 설명하는 이벤트 플로우를 그래픽으로 설명하는 활동 다이어그램이 포함됩니다.

그림 3의 예제에서, 그림 3 의 "액터 및 유스 케이스의 글로벌 보기" 다이어그램에는 모든 **유스 케이스** 및 **액터**와 해당 관계가 포함됩니다. 그러나 "기본" 다이어그램에는 "기본" 다이어그램이 존재하는 패키지의 요소만 포함됩니다. 여러 **액터** 및 **유스 케이스**가 있는 경우, "액터 및 유스 케이스의 글로벌 보기" 다이어그램에 대한 정보는 여러 다이어그램을 사용하여 표현할 수 있습니다.

"유스 케이스 보기" 다이어그램은 소프트웨어 아키텍처의 유스 케이스 보기를 나타냅니다. 아키텍처 보기에 대한 자세한 정보는 RUP 를 참조하십시오.

필요한 경우, 액터 및 유스 케이스 패키지에 추가 패키지를 작성하여 그림 4에 표시된 대로 포함된 모델 요소를 세부 구성할 수 있습니다.

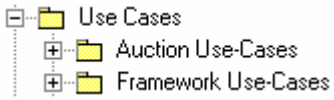

그림 4: 추가 유스 케이스 패키지 파티션

6. 분석 모델

분석 모델에는 **분석 클래스**와 분석 **유스 케이스 실현(realization)**이 표시됩니다.

참고: 별도의 **분석 모델** 및 **디자인 모델**의 유지보수 여부는 프로젝트에 따라 결정됩니다. 개별 **분석 모델**을 작성하지만 유지하지 않는 경우, **분석 클래스**가 해당 **디자인 모델** 파티션[11] 으로 이동되어 정제됩니다. 또 다른 옵션은 **디자인 모델에서 분석 클래스** 및 분석 **유스 케이스 실현(realization)**을 작성한 후 해당 디자인으로 발전시키는 것입니다. XDE 에서 **디자인 모델**을 나타내는 방법에 대한 자세한 정보는 the 디자인 모델 섹션을 참조하십시오.

권장되는 **분석 모델** 구조는 그림 5에 표시됩니다.

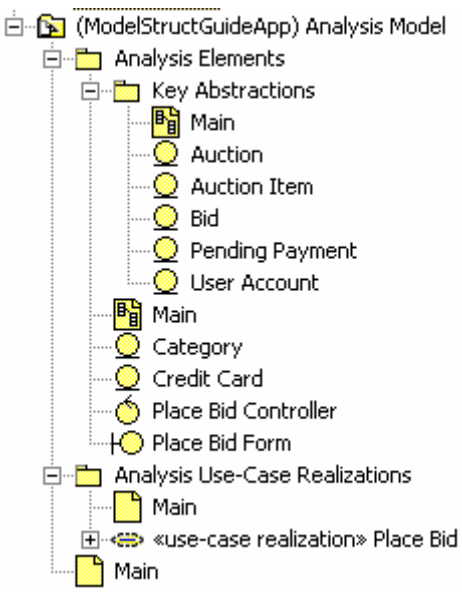

그림 5: 분석 모델 구조

"분석 요소" 패키지에는 **분석 클래스**가 포함됩니다. **분석 클래스**의 인스턴스는 "분석 유스 케이스 실현(realization)" 패키지의 다이어그램에 표시됩니다.

분석 클래스 이외에, "분석 요소" 패키지에 패키지를 정의하여 포함된 **분석 클래스**를 보다 세분화할 수 있습니다(그림 5의 "핵심 추상 패키지 참조). 이러한 추가 파티션은 특히 별도의 **분석 모델**을 유지보수하지 않으려는 경우 선택적입니다. 이러한 경우, **분석 클래스**를 "임시" 클래스로 간주할 수 있으므로(즉, 디자인 요소로 발전될 때까지만 존재하므로) 해당 조직은 중요하지 않습니다. 한 가지 예외는 핵심 추상 **분석 클래스**입니다.

그림 5와 같이, "핵심 추상" 패키지에는 시스템의 핵심 추상을 나타내는 것으로 간주되는 **분석 클래스**가 포함됩니다. 앞에서 설명한 대로 이 패키지는 선택적입니다. 한 가지 대안은 "분석 요소"

[11] 다음에 설명하겠지만 올바른 "디자인 모델 파티션"은 XDE 라운드트립 모델 중 하나의 패키지입니다. 기술 특정 요소의 디자인은 라운드트립 모델에서 수행되기 때문입니다.

패키지의 클래스 다이어그램에 핵심 추상을 나타내는 것입니다. 그러나 별도의 패키지를 작성함으로써 **분석 클래스**를 핵심 추상으로서 보다 명시적으로 분류할 수 있습니다. 실제로, 별도의 **분석 모델** 전체를 유지보수하지 않는 경우에도, 일부 프로젝트에서 핵심 추상 **분석 클래스**를 유지보수할 수 있습니다. 이러한 경우, 유지보수되는 **분석 클래스**를 포함할 별도의 패키지를 정의하는 것이 좋습니다.
참고: 핵심 추상은 "전체 디자인 모델"의 "논리 보기: 핵심 추상" 다이어그램에도 나타납니다. 자세한 정보는 디자인 모델 섹션을 참조하십시오.

"분석 유스 케이스 실현(realization)" 패키지에는 **유스 케이스** 수행 방법을 "분석 요소" 패키지의 **분석 클래스** 관점에서 설명하는 분석 레벨 **유스 케이스 실현**이 포함됩니다. 각 분석 **유스 케이스 실현**은 **유스 케이스 모델**의 **유스 케이스**를 실현하고 해당 **유스 케이스**와 동일한 이름을 가지며 그림 6에 표시된 구조를 갖습니다.

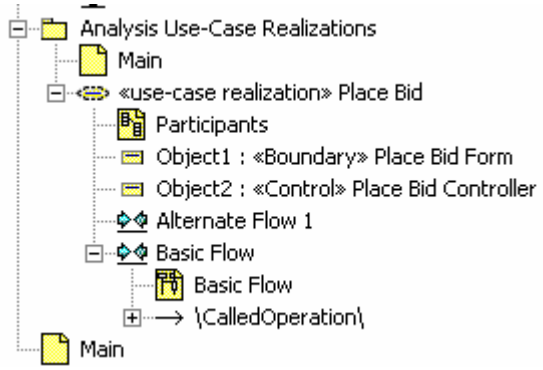

그림 6: "분석 유스 케이스 실현(realization)" 패키지 구조

"참가자" 다이어그램에는 **유스 케이스 실현(realization)**에 참여하는 **분석 클래스**("분석 요소" 패키지에 포함), 즉 해당 인스턴스가 상호작용 다이어그램에 나타나는 **분석 클래스**와, 상호작용 다이어그램에서 설명하는 협업을 지원하는 관계가 표시됩니다.

"플로우" 상호작용 인스턴스("기본 플로우" 및 "대체 플로우 1")에는 이벤트의 **유스 케이스** 플로우에 대해 설명하는 시퀀스 다이어그램이 포함됩니다. 중요한 이벤트 유스 케이스 플로우 각각에 대해 하나의 상호작용 인스턴스가 필요합니다. 상호작용 인스턴스의 시퀀스 다이어그램은 연관된 **유스 케이스** 실행 시 참여 **분석 클래스** 간의 플로우에 대해 설명합니다.

7. 디자인 모델

RUP **디자인 모델**은 여러 XDE 모델(즉, "전체 디자인 모델"과, 별도의 XDE 라운드트립 모델에 상주하는 라운드트립 디자인 요소)로 나타냅니다. 라운드트립 디자인 요소는 라운드트립 엔지니어링에 참여하는 세부 디자인 요소입니다. 개별 라운드트립 모델에서 사용 가능한 자동화는 이러한 방식으로 활용할 수 있습니다. 예를 들어, XDE EJB 패턴은 EJB 를 지정하는 클래스를 작성하는 데 사용할 수 있습니다.

"전체 디자인 모델"은 응용프로그램의 전체 디자인에 대해 설명하며 여러 XDE 라운드트립 모델에 적용되는 요소가 포함됩니다. 이 모델에는 개별 라운드트립 모델 조직의 기반이 되는 논리 파티션과, 모든 요소를 통합하는 **유스 케이스 실현(realization)**이 포함됩니다. **유스 케이스 실현**은 여러 라운드트립 모델의 디자인 요소 간 협업에 대해 설명합니다. "전체 디자인 모델"에는 라운드트립 디자인 요소를 참조하는 다이어그램이 포함됩니다. 개별 XDE 라운드트립 모델에 대한 정보는 구현 모델섹션을 참조하십시오.

또는 동일한 XDE 코드 모델에 **디자인 모델**과 **구현 모델**을 나타낼 수도 있습니다. 이는 소규모 팀에서 대상 구현 언어가 하나인 경우에만 가능합니다. 소규모 프로젝트 구조 예제는 XDE 프로젝트 구조 섹션을 참조하십시오.

"전체 디자인 모델"의 유지보수는 선택적이지만 다이어그램을 구성하고 추상 레벨 등을 향상시키며,

적용할 구현 메커니즘을 지정하면서 동시에 디자인 요소의 위치를 제공하기 위해서는 이 모델을 유지보수하는 것이 좋습니다.

권장되는 "전체 디자인 모델" 구조는 그림 7에 표시됩니다.

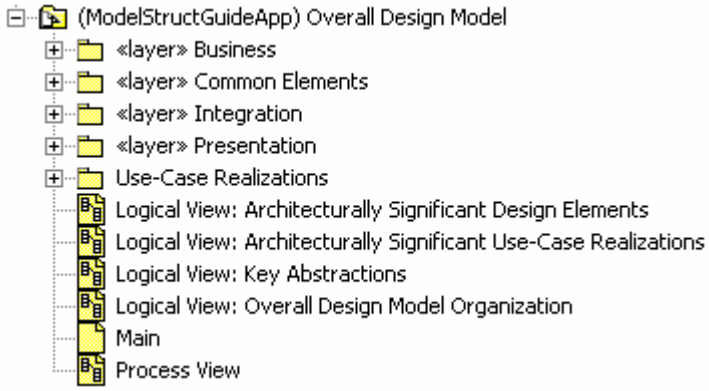

그림 7: 전체 디자인 모델 구조

전체 디자인 모델에는 다음과 같은 패키지가 포함됩니다.

- 계층 패키지에는 시스템의 디자인 요소(**디자인 클래스**, **인터페이스** 및 **디자인 서브시스템**) 또는 참조 다이어그램이 포함됩니다. 이 구조는 디자인 계층 섹션에서 설명하는 특정 파티션 전략을 나타냅니다.
- "유스 케이스 실현(realization)" 패키지에는 디자인 레벨 유스 케이스 실현이 포함됩니다. 유스 케이스 실현의 내부 구조는 디자인 유스 케이스 실현 섹션에서 자세히 설명합니다.

아키텍처 보기를 나타내는 다이어그램에는 다이어그램 이름의 "보기"가 포함됩니다. 아키텍처 보기에 대한 자세한 정보는 RUP 를 참조하십시오.

"논리 보기: 핵심 추상" 다이어그램에는 시스템의 핵심 추상이 포함됩니다. 이러한 핵심 추상을 유지보수하는 데는 다음과 같은 여러 가지 옵션이 있습니다.

- 전체 **분석 모델**이 유지보수됩니다. 이러한 경우, "논리 보기: 핵심 추상" 다이어그램에는 **분석 모델**에서 시스템의 핵심 추상을 나타내는 **분석 클래스**가 포함됩니다.
- 부분 **분석 모델**, 즉 핵심 추상만 유지보수됩니다. 이러한 경우, "논리 보기: 핵심 추상" 다이어그램에는 **분석 모델**에서 시스템의 핵심 추상을 나타내는 **분석 클래스**가 포함됩니다.
- **분석 모델** 파트가 유지보수되지 않습니다. 이러한 경우, 핵심 추상을 나타내는 **분석 클래스**는 **디자인 모델**의 "핵심 추상" 패키지에 유지보수할 수 있습니다.

분석 모델에 대한 자세한 정보는 분석 모델 섹션을 참조하십시오.

7.1 디자인 계층

계층 패키지에는 **분석 클래스**에서 발전되는 시스템 디자인 요소(예: **디자인 클래스**, **인터페이스** 및 **디자인 서브시스템**)가 포함됩니다. 계층 패키지에는 포함된 디자인 요소를 보다 세분화하는 원하는 수의 서브패키지가 포함될 수 있습니다. 디자인 **유스 케이스 실현(realization)**("디자인 모델"의 "유스 케이스 실현" 패키지에 포함되며 디자인 유스 케이스 실현 섹션의 표제에서 설명)은 이 패키지에 포함된 디자인 요소 관점에서 작성됩니다.

디자인 모델은 여러 파티션 전략을 따를 수 있습니다. 이 섹션에서 설명하는 파티션 전략은 그림 8에 표시됩니다.

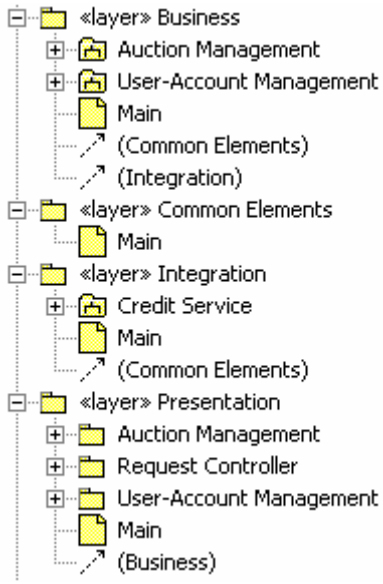

그림 8: 디자인 패키지 파티션 예제

이 예제에서 첫 번째 레벨 패키지는 계층으로 간주되며 각 계층마다 특정 책임이 있습니다. 두 번째 레벨 패키지는 계층 패키지 요소를 비즈니스 기능성에 따라 보다 세분화합니다.

"프리젠테이션" 계층 패키지는 일반 사용자와의 상호작용을 처리합니다. J2EE 응용프로그램에서, "프리젠테이션" 계층 패키지에 상주하는 디자인 요소에는 JSP(Java Server Pages)가 포함됩니다. "프리젠테이션" 계층 패키지를 서브패키지로 보다 세분화하여 관련 **유스 케이스** 세트(예: 그림 8의 경매 관리 패키지)에 속하는 요소를 그룹화할 수 있습니다.

"비즈니스" 계층 패키지는 비즈니스 처리를 수행합니다. 이 문서에서 설명하는 "디자인 모델" 구조의 경우, "비즈니스" 계층 패키지는 주요 비즈니스 기능당 하나씩 한 세트의 디자인 서브시스템 패키지(예: 그림 8의 경매 관리 및 사용자 계정 관리, 서브시스템 패키지)로 구성됩니다. **디자인 서브시스템** 패키지는 디자인 서브시스템 섹션의 표제에서 자세히 설명합니다.

"통합" 계층 패키지는 데이터베이스 및 외부 시스템을 포함하여 백엔드 자원에 대한 액세스를 제공합니다. 이 문서에서 설명하는 **디자인 모델** 구조의 경우, "통합" 계층 패키지 또한 외부 시스템당 하나씩 디자인 서브시스템 패키지(예: 그림 8의 신용 서비스 서브시스템 패키지)로 구성됩니다. **디자인 서브시스템** 패키지는 디자인 서브시스템 섹션의 표제에서 자세히 설명합니다.

"공통 요소" 계층 패키지에는 계층에서 공유되는 요소가 포함됩니다.

이 섹션에서 설명하는 구조는 다른 파티션 전략을 반영하는 다른 구조로 바꿀 수 있습니다.

7.2 디자인 서브시스템

디자인 서브시스템은 "전체 디자인 모델"에서 서브시스템 패키지로 표시됩니다. 각 디자인 서브시스템 패키지는 구조가 동일해야 합니다. 해당 구조의 스펙은 **디자인 서브시스템**에 대해 캡처되는 세부사항 레벨에 따라 다릅니다.

보다 견고한 정규 **디자인 서브시스템** 구조의 예제가 그림 9에 표시됩니다.

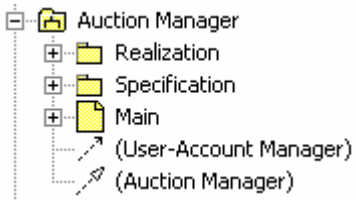

그림 9: 디자인 서브시스템 구조

이 디자인 서브시스템 패키지 구조는 디자인 서브시스템 패키지 내부의 별도의 "스펙" 및 "실현(realization)" 패키지의 정의를 지원합니다. 이 구조는 *UML Components: A Simple Process for Specifying Component-Based Software*(저자: J. Cheesman, J. Daniels)의 영향을 받았습니다. 이러한 파티션을 포함하지 않는 단순 디자인 서브시스템 패키지 구조는 이 문서에 정의된 다른 모델 파일 구조에 영향을 주지 않고 사용할 수 있습니다. 각 "스펙" 및 "실현" 패키지는 다음 섹션에서 설명합니다.

7.2.1 서브시스템 스펙

"스펙" 패키지에는 **디자인 서브시스템** 인터페이스의 설명이 포함됩니다. [12] 서브시스템 스펙 예제가 그림 10에 표시됩니다.

그림 10: 디자인 서브시스템 스펙 예제

7.2.2 서브시스템 실현(realization)

"실현(realization)" 패키지에는 **디자인 서브시스템** 스펙을 실현하는 방법에 대한 설명이 포함됩니다. 디자인 서브시스템 패키지의 "실현" 패키지 예제가 그림 11에 표시됩니다.

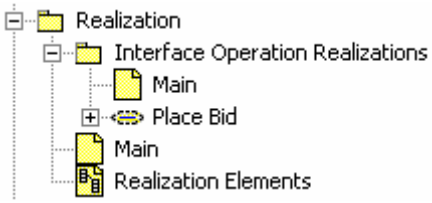

그림 11: 디자인 서브시스템 실현(realization) 예제

"실현(realization) 요소" 다이어그램에는 서브시스템을 실현하는 디자인 요소에 대한 참조가 포함됩니다. 디자인 요소는 라운드트립 엔지니어링에 참여하는 별도 XDE 코드 모델이나 "실현(realization)" 패키지에 상주할 수 있습니다. 자세한 정보는 XDE 라운드트립 모델 섹션을 참조하십시오.

[12] 이 단순 예제에서 단지 인터페이스를 위해 별도의 패키지가 필요한지 의문을 가질 수 있습니다. 그러나 실제 프로젝트에서는 해당 패키지를 유지보수할 필요가 있습니다. 이 패키지는 서브시스템과 특히 오퍼레이션에 대한 전제 조건 및 사후 조건과 같은 인터페이스 제한조건에 대해 설명하는 문서에 대한 참조를 포함할 수 있기 때문입니다.

"인터페이스 오퍼레이션 실현(realization)"에는 서브시스템 요소가 "스펙" 패키지에 포함된 **디자인 서브시스템** 인터페이스의 중요한 오퍼레이션을 실현하는 방법을 설명하는 협업 인스턴스가 포함됩니다. 중요한 서브시스템 인터페이스 오퍼레이션마다 하나의 협업 인스턴스가 존재합니다. [13] "인터페이스 오퍼레이션 실현 " 예제가 그림 12에 표시됩니다.

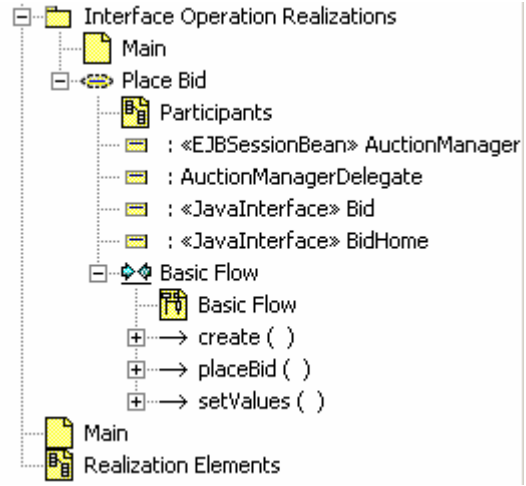

그림 12: 인터페이스 오퍼레이션 실현(realization) 패키지 예제

분석 레벨 **유스 케이스 실현(realization)**(앞부분의 분석 모델 섹션 참조)과 디자인 레벨 **유스 케이스 실현**(뒷부분의 디자인 유스 케이스 실현 섹션 참조)과 같이, 각 인터페이스 오퍼레이션 실현에는 실현에 참여하는 서브시스템 요소를 포함하는 클래스 다이어그램(그림 12의 "참가자" 다이어그램)과, 해당 참가자가 서브시스템 인터페이스 오퍼레이션을 수행하기 위해 협업하는 방법에 대해 설명하는 상호작용 다이어그램(그림 12의 "기본 플로우" 다이어그램)이 포함됩니다.

7.3 디자인 유스 케이스 실현(realization)

"유스 케이스 실현(realization)" 패키지에는 디자인 레벨 **유스 케이스 실현**이 포함됩니다. 각 **유스 케이스 실현**은 **유스 케이스 모델**의 **유스 케이스**와 연관되며 해당 **유스 케이스**와 동일한 이름을 갖습니다. 또한 그림 16 의 구조를 나타내야 합니다.

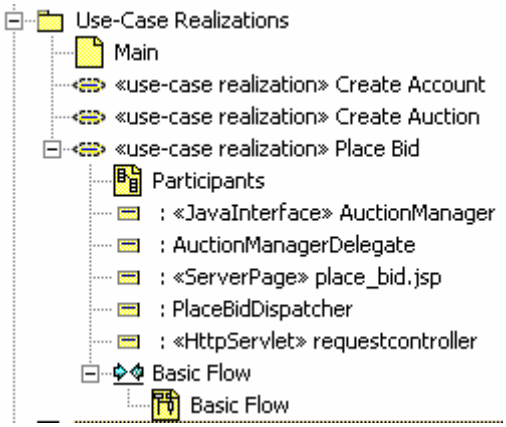

그림 13: 디자인 유스 케이스 실현(realization) 구조

유스 케이스 실현(realization) "참가자" 다이어그램에는 **유스 케이스 실현**에 참여하는 디자인 요소(즉, 해당 인스턴스가 **유스 케이스 실현** 상호작용 다이어그램에 표시되는 디자인 요소)와, 상호작용

[13] 이 레벨에서 모든 오퍼레이션을 정의할 필요는 없으며 일부 단순 오퍼레이션의 경우 별도의 협업 인스턴스가 필요하지 않습니다.

다이어그램에서 설명하는 협업을 지원하는 관계가 표시됩니다.

"기본 플로우" 다이어그램은 연관된 **유스 케이스** 실행 시 참여 디자인 요소 간 플로우를 설명하는 상호작용 다이어그램의 예제입니다. **유스 케이스**의 각 이벤트 플로우마다 하나의 상호작용 인스턴스가 필요합니다.

유스 케이스 실현(realization 다이어그램에는 별도의 XDE 라운드트립 모델에 실제로 상주하는 디자인 요소에 대한 참조를 포함할 수 있습니다(일반적으로는 실제로 포함합니다). **유스 케이스 실현**(realization)에는 별도의 라운드트립 모델의 요소 간 협업이 표시됩니다.

8. 데이터 모델

RUP **데이터 모델**은 다음과 같은 여러 XDE 모델 파일로 표시됩니다.

- **논리 데이터 모델**(선택사항). 데이터베이스 논리 디자인의 응용프로그램 독립 보기인 논리 데이터 모델을 나타냅니다.
- **실제 데이터 모델**. 데이터베이스 벤더 특정 실제 데이터 모델을 나타냅니다. 이 모델에는 데이터베이스 테이블의 특정 특성을 정의하기 위한 세부 모델 요소가 포함됩니다. "실제 데이터 모델" XDE 모델 파일에는 또한 벤더 특정 데이터베이스의 테이블을 구현하기 위한 데이터베이스 특정 구현 아티팩트가 포함됩니다.
- **도메인 모델**(선택사항). "실제 데이터 모델"에서 일관적인 데이터 유형을 정의하는 데 사용되는 데이터베이스 벤더 특정 데이터 유형을 나타냅니다.

XDE 모델 파일을 분리함으로써 **전체 디자인 모델**, **데이터 모델** 및 실제 데이터베이스 간에 지원되는 자동화에 대한 최적의 유연성을 제공합니다.

이러한 XDE 모델 파일 각각은 아래에서 자세히 설명합니다.

8.1 논리 데이터 모델 (선택사항)

논리 데이터 모델은 프로젝트에서 데이터베이스 디자인에 중요한 핵심 엔티티 및 관계에 대한 독립형 논리 데이터 표시를 작성해야 하는 상황에서 사용할 수 있습니다. XDE 논리 데이터 모델 작성은 선택적입니다. 데이터베이스 디자인 팀에서 XDE 실제 데이터 모델에 직접 초기 실제 데이터베이스 디자인 구조를 작성하기 위해 대신 **디자인 모델**의 지속적 **디자인 클래스**를 **데이터 모델**의 테이블로 변환할 수 있기 때문입니다(아래 실제 데이터 모델 섹션 참조).

XDE 논리 데이터 모델은 필요에 따라 주제 영역 패키지로 파티션될 수 있습니다. 주제 영역 패키지는 엔티티 클래스의 논리 그룹을 정의합니다. XDE 논리 데이터 모델에는 또한 여러 주제 영역에 포함되는 모델 요소를 포함하는 "공통 요소" 패키지가 포함될 수 있습니다.

이름에 "보기"가 포함되는 다이어그램은 아키텍처의 데이터 보기를 문서화하는 데 사용됩니다. "데이터 보기: 전체 논리 데이터 모델 조직" 다이어그램은 XDE 논리 데이터 모델의 주요 파티션(예: 패키지)에 표시되는 대로, 논리 데이터 모델의 상위 레벨 데이터 조직을 문서화하는 데 사용됩니다. "데이터 보기: 핵심 논리 데이터 요소"는 **데이터 모델**의 핵심 논리 요소를 문서화하는 데 사용됩니다. 아키텍처 보기에 대한 자세한 정보는 RUP 를 참조하십시오.

권장되는 논리 데이터 모델 구조의 예제가 그림 14에 표시됩니다.

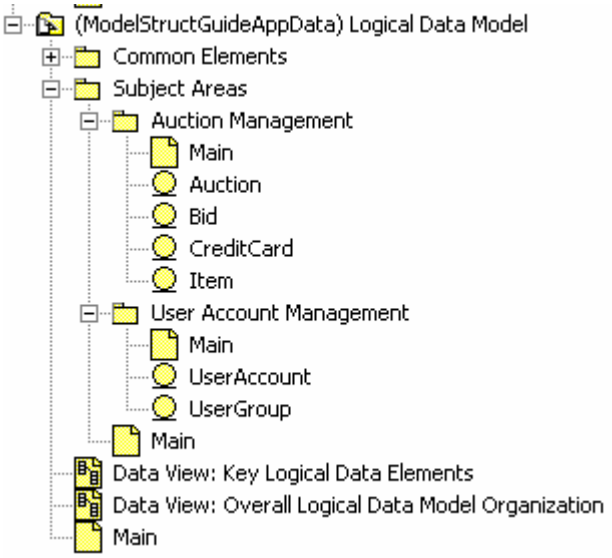

그림 14: 논리 데이터 모델 구조

이 예제에는 "경매 관리" 및 "사용자 계정 관리" 주제 영역 패키지가 있습니다. 각 주제 영역 패키지에는 논리 데이터 모델을 구성하는 엔티티 클래스가 포함됩니다. 일부 유사한 점도 있지만 **디자인 모델**의 패키지 구조에 대한 직접 맵핑은 없습니다.

8.2 실제 데이터 모델

실제 데이터 모델에는 XDE 데이터 모델러 포워드 엔지니어링 기능을 통해 데이터베이스를 구현하는 데 사용되는 세부 데이터베이스 테이블 및 스토어드 프로시저 디자인이 포함됩니다. 실제 데이터 모델은 또한 데이터베이스의 실제 저장영역 구성을 정의하는 데 사용되는 모델 요소로 구성됩니다. 일반적으로, 모델 요소에는 대상 저장 매체에서 데이터베이스 테이블의 실제 레이아웃을 구성하는 데이터베이스 및 테이블 공간이 포함됩니다.

실제 데이터 모델을 작성하는 경우, 데이터베이스 디자이너가 해당 대상 데이터베이스를 선택해야 합니다. 지원되는 데이터베이스는 DB2 MVS, DB2 UDB, Oracle, Sybase 및 SQL Server 입니다. XDE 에서 XDE 모델 파일의 기본 이름은 선택한 데이터베이스의 이름입니다. 이 문서의 "실제 데이터 모델" 예제에서, XDE 모델 파일 이름은 "실제 데이터 모델"로 갱신되었습니다. 데이터베이스 디자이너는 "실제 데이터 모델"을 작성할 때 기본 이름을 허용할 수 있습니다.

권장되는 실제 데이터 모델 구조의 예제가 그림 15에 표시됩니다.

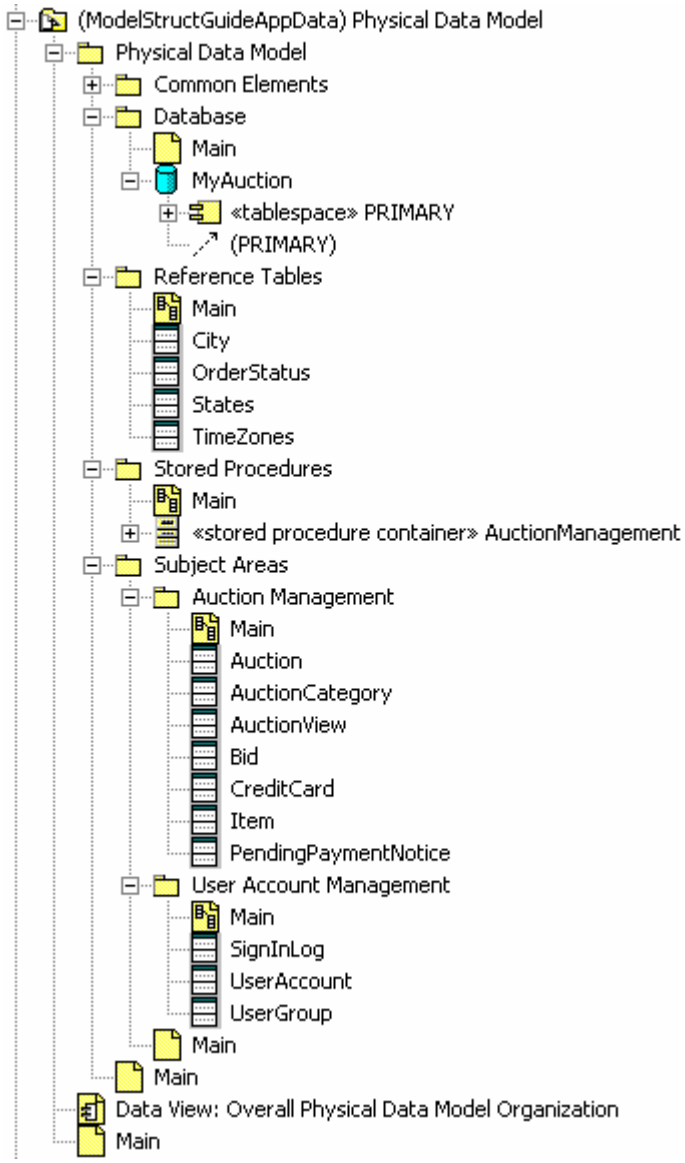

그림 15: "실제 데이터 모델" 구조

"공통 요소" 패키지에는 여러 주제 영역에 포함되는 데이터베이스 테이블 및 보기가 포함됩니다.

"데이터베이스" 패키지에는 데이터베이스의 실제 저장영역 구성을 정의하는 모델 요소가 포함됩니다. 이 패키지에는 대상 저장 매체에서 데이터베이스 테이블의 실제 레이아웃을 구성하는 데이터베이스 및 테이블 공간이 포함됩니다. 테이블 공간은 데이터베이스에서 테이블 그룹을 논리적으로 지정하는 데 사용됩니다. 테이블 공간 정의에 대한 가이드라인은 RUP 를 참조하십시오. "데이터베이스" 패키지는 응용프로그램의 복잡도에 따라 필요한 대로 하위 레벨 패키지로 파티션할 수 있습니다.

그림 15의 예제에서, "데이터베이스" 패키지에는 단일 데이터베이스인 MyAuction, 이 데이터베이스의 연관 테이블 공간인 PRIMARY 및 테이블 실현(realization) 관계가 포함됩니다. 테이블 공간의 이름은 데이터베이스 프로젝트의 적절한 이름으로 지정할 수 있습니다. MyAuction 데이터베이스의 경우, 하나의 PRIMARY 테이블 공간만 정의됩니다. 포워드 엔지니어링을 수행하는 경우, 데이터베이스 테이블 공간과의 실현 관계를 통해 데이터베이스에 링크된 테이블이 작성됩니다(데이터베이스 또는 DDL).

"참조 테이블" 패키지에는 응용프로그램에서 필요한 "고정" 데이터 정보가 들어 있는 정적 데이터

테이블이 포함됩니다.

"스토어드 프로시저" 패키지에는 데이터베이스 스토어드 프로시저를 나타내는 모든 클래스(스토어드 프로시저 컨테이너 클래스 및 연관 스토어드 프로시저 오퍼레이션)가 포함됩니다. 단일 테이블과 연결되는 스토어드 프로시저는 "스토어드 프로시저 중심" 또는 "테이블 중심" 보기[14]를 나타낼 지 여부에 따라 스토어드 프로시저 패키지 또는, 스토어드 프로시저가 참조하는 테이블을 가진 주제 영역 패키지에 패키지화할 수 있습니다.

"주제 영역" 패키지에는 논리적으로 관련이 있는 테이블 및 보기 세트를 그룹화하는 패키지가 포함됩니다.[15] 보기는 테이블과 함께 주제 영역 패키지에 작성하도록 권장합니다. 이 권장사항은 구성 측면에서 유용합니다. 즉, 테이블과 동일한 주제 영역에 보기를 배치하여 보기를 사용하는 주제 영역에서 해당 보기를 쉽게 보관할 수 있습니다. 그림 15의 예제에는 "경매 관리" 및 "사용자 계정 관리" 주제 영역 패키지가 있습니다. 주제 영역 패키지의 수는 응용프로그램의 복잡도에 따라 다릅니다. 그러나 일반적으로, 논리 데이터 모델의 주제 영역 패키지가 실제 데이터 모델의 주제 영역 패키지의 "기반"이 됩니다. 논리 데이터 모델의 주제 영역은 실제 데이터 모델 주제 영역의 추상입니다.

주제 영역 패키지의 테이블에는 테이블에 정의된 열과 트리거가 포함됩니다. 이 테이블은 다음 중 한 가지 방법으로 작성됩니다.

- XDE 클래스 대 테이블 변환 기능.
- 기존 데이터베이스에 대한 XDE 리버스 엔지니어링 기능[16]
- 데이터베이스 디자이너에 의한 수동 작성

기존 데이터베이스에 대한 리버스 엔지니어링을 수행하면 XDE 실제 데이터 모델에 스키마 패키지가 작성됩니다. 이러한 패키지의 이름은 리버스 엔지니어링이 수행되는 데이터베이스의 데이터베이스 소유자[17] 를 기반으로 합니다. 리버스 엔지니어링이 수행된 테이블은 "주제 영역" 패키지 내부의 주제 영역 패키지로 이동하고 리버스 엔지니어링이 수행된 스키마 패키지는 삭제하도록 권장합니다. 테이블을 주제 영역 패키지로 이동함으로써, 데이터베이스 디자이너가 필요에 따라 테이블을 갱신할 수 있도록 테이블을 구성할 수 있습니다.

이름에 "보기"가 포함되는 다이어그램은 아키텍처의 데이터 보기를 문서화하는 데 사용됩니다. "데이터 보기: 전체 실제 데이터 모델 조직" 다이어그램은 XDE 실제 데이터 모델의 주요 파티션(예: 패키지)에 표시되는 대로, 실제 데이터 모델의 상위 레벨 데이터 조직을 문서화하는 데 사용됩니다. 아키텍처 보기에 대한 자세한 정보는 RUP 를 참조하십시오.

8.3 도메인 모델(선택사항)

도메인 모델은 데이터베이스의 사용자 정의 데이터 유형을 저장하는 데 사용되는 선택적 XDE 모델입니다. 데이터베이스 디자이너는 도메인을 사용하여 데이터베이스 디자인에서 요소 특성을

---

[14]테이블 중심 보기에서는 모든 데이터베이스 디자인/오퍼레이션을 하나의 보기에서 보다 잘 이해할 수 있습니다. 스토어드 프로시저 중심 보기에서는 스토어드 프로시저를 쉽게 찾아 변경/유지할 수 있습니다.

[15]논리 및 실제 데이터베이스 주제 영역 패키지를 유지하기 위한 추가 유지보수 작업이 필요하다는 이유로 실제 데이터 모델에서의 주제 영역 패키지 사용에 대해 의문을 가질 수도 있습니다. 여기서 실제 데이터 모델의 주제 영역은 논리 데이터 모델(사용하는 경우)과의 일관성을 위해, 또한 실제 데이터 모델이 "대규모" 모델이고 논리 데이터 모델이 없는 경우에 사용됩니다. 이러한 경우 주제 영역 패키지를 사용하여 클래스에서 테이블로의 변환에서 생성된 테이블을 관리할 수 있습니다.

[16] 일반적으로 데이터베이스에 대한 리버스 엔지니어링이 한 번 수행된 후 XDE 의 비교 및 동기화 기능을 사용하여 모든 향후 갱신사항이 동기화됩니다.

[17]XDE 에서는 데이터베이스 소유자가 <<database>> 컴포넌트의 특성으로 캡처됩니다. 위치 특성 내부에는 스키마 속성이 연결 문자열의 일부로 존재합니다. 데이터베이스에 대한 리버스 엔지니어링을 수행하는 경우, 이는 일반적으로 데이터베이스 소유자입니다.

재사용할 수 있습니다. 즉, 데이터베이스 디자이너는 도메인을 사용하여 전체 데이터베이스에서 열의 특성을 지속적으로 문서화합니다. 열의 이름은 테이블에 정의되며 도메인을 사용하여 열의 *TypeExpression* 을 정의합니다.

도메인 모델의 권장 구조 예제[18] 가 그림 14에 표시됩니다.

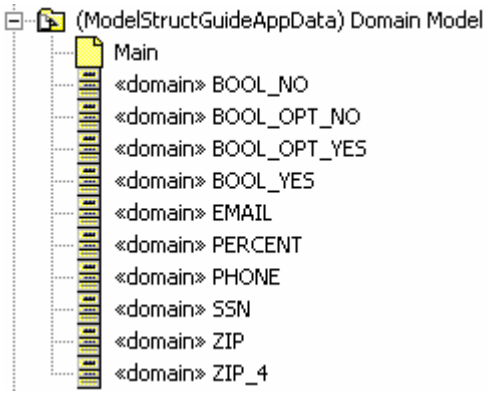

그림 16: "도메인 모델" 구조

이 예제에서는 SQL Server 도메인 값이 "SQL Server 도메인" 패키지에 구성됩니다. 데이터베이스 디자이너가 여러 도메인을 정의하는 경우, 해당 데이터베이스 디자이너는 "SQL Server 도메인" 패키지의 패키지를 사용하여 도메인을 구성해야 합니다.

9. 구현 모델

RUP **구현 모델**은 여러 XDE 모델("전체 구현 모델" 및 여러 XDE 라운드트립 모델)로 표시됩니다.[19] 여러 라운드트립 모델을 사용함으로써 Java 패키징 구조 대 가상 디렉토리 구조 설명과 같은 다양한 구현 관심사항을 가장 효과적으로 나타낼 수 있습니다. 또한 개별 라운드트립 모델에서 사용 가능한 자동화를 활용할 수 있습니다. 예를 들어, XDE 는 구현 기술 특정 모델 요소 작성과, 라운드트립 엔지니어링을 사용한 해당 모델 요소와 소스 코드 동기화를 위한 자동화를 제공합니다.

"전체 구현 모델"은 응용프로그램의 전체 구현에 대해 설명하며 여러 라운드트립 모델에 적용되는 요소가 포함됩니다. 이 모델에는 라운드트립된 디자인 요소의 요소를 참조하는 다이어그램이 포함되며 일반적으로 모델 요소를 "소유"하지 않습니다. "전체 구현 모델"은 XDE 라운드트립 엔지니어링에 참여하지 않습니다.

"전체 구현 모델"을 유지보수하는 것은 선택적입니다. 그러나 이 모델은 통합 단위(RUP 에서 **구현 서브시스템**으로 정의)을 포함한 전체 구현 구조에 대해 설명하고 아키텍처의 구현 보기를 문서화하는 데 유용합니다. **구현 서브시스템**은 구현 서브시스템 섹션에서 자세히 설명합니다.

XDE 라운드트립 모델의 수는 정의된 XDE 프로젝트 및 모델 조직에 따라 결정됩니다(자세한 정보는 XDE 프로젝트 구조 섹션을 참조하십시오). 그러나 한 가지 옵션은 각 **구현 서브시스템**마다 별도의 프로젝트(및 라운드트립 모델)를 정의하는 것입니다. XDE 에서 **구현 서브시스템을** 나타내는 방법에 대한 정보는 구현 서브시스템 섹션을 참조하십시오. 또는 앞에서 설명한 것처럼, 소규모 팀에서 단일 대상 구현 언어를 사용하는 경우 단일 XDE 라운드트립 모델을 사용하여 RUP **디자인 모델**과 **구현 모델**을 모두

[18] XDE 에서는 DB2, Oracle, Sybase 및 SQL Server 와 같은 여러 벤더 데이터베이스가 지원됩니다. 도메인 XDE 데이터 모델을 작성하는 경우, 데이터베이스 디자이너는 해당 벤더 데이터베이스를 선택하여 도메인 XDE 데이터 모델을 작성합니다. XDE 는 선택한 데이터베이스 벤더의 기본 도메인 목록을 작성합니다.

[19] XDE 라운드트립 모델은 혼성 모델입니다. 이 모델은 RUP **디자인 모델**과 RUP **구현 모델**을 나타내는 데 사용됩니다. XDE 라운드트립 모델의 모델 요소는 RUP **디자인 클래스**(실제 클래스로 직접 맵핑되는 클래스를 **디자인 클래스**로 간주)와 실제 구현 파일을 나타냅니다. XDE 라운드트립 모델의 구조는 실제 디렉토리 구조를 나타냅니다.

나타낼 수 있습니다.

"전체 구현 모델" 예제가 그림 17에 표시됩니다.

그림 17: 전체 구현 모델 구조

그림 17과 같이 "전체 구현 모델"에는 다이어그램만 포함됩니다. 아키텍처 보기를 나타내는 다이어그램에는 다이어그램 이름의 "보기"가 포함됩니다. 아키텍처 보기에 대한 자세한 정보는 RUP 를 참조하십시오.

"구현 보기: 배치 가능한 구현 요소" 다이어그램은 XDE 배치 모델의 노드로 배치될 아카이브 파일을 참조합니다. 아카이브 파일은 실제로 배치 모델에 포함됩니다. 자세한 정보는 배치 모델 섹션을 참조하십시오.

"구현 보기: 구현 서브시스템" 다이어그램은 응용프로그램의 구현 서브시스템을 참조합니다. 다이어그램에서 **구현 서브시스템** 간의 종속성 관계를 나타낼 수 있습니다. 이러한 종속성은 **구현 서브시스템**을 통합해야 하는 순서를 결정하는 **구현 서브시스템** 가져오기를 나타냅니다. XDE 에서 구현 서브시스템을 나타내는 방법에 대한 정보는 구현 서브시스템 섹션을 참조하십시오.

"구현 보기: 구현 모델 구조" 다이어그램에는 **구현 모델**을 나타내는 데 사용된 모든 XDE 모델에 대한 참조와 해당 관계가 포함됩니다.
참고: 각 **구현 서브시스템**마다 개별 프로젝트를 정의하는 경우, "구현 보기: 구현 모델 구조" 다이어그램의 컨텐츠가 "구현 보기: 구현 서브시스템" 다이어그램과 중복되어 생략할 수 있습니다. 구현 서브시스템에 대한 자세한 정보는 구현 서브시스템 섹션을 참조하십시오.

9.1 구현 서브시스템

RUP **구현 서브시스템**은 통합 단위입니다.[20] 따라서 이 서브시스템은 J2EE 모듈과 효과적으로 통합됩니다. **구현 서브시스템**은 J2EE 모듈에도 패키지화되어 배치될 수 있습니다. XDE 프로젝트 구조를 설정하려면 각 J2EE 아카이브마다 XDE 프로젝트를 정의해야 하므로 식별된 **구현 서브시스템**을 사용하여 세부 디자인 및 구현을 지원하도록 정의해야 하는 XDE 프로젝트를 구동시킬 수 있습니다. 특히 XDE 에서 RUP **구현 서브시스템**을 XDE 프로젝트로 나타낼 수 있습니다. 이 방법은 이 가이드라인에서 XDE 프로젝트를 사용하여 **구현 서브시스템**을 나타내는 경우 사용되는 접근 방식입니다.

이 가이드라인의 예제에 대한 **구현 서브시스템**은 다음과 같습니다.

- "전체 디자인 모델"의 각 디자인 서브시스템에 대한 구현 서브시스템 (예: 사용자 계정 관리자 및 경매 관리자)
- 경매 관리 프리젠테이션 요소에 대한 구현 서브시스템
- 사용자 계정 관리 프리젠테이션 요소에 대한 구현 서브시스템

[20] **구현 서브시스템**을 식별하기 위한 가이드라인은 이 문서의 범위에 포함되지 않습니다. 자세한 정보는 RUP 를 참조하십시오.

연관 XDE 프로젝트 및 모델은 그림 18에 표시됩니다.

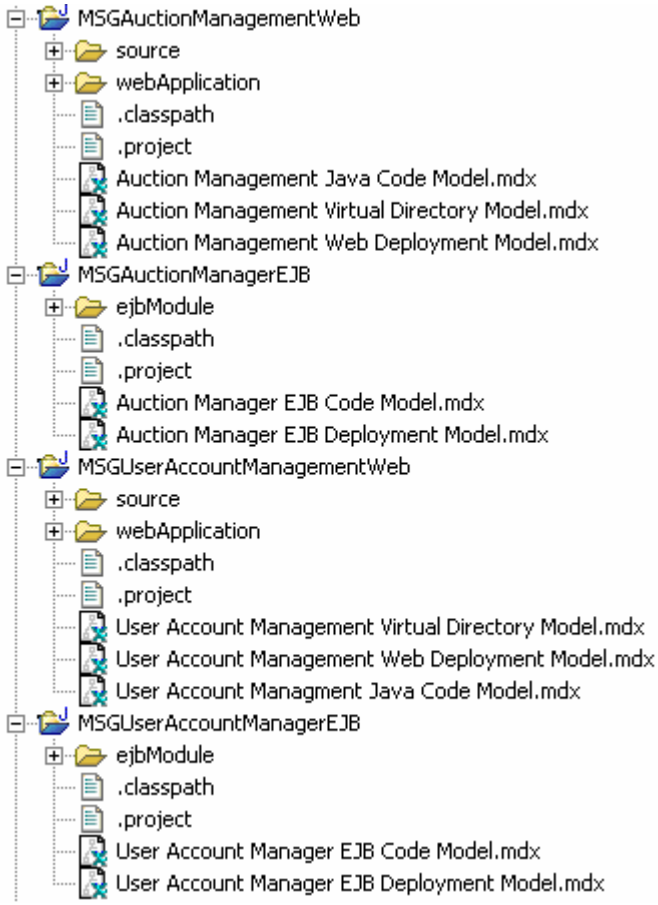

그림 18: XDE 프로젝트로서의 구현 서브시스템

여러 웹 및 EJB 프로젝트를 정의하는 경우, 프로젝트와 포함된 모델 파일에는 반드시 고유한 이름을 지정해야 합니다. 고유한 이름을 지정해야 모델에서 다이어그램을 쉽게 해석하고 모델 간 참조를 해석할 수 있습니다. 그림 18에 표시된 예제에서는 **구현 서브시스템**의 이름이 프로젝트 이름과 포함된 각 모델의 이름에 모두 사용되었습니다.

또는 소규모 프로젝트의 경우, XDE 에서 RUP **구현 서브시스템**을 XDE 모델의 패키지로 나타낼 수 있습니다. 그림 19 는 XDE 패키지("auctionmanager" 및 "useraccountmanager" 패키지)를 사용하여 각 구현 서브시스템을 나타내는 경우의 예제입니다.

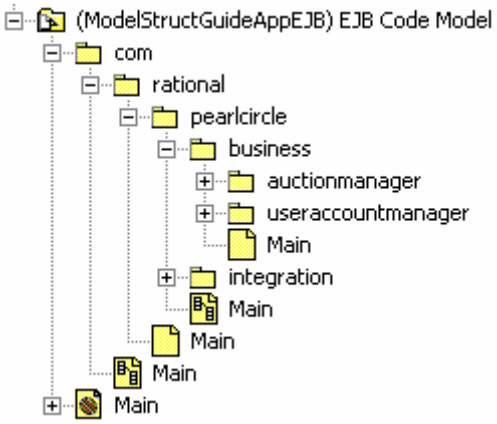

그림 19: 패키지로서의 구현 서브시스템

9.2 XDE 라운드트립 모델

이 섹션은 다음 XDE 라운드트립 모델에 대한 예제를 제공합니다.

- (EJB 프로젝트) EJB 코드 모델(EJB 프로젝트: EJB 코드 모델 섹션 참조)
- (웹 프로젝트) Java 코드 모델(웹 프로젝트: Java 코드 모델 섹션 참조)
- (웹 프로젝트) 가상 디렉토리 모델(웹 프로젝트: 가상 디렉토리 모델 섹션 참조)

일반적으로, XDE 라운드트립 모델을 구조화하는 경우 구현 제한조건을 고려하여 가능한 "전체 디자인 모델"(디자인 모델 섹션 참조)에서 설명한 논리 구조와 유사하게 구조를 조정하는 것이 좋습니다. **디자인 모델**과 **구현 모델**의 구조를 가능한 유사하게 조정함으로써 두 모델 간의 추적성이 명확히 내재되어 유지보수가 용이합니다. 이러한 조정 작업은 **디자인 모델**과 **구현 모델** 간의 맵핑을 아키텍처 유지보수 및 관리 작업의 일부로 유지보수 및 관리해야 한다는 점에서 중요합니다.

XDE 라운드트립 모델에 포함되는 **디자인 클래스**는 다음 기능을 통해 작성할 수 있습니다.

- 새 요소를 작성하거나 **분석 클래스**와 같은 기술 종속 요소에서 요소를 변환함으로써 EJB, Servlet 등과 같은 구현 기술 종속 요소를 작성하기 위한 XDE 자동화
- 기존 구현의 XDE 리버스 엔지니어링
- 수동 작성

9.2.1 EJB 프로젝트: EJB 코드 모델

EJB 코드 모델에는 EJB 를 구현하는 데 필요한 Java 자원(예: 구현 Bean 클래스, 홈 인터페이스 클래스, 원격 인터페이스 클래스 등)이 포함됩니다.

EJB 코드 모델의 소스 루트 특성은 소스 코드가 배치될 디렉토리로 설정해야 합니다. 예를 들어, EJB 코드 모델의 소스 루트 특성은 EJB 프로젝트의 "ejbModule" 서브디렉토리로 설정할 수 있습니다.[21]

[21] XDE 프로젝트 작성 마법사를 사용하여 프로젝트를 작성하는 경우, 프로젝트 서브디렉토리가 자동으로 작성되며 코드 모델의 소스 루트 특성이 자동으로 설정됩니다.

EJB 코드 모델의 예제가 그림 20에 표시됩니다.

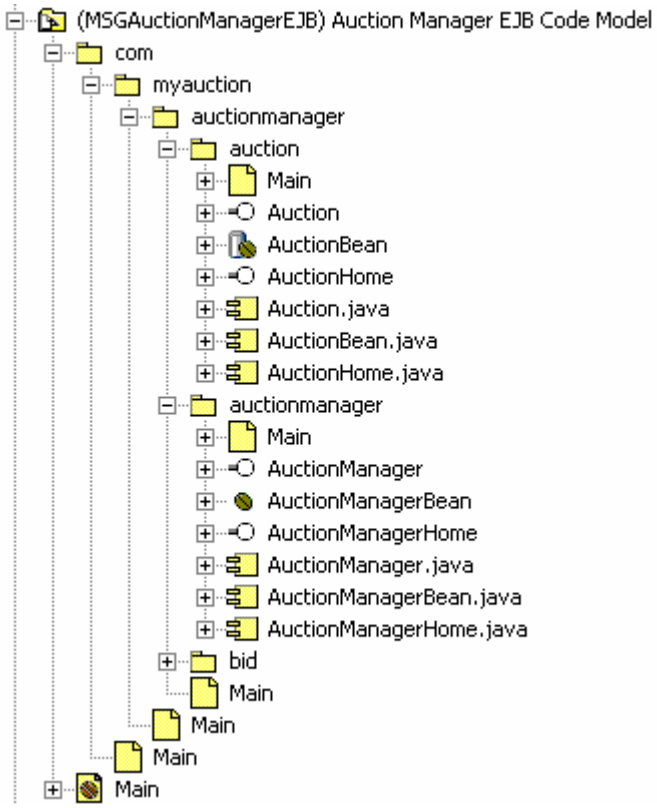

그림 20: EJB 코드 모델 예제

그림 20에 표시된 EJB 코드 모델에는 경매 관리자 **디자인 서브시스템**을 구현하는 Java 자원이 포함됩니다. 이 모델은 "전체 디자인 모델"에서 "비즈니스" 계층 패키지의 "경매 관리자" **디자인 서브시스템** 패키지와 유사한 구현입니다(디자인 계층 섹션 참조).

그림 20과 같이, EJB 코드 모델 구조는 도메인 이름을 초기 Java 패키지 이름으로 사용하는 규칙을 따릅니다. 샘플 응용프로그램의 도메인 이름은 "www.myauction.com"입니다. 따라서 구현 요소를 포함하는 패키지는 "com" 패키지의 "myauction" 패키지 내부에 배치됩니다. 결과적으로, "myauction" 패키지의 모든 Java 요소는 접두부가 "com.myauction"인 완전한 이름을 갖게 됩니다. 예를 들어, "auction" 패키지의 완전한 이름은 "com.myauction.business.auctionmanager.auction"입니다. 도메인 이름을 초기 Java 패키지 이름으로 사용함으로써 써드파티 Java 클래스 라이브러리가 통합되더라도 Java 클래스 이름의 고유성을 유지할 수 있습니다.

"myauction" 패키지에서, 해당 구조는 "전체 디자인 모델"(디자인 모델 섹션 참조)의 구조를 반영합니다. 경매 관리자 **디자인 서브시스템**("business" 패키지)을 포함하는 디자인 계층의 패키지와 경매 관리자 **디자인 서브시스템**("auctionmanager" 패키지)을 나타내는 패키지가 있습니다. EJB 코드 모델에는 또한 관련 모델 요소를 수집하기 위한 추가 패키지(예: "auctionmanager", "auction" 및 "bid" 패키지)가 정의되었습니다.
참고: Java 프로그래밍 언어에서는 패키지 이름에 공백을 허용하지 않으므로 Java 패키지 이름은 연관된 "전체 디자인 모델" 패키지의 이름과 일치하지 않을 수 있습니다.

그림 20과 같이, EJB 코드 모델에는 소스 코드 파일의 비주얼 표시(.java 요소)뿐만 아니라 해당 구현 요소를 지정하는 클래스도 포함됩니다. 이 클래스는 구현(XDE 의 경우 라운드트립 엔지니어링)할 수 있는 상태로 발전 및 성숙된 RUP **디자인 클래스**를 나타냅니다. 참고: XDE 는 EJB 를 지정하는 데 사용되는 모든 클래스를 자동으로 작성하는 패턴을 제공합니다.

9.2.2 웹 프로젝트: Java 코드 모델

웹 프로젝트 Java 코드 모델에는 Java 웹 자원(예: JavaBean, Servlet, 헬퍼 클래스 등)이 포함됩니다.

웹 프로젝트의 Java 코드 모델의 소스 루트 특성은 소스 코드가 배치될 디렉토리로 설정해야 합니다. 예를 들어, 웹 프로젝트의 Java 코드 모델의 소스 루트 특성은 웹 프로젝트의 "Java 소스" 서브디렉토리로 설정할 수 있습니다.[22]

웹 프로젝트 Java 코드 모델"의 예제가 그림 21에 표시됩니다.

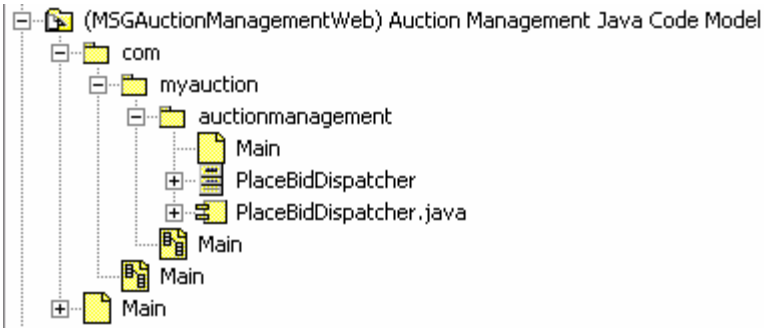

그림 21: 웹 프로젝트 Java 코드 모델 예제

그림 21에 표시된 Java 코드 모델에는 경매 관리 프리젠테이션 디자인 요소를 구현하는 Java 자원이 포함됩니다. 이 모델은 "전체 디자인 모델"에서 "프리젠테이션" 계층 패키지의 "경매 관리" 패키지와 유사한 Java 구현입니다(디자인 계층 섹션 참조).

그림 21와 같이, 웹 프로젝트 Java 코드 모델 구조는 도메인 이름을 초기 Java 패키지 이름으로 사용하는 규칙을 따릅니다. 샘플 응용프로그램의 도메인 이름은 "www.myauction.com"입니다. 따라서 구현 요소를 포함하는 패키지는 "com" 패키지의 "myauction" 패키지 내부에 배치됩니다. 결과적으로, "myauction" 패키지의 모든 Java 요소는 접두부가 "com.myauction"인 완전한 이름을 갖게 됩니다. 예를 들어, "auctionmanagement" 패키지의 완전한 이름은 "com.myauction.presentation.auctionmanagement"입니다. 도메인 이름을 초기 Java 패키지 이름으로 사용함으로써 써드파티 Java 클래스 라이브러리가 통합되더라도 Java 클래스 이름의 고유성을 유지할 수 있습니다.

"myauction" 패키지에서, 해당 구조는 "전체 디자인 모델"(디자인 모델 섹션 참조)의 구조를 반영합니다. 경매 관리 프리젠테이션 요소를 포함하는 디자인 계층의 패키지("presentation" 패키지)와 경매 관리 프리젠테이션 요소 ("auctionmanagement" 패키지)를 나타내는 패키지가 있습니다. 참고: Java 프로그래밍 언어에서는 패키지 이름에 공백을 허용하지 않으므로 Java 패키지 이름은 연관된 "전체 디자인 모델" 패키지의 이름과 일치하지 않을 수 있습니다.

그림 21과 같이, 웹 프로젝트 Java 코드 모델에는 소스 코드 파일의 비주얼 표시(.java 요소)뿐만 아니라 해당 구현 요소를 지정하는 클래스도 포함됩니다. 이 클래스는 구현(XDE 의 경우 라운드트립 엔지니어링)할 수 있는 상태로 발전 및 성숙된 RUP **디자인 클래스**를 나타냅니다.

9.2.3 웹 프로젝트: 가상 디렉토리 모델

가상 디렉토리 모델에는 비 Java 소스 코드 웹 자원이 포함됩니다(예: JSP 및 HTML 페이지). XDE 웹 프로젝트당 여러 가상 디렉토리 모델이 존재할 수 있습니다. 가상 디렉토리 모델이 여러 개인 경우 가상 디렉토리가 여러 개인 J2EE 응용프로그램을 개발할 수 있습니다. 이러한 응용프로그램의 경우 결과 웹 사이트는 실제로 공통 루트가 없는 디렉토리로 분할됩니다. 예를 들어, 온라인 소매 상점에서는 카탈로그

[22] XDE 프로젝트 작성 마법사를 사용하여 프로젝트를 작성하는 경우, 프로젝트 서브디렉토리가 자동으로 작성되며 코드 모델의 소스 루트 특성이 자동으로 설정됩니다.

쇼핑에 www.mystore.com 을 사용하며 주문 상태를 모니터링하기 위해서는 order.mystore.com 을 사용합니다.

가상 디렉토리 모델의 소스 루트 특성은 웹 컨테이너에 배치될 웹 자원을 저장할 디렉토리로 설정되어야 합니다. 예를 들어, 가상 디렉토리 모델의 소스 루트 특성은 웹 프로젝트의 "Web Content" 서브디렉토리로 설정할 수 있습니다.[23]

가상 디렉토리 모델의 예제가 그림 22에 표시됩니다.

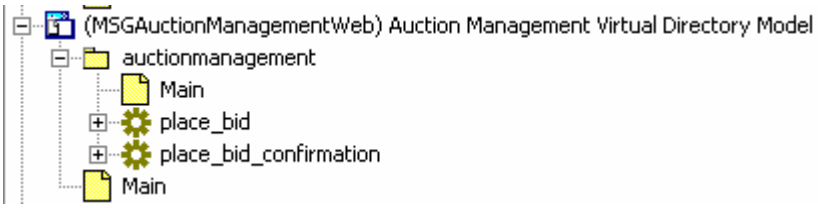

그림 22: 가상 디렉토리 모델 구조

그림 22에 표시된 가상 디렉토리 모델에는 경매 관리 프리젠테이션 디자인 요소를 구현하는 Java 자원이 포함됩니다. 이 모델은 "전체 디자인 모델"에서 "프리젠테이션" 계층 패키지의 "경매 관리" 패키지와 유사한 비 Java 소스 코드 구현입니다(디자인 계층 섹션 참조).

이 예제에서, 가상 디렉토리 모델의 구조는 "전체 디자인 모델"의 구조를 반영합니다(디자인 계층 섹션 참조). "전체 디자인 모델"의 각 패키지마다 해당 컨텐츠가 웹 컨테이너에 배치될 비 Java 요소(예: JSP 및 HTML 페이지)로 구현되는 패키지가 있습니다. 웹 서버에서 액세스하는 디렉토리의 이름 지정 관련 제한조건으로 인해, 가상 디렉토리의 디렉토리 이름이 항상 "전체 디자인 모델"의 디렉토리 이름과 일치하는 것은 아닙니다.

그림 22와 같이 가상 디렉토리 모델에는 구현 요소를 지정하는 클래스가 시각적으로 표시됩니다. 이 클래스는 구현(XDE 의 경우 라운드트립 엔지니어링)할 수 있는 상태로 발전 및 성숙된 RUP **디자인 클래스**를 나타냅니다. 그러나 EJB 코드 모델 및 웹 프로젝트 Java 코드 모델과 달리 XDE 는 가상 디렉토리 모델에 연관된 소스 코드 파일을 시각적으로 표시하지 않습니다. 연관된 소스 코드 파일의 이름은 클래스 특성으로 유지보수됩니다.

10. 배치 모델

RUP **배치 모델**은 여러 XDE 모델("EAR 배치 모델" 및 한 세트의 개별 XDE 배치 모델)로 표시됩니다. 배치될 요소를 포함하는 모든 XDE 프로젝트마다 하나의 모델이 표시됩니다. 여러 배치 모델을 사용하는 경우 XDE 의 배치 자동화를 활용할 수 있습니다. XDE 는 J2EE 아카이브 파일 및 배치 설명자의 작성 및 정제와, 이러한 모델 요소와 소스가 아카이브에 패키지화되는 모델 요소와의 동기화를 자동화합니다.

이 섹션은 다음 배치 모델의 예제를 제공합니다.

- (응용프로그램 프로젝트) EAR 배치 모델(EAR 배치 모델 섹션 참조)
- (EJB 프로젝트) EJB 배치 모델(EJB 배치 모델 섹션 참조)
- (웹 프로젝트) 웹 배치 모델(웹 배치 모델 섹션 참조)

다음은 XDE 의 배치와 관련하여 유의해야 할 몇 가지 일반 항목입니다.

- 배치 설명자는 XDE 배치 모델에 파일(UML 아티팩트)로 명시적으로 표시되지 않습니다. 대신 XDE 배치 모델의 컨텐츠로 표시됩니다. XDE 는 배치 모델에 포함된 요소와 이 요소에 지정된 특성 값을 사용하여 모델의 배치 설명자 파일에 기록된 컨텐츠 정보를 판별합니다.

[23] XDE 프로젝트 작성 마법사를 사용하여 프로젝트를 작성하는 경우, 프로젝트 서브디렉토리가 자동으로 작성되며 가상 디렉토리 모델의 소스 루트 특성이 자동으로 설정됩니다.

- XDE 는 단일 EJB-JAR 또는 WAR 만 배치하는 경우에도 모든 배치에 EAR 배치 모델을 사용합니다. 이는 모든 배치에 EAR 을 사용해야 하는 대부분의 응용프로그램 서버의 제한사항을 반영한 것입니다. 따라서 EJB-JAR 또는 WAR 배치만 모델링하는 경우에도 XDE 에서 지원하는 응용프로그램 서버로의 배치에도 EAR 배치 모델을 사용해야 합니다. 그러나 XDE 는 파일 시스템으로 모든 아카이브를 *내보낼* 수 있습니다. 이 기능은 XDE 에서 지원하지 않는 응용프로그램 서버에 배치하는 데 사용할 수 있습니다. 아카이브를 내보낸 후 서버 특정 도구를 호출하여 배치를 완료할 수 있습니다.
- 배치 환경(테스트, 프로덕션 등)마다 다른 J2EE 아카이브 파일을 정의할 수 있습니다. 이 아카이브는 동일한 XDE 배치 모델 또는 별도 XDE 배치 모델에 정의할 수 있습니다. XDE 자동화는 두 가지 경우를 모두 지원하지만, 프로젝트의 기본 배치 모델과 배치 모델당 기본 아카이브를 정의합니다. 그러나 이러한 정의는 XDE 배치 모델에 모델링될 때 조정할 수 있습니다. 어떠한 경우라도 EJB 배치 모델을 EJB 프로젝트에 정의해야 하며 웹 배치 모델은 웹 프로젝트에 포함되어야 합니다().[24]

10.1 EAR 배치 모델

"EAR 배치 모델"에는 J2EE 응용프로그램 아카이브 파일(.EAR 파일), EAR 배치 설명자 정보 및 EAR 을 전개할 노드가 시각적으로 표시됩니다. "EAR 배치 모델"에는 또한 EAR 에 포함되는 J2EE 모듈을 보여주는 다른 배치 모델의 다이어그램이 포함됩니다.

"EAR 배치 모델"은 또한 응용프로그램의 전체 배치 구성과 아키텍처의 배치 보기를 설명하는 데 사용할 수 있습니다. "EAR 배치 모델"에는 모든 배치 노드와 해당 연결은 물론 특정 노드에 배치될 아카이브와 해당 노드를 보여주는 다이어그램이 포함될 수 있습니다. 이러한 다이어그램은 모든 개별 배치 모델의 요소(노드 및 아카이브)를 참조합니다.

"EAR 배치 모델"의 예제가 그림 23에 표시됩니다.

그림 23: EAR 배치 모델

"배치 보기" 다이어그램은 아키텍처의 배치 보기를 나타냅니다. 이 다이어그램에는 배치 노드와 노드 간 연결 및 해당 노드에 배치될 J2EE 아카이브가 포함됩니다. 경우에 따라 이 다이어그램에는 J2EE 응용프로그램 아카이브 및 이 아카이브가 전개될 응용프로그램 서버 노드만 포함될 수 있습니다. 그러나 독립형 J2EE 모듈 아카이브가 특정 노드로 배치되는 경우, 이 다이어그램에는 노드에 배치될 아카이브와 해당 노드가 표시되어야 합니다. 아키텍처 보기에 대한 정보는 RUP 를 참조하십시오.

[24] EJB 배치 모델이 EJB 프로젝트에 포함되어야 하지만 해당 EJB 코드 모델과 *동일한*프로젝트가 아니어도 됩니다. 실제로는 여러 코드 모델의 EJB 를 하나의 배치 모델로 "혼합 및 조정"할 수 있습니다. 웹 모델에도 동일한 내용이 적용됩니다.

10.2 EJB 배치 모델

EJB 배치 모델에는 EJB 컴포넌트, EJB-JAR 아티팩트 및 EJB 배치 설명자 정보가 포함됩니다. EJB 배치 모델에는 또한 EJB-JAR 에 포함되는 구현 요소를 보여주는 EJB 코드 모델의 다이어그램이 포함됩니다. 즉, EJB 배치 모델에서는 EJB 컴포넌트를 사용하여 구현 요소를 나타내며 EJB 컴포넌트는 EJB-JAR 로 맵핑됩니다.

EJB 배치 모델의 예제가 그림 24에 표시됩니다.

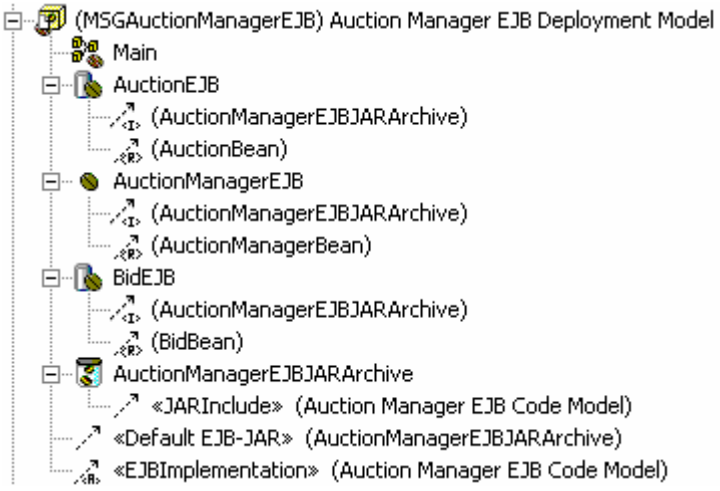

그림 24: EJB 배치 모델 예제

10.3 웹 배치 모델

웹 배치 모델에는 웹 컴포넌트, WAR 아카이브 파일 및 웹 배치 설명자 정보가 포함됩니다. 웹 배치 모델에는 또한 WAR 에 포함되는 구현 요소를 보여주는 웹 프로젝트 라운드트립 모델의 다이어그램이 포함됩니다. 즉, 웹 배치 모델에서는 웹 컴포넌트를 사용하여 구현 요소를 나타내며 웹 컴포넌트는 WAR 로 맵핑됩니다.

웹 배치 모델의 예제가 그림 25에 표시됩니다.

그림 25: 웹 배치 모델

Rational
the software development company

본사 안내:
Rational Software
18880 Homestead Road
Cupertino, CA 95014
전화번호: (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
전화번호: (781) 676-2400

수신자 부담 전화번호: (800) 728-1212
전자 우편: info@rational.com
웹: www.rational.com
전세계 지사 안내: www.rational.com/worldwide