UML 을 사용하여 웹 응용프로램 아키텍처

Jim Conallen

Rational Software 백서

TP 157, 6/99

이 자료 버전은 Communications of the ACM 의 1999 년 10 월(볼륨 42, 번호 10)호에 기재되어 있습니다.

Rational®

the software development company

목차

요약

웹 응용프로그램은 점점 더 복잡해지고 업무 관련 중요성 또한 증가하고 있습니다. 이 복잡도를 관리하려면 웹 응용프로그램을 모델링해야 합니다. UML(Unified Modeling Language)은 소프트웨어 중심 시스템을 모델링하기 위한 표준 언어입니다. UML 로 웹 응용프로그램을 모델링하는 경우 해당 컴포넌트 중 일부가 표준 UML 모델링 요소와 잘 맞지 않음을 알 수 있습니다. 전체 시스템에 단일 모델링 표기법을 적용하려면(웹 컴포넌트 및 일반적인 중간 층 컴포넌트) UML 을 확장해야 합니다. 이 문서에서는 정규 확장 메커니즘을 사용하여 확장된 UML 을 나타냅니다. 확장은 웹 특정 컴포넌트를 나머지 시스템 모델에 통합할 수 있고 웹 응용프로그램의 디자이너, 구현자 및 설계자에게 적합한 올바른 추상 및 세부사항 레벨을 나타내도록 디자인됩니다.

개요

과거 몇 년 동안 IT 용어에 웹 응용프로그램이라는 새 용어가 추가되었습니다. 비즈니스 소프트웨어 시스템 사용자라면 누구나 웹 응용프로그램 빌드 계획을 갖고 있는 것처럼 보이지만 비즈니스와 관련이 없는 많은 소프트웨어 노력에도 관심을 갖고 있습니다. 이 아키텍처의 얼리어댑터(early adopter)에게 있어 웹 응용프로그램이라는 용어는 시스템과 마찬가지로 성공적인 소규모 웹 사이트의 추가 기능에서 견고한 n-층 응용프로그램으로 발전된 개념입니다. 웹 응용프로그램이 전세계에 분산되어 있는 수만 명의 동시 사용자를 지원하는 경우는 드문 일이 아닙니다. 따라서 웹 응용프로그램을 어떻게 설계하는가가 중요합니다.

테스트와 관련해서는 웹 응용프로그램이라는 용어가 사용자에 따라 약간 다른 의미를 갖습니다. 즉, 사용자에 따라 웹 응용프로그램을 Java 사용 또는 웹 서버 사용 관점에서 접근합니다. 그러나 실제로는 그 중간 관점에서 이해하는 것이 일반적입니다. 이 문서에서는 그 취지에 맞게 웹 응용프로그램을 사용자 입력(탐색 및 데이터 입력)이 비즈니스 상태에 영향을 주는 웹 시스템(웹 서버, 네트워크, HTTP, 브라우저)으로 정의합니다. 이 정의는 웹 응용프로그램을 비즈니스 상태를 갖는 소프트웨어 시스템으로 설정하고 해당 "프론트 엔드"는 대부분 웹 시스템을 통해 전달되는 것으로 설정합니다.

웹 응용프로그램의 일반 아키텍처는 클라이언트 서버 시스템의 일반 아키텍처와 같지만 몇 가지 뚜렷한 차이가 있습니다. 웹 응용프로그램의 가장 큰 장점은 그 배치에 있습니다. 웹 응용프로그램의 배치는 일반적으로 네트워크에 서버측 컴포넌트를 설정하는 것으로서 클라이언트 파트에서는 특별한 소프트웨어 또는 구성이 필요하지 않습니다. 또 다른 중요 차이점은 클라이언트 및 서버 통신의 특성입니다. 웹 응용프로그램의 기본 통신 프로토콜은 HTTP 입니다. HTTP 는 최대 통신 처리량이 아닌 견고성 및 결함 허용을 위해 디자인된 비연결형 프로토콜입니다. 웹 응용프로그램의 클라이언트와 서버 간 통신은 일반적으로 서버측 오브젝트와 클라이언트측 오브젝트 간의 직접 통신이 아닌 웹 페이지 탐색에 초점이 맞추어집니다. 특정 추상 레벨의 경우, 웹 응용프로그램의 모든 메시지 전달은 웹 페이지 엔티티의 요청 및 수신으로 설명할 수 있습니다. 일반적으로 웹 응용프로그램의 아키텍처는 동적 웹 사이트의 아키텍처와 크게 다르지 않습니다.

웹 응용프로그램과 웹 사이트의 차이점은 동적인 측면을 고려하더라도 해당 사용법과 관련이 있습니다. 웹 응용프로그램은 비즈니스 로직을 구현하며 이 로직을 사용함으로써 비즈니스 상태(시스템에서 캡처)가 변경됩니다. 이는 모델링 노력의 핵심을 정의한다는 점에서 중요합니다. 웹 응용프로그램은 비즈니스 로직을 실행하므로 가장 중요한 시스템 모델은 프리젠테이션 세부사항이 아닌 비즈니스 로직 및 비즈니스 상태에 초점이 맞추어집니다. 올바른 시스템 기능을 위해서는 프리젠테이션도 중요하긴 하지만 비즈니스 관심사항과 프리젠테이션 관심사항을 명확히 구분해야 합니다. 프리젠테이션 문제가 중요하거나 복잡한 문제인 경우 해당 문제 역시 모델링되어야 하지만 비즈니스 로직 모델의 핵심 파트가 아닐 수도 있습니다. 또한 프리젠테이션 관련 자원은 보다 예술적이고 비즈니스 규칙 구현과는 관련이 적은 것이 일반적입니다.

웹 시스템 개발과 연관된 방법론/표기법 중 하나는 RMM(Relationship Management Methodology)입니다. RMM 은 인트라넷 및 인터넷 웹 시스템의 디자인, 생성 및 유지보수를 위한 방법론으로서 주요 목적은 데이터베이스 기반 동적 웹 사이트의 유지보수 비용을 절감하는 것입니다. 이 방법론은 원활한 디자인 논의를 가능하게 하는 시스템의 비주얼 표현을 지원합니다. 또한 이 방법론은 웹 페이지의 비주얼 요소 분해와, 데이터베이스 엔티티와의 연관 관계를 포함하는 반복 프로세스입니다. RMM 은 동적 웹 사이트의 작성 및 유지보수를 위한 "종합적" 접근 방식입니다.

RMM 은 웹 응용프로그램 빌드를 완전하게 지원하지 않습니다. 비즈니스 로직 중심의 웹 응용프로그램에는 RMM 표기법에서 올바르게 다루지 않는 비즈니스 로직을 구현하기 위한 여러 가지 기술 메커니즘이 포함됩니다. 클라이언트측 스크립팅, 애플릿 및 ActiveX 제어와 같은 이러한 기술은 시스템의 비즈니스 규칙 실행에 중요한 컨트리뷰션을 합니다. 또한 웹 응용프로그램은 분산형 오브젝트 시스템의 전달 메커니즘으로 사용할 수 있습니다. 애플릿 및 ActiveX 제어에는 웹 서버와 관계 없이 RMI 또는 DCOM 을 통해 서버측 컴포넌트와 비동기식으로 상호작용하는 컴포넌트가 포함될 수 있습니다. 정교한 응용프로그램은 또한 클라이언트의 여러 브라우저 인스턴스 및 프레임을 사용합니다. 이러한 인스턴스 및 프레임은 자체 통신 메커니즘을 설정하고 유지보수합니다.

이러한 메커니즘은 모두 시스템의 비즈니스 로직에 컨트리뷰션하므로 그에 따라 모델링되어야 합니다. 또한 이러한 메커니즘은 비즈니스 로직 파트만 나타내므로 나머지 시스템 모델과 통합되어야 합니다. 많은 경우에 있어 서버측 특정 층의 웹 서버 뒤에서 대량 비즈니스 로직이 실행됩니다. 모델링 언어 및 표기법에 대한 선택은 일반적으로 응용프로그램에서 이 측의 요구에 따라 결정됩니다. OMG 에서 UML 을 공식 오브젝트 모델링 언어로 승인함으로써 보다 많은 시스템을 UML 표기법으로 표현할 수 있습니다. 대부분의 경우 UML 은 소프트웨어 중심 시스템을 모델링하기 위한 언어로 선택됩니다. 이러한 경우 웹 응용프로그램 모델링 관련 기본 문제는 "나머지 응용프로그램과 함께 웹 특정 컴포넌트에서 실행되는 비즈니스 로직을 표현하는 방법"입니다. 해답은 UML 을 사용하여 해당 웹 특정 요소 및 기술로 시스템의 비즈니스 로직 실행을 표현할 수 있는 기능과 관련이 있습니다.

이 문서는 웹 응용프로그램 관련 문제 및 가능한 솔루션에 대한 기본적인 설명을 위해 작성되었으며 웹 응용프로그램에서 구조적으로 중요한 특정 컴포넌트와, UML 로 해당 컴포넌트를 모델링하는 방법을 중점 설명합니다. 이 문서는 또한 사용자가 UML, 객체 지향 프린시펄 및 웹 응용프로그램 개발에 대해 잘 알고 있는 것으로 가정합니다. 이 문서에서 설명하는 작업은 다음과 같은 몇 가지 올바른 가정을 기반으로 합니다.

- 웹 응용프로그램은 점점 더 복잡해지고 업무상 중요한 역할을 수행하고 있는 소프트웨어 중심 시스템입니다.
- 소프트웨어 시스템의 복잡도를 관리하기 위한 한 가지 방법은 시스템을 추상화하고 모델링하는 것입니다.
- 소프트웨어 시스템은 일반적으로 각각 다른 시점, 추상 레벨 및 세부사항을 나타내는 여러 모델로 구성됩니다.
- 올바른 추상 및 세부사항 레벨은 개발 프로세스의 아티팩트 및 활동에 따라 다릅니다.
- 소프트웨어 중심 시스템의 표준 모델링 언어는 UML(Unified Modeling Language)입니다.

이 문서에서 언급한 개념 및 아이디어는 현재 작성 중인 "UML 로 웹 응용프로그램 빌드"에서 자세히 다룹니다. 이 책은 올해 후반부에 Addison Wesley Longman 의 Object Technology Series 에 출간될 예정입니다.

모델링

모델은 특정 세부사항을 단순화함으로써 시스템 이해를 용이하게 합니다. 모델링 대상의 선택은 문제점에 대한 이해 및 솔루션 형태에 큰 영향을 미칩니다. 웹 응용프로그램은 다른 소프트웨어 중심 시스템과 같이 일반적으로 한 세트의 모델(유스 케이스 모델, 구현 모델, 배치 모델, 보안 모델 등)로 표현됩니다. 웹 시스템에서만 사용하는 추가 모델로서, 시스템 전체의 웹 페이지 및 탐색 루트 추상을 나타내는 사이트 맵이 있습니다.

오늘날 사용하는 대부분의 모델링 기법은 웹 응용프로그램의 다양한 모델을 개발하는 데 적합하므로 더 이상 설명하지 않습니다. 그러나 중요한 모델 중 하나인 ADM(Analysis/Design Model)은 웹 페이지, 웹 페이지와 연관된 실행 가능 코드와 기타 요소를 모델에 함께 포함할 때 몇 가지 어려움이 있습니다.

모델링 방법을 결정하는 경우 올바른 추상 및 세부사항 레벨을 결정해야 모델 사용자에게 유익한 방법을 제공할 수 있습니다. 일반적으로 시스템 아티팩트를 모델링하는 것이 가장 좋습니다. 시스템 아티팩트는 최종 제품을 생성하기 위해 구성 및 조작되는 실제 엔티티입니다. 웹 서버의 내부 또는 웹 브라우저의 세부사항 모델링은 웹 응용프로그램의 디자이너와 설계자에게 도움이 되지 않습니다. 페이지, 페이지 간 링크, 페이지 작성에 사용된 모든 동적 컨텐츠 및 클라이언트 페이지의 동적 컨텐츠에 대한 모델링이 매우 중요합니다. 이러한 아티팩트는

디자이너가 디자인하고 구현자가 구현하는 대상입니다. 모델링 대상은 클라이언트와 서버의 페이지, 하이퍼링크 및 동적 컨텐츠입니다.

다음은 이러한 아티팩트를 모델링 요소에 맵핑하는 단계입니다. 예를 들어, 하이퍼링크는 모델의 연관 요소에 맵핑됩니다. 하이퍼링크는 페이지 간 탐색 경로를 나타냅니다. 따라서 페이지는 모델의 논리 보기 내 클래스에 맵핑됩니다. 웹 페이지가 모델의 클래스인 경우 페이지의 스크립트는 클래스 오퍼레이션에 맵핑됩니다. 스크립트의 페이지 범위 변수는 클래스 속성에 맵핑됩니다. 웹 페이지에 서버에서 실행되는 스크립트 세트(페이지의 동적 컨텐츠 준비)와 클라이언트에서만 실행되는 완전히 다른 스크립트 세트(예: JavaScript)를 함께 포함하려는 경우 문제점이 발생합니다. 이러한 시나리오에서는 모델의 웹 페이지 클래스를 고려할 때, 서버에서 활성화되는 오퍼레이션, 속성 및 관계와 사용자가 클라이언트 페이지와 상호작용할 때 활성화되는 오퍼레이션, 속성 및 관계가 명확하지 않습니다. 또한 웹 응용프로그램에서 전달되는 웹 페이지는 실제로 시스템 컴포넌트로서 가장 효과적으로 모델링됩니다. 단순히 웹 페이지를 UML 클래스로 맵핑하는 것만으로는 시스템 이해에 도움이 되지 않습니다.

UML 설계자는 UML 만으로는 특정 도메인 또는 아키텍처의 관련 시맨틱을 캡처하는 데 충분하지 않음을 알게 되었습니다. 또한 이러한 목적을 달성하기 위해 종사자(practitioner)가 UML 의 시맨틱을 확장할 수 있는 정규 확자 메커니즘이 정의되었습니다. 이 메커니즘을 이용하면 *스트레오타입*, *태그 값* 및 모델 요소에 적용할 수 있는 *제한조건*을 정의할 수 있습니다.

*스트레오타입*은 모델링 요소에 대한 새 시맨틱 의미를 정의할 수 있는 부가 기능입니다. *태그 값*은 모델링 요소에 "태그 값을 추가"할 수 있는 모델링 요소와 연관될 수 있는 키 값 쌍입니다. *제한조건*은 올바른 모델 형태를 정의하는 규칙입니다. 제한조건은 자유 양식 텍스트 또는 보다 정형화된 OCL(Object Constraint Language)로 표현할 수 있습니다.

이 문서에서 설명하는 작업에서는 웹 응용프로그램에 확장된 UML 을 사용합니다. 이러한 확장은 이 문서의 범위를 벗어나는 것이지만 대부분의 개념 및 설명은 이 문서에서 논의됩니다.

모델링에 대한 마지막 사항으로서 비즈니스 로직과 프리젠테이션 로직을 명확히 구분해야 합니다. 일반 비즈니스 응용프로그램의 경우 비즈니스 로직만 ADM 에 포함되어야 합니다. 애니메이션 단추, 풍선 도움말 및 기타 UI 개선사항과 같은 프리젠테이션 세부사항은 일반적으로 ADM 에 포함되지 않습니다. 이러한 세부사항은 응용프로그램에 대한 별도 UI 모델이 생성되는 경우 해당 모델에 포함됩니다. ADM 은 기본적으로 비즈니스 문제점 및 솔루션 공간을 표현해야 합니다. 수많은 웹 아티스트가 활동하는 오늘날, 웹 페이지의 룩앤필은 기존 개발자가 아닌 전문가(기술 그래픽 아티스트)에 의해 가장 효과적으로 디자인 및 구현됩니다.

웹 응용프로그램 아키텍처

웹 응용프로그램의 기본 아키텍처에는 브라우저, 네트워크 및 웹 서버가 포함됩니다. 브라우저는 서버에서 "웹 페이지"를 요청합니다. 각 페이지는 컨텐츠 및 형식화 지시사항이 혼합되어 HTML 형식으로 표현됩니다. 일부 페이지에는 브라우저가 해석하는 클라이언트측 스크립트가 포함됩니다. 이 스크립트는 표시 페이지의 추가 동적 동작을 정의하며 종종 브라우저, 페이지 컨텐츠 및 페이지에 포함된 추가 제어(애플릿, ActiveX 제어 및 플러그인)와 상호작용합니다. 사용자는 페이지 컨텐츠를 보고 상호작용합니다. 사용자는 때때로 페이지의 필드 요소에 정보를 입력하고 서버에 제출하여 처리합니다. 사용자는 또한 하이퍼링크를 통해 시스템의 다른 페이지를 탐색함으로써 시스템과 상호작용할 수 있습니다. 두 가지 경우 모두 사용자는 시스템의 "비즈니스 상태"를 변경할 수 있는 입력을 시스템에 제공합니다.

클라이언트 관점에서 볼 때 웹 페이지는 항상 HTML 형식 문서입니다. 그러나 서버 관점에서 볼 때 "웹 페이지"는 여러 가지 다른 방법으로 Manifest 됩니다. 초기 웹 응용프로그램에서는 CGI(Common Gateway Interface)로 동적 웹 페이지를 빌드했습니다. CGI 는 페이지 요청과 함께 전달된 정보에 액세스하기 위해 사용할 스크립트 및 컴파일 모듈용 인터페이스를 정의합니다. CGI 기반 시스템에서는 일반적으로 페이지 요청에 대한 응답으로 스크립트를 실행할 수 있도록 웹 서버에 특수 디렉토리가 구성됩니다. CGI 스크립트가 요청되면, 서버에서 다른 HTML 형식 파일과 같이 파일 컨텐츠만 리턴하지 않고 해당 해석기(일반적으로 PERL 쉘)로 파일을 처리 또는 실행하고 해당 출력을 다시 요청 클라이언트에 전달합니다. 이 처리 작업의 최종 결과는 HTML 형식 스트림이며 이 스트림은 다시 요청 클라이언트로 송신됩니다. 파일을 처리하는 동안에는 시스템에서 비즈니스 로직이 실행되며 이 때 데이터베이스 또는 중간 층 컴포넌트와 같은 서버측 자원과 상호작용할 수 있습니다.

오늘날의 웹 서버는 이 기본 디자인을 기반으로 개선되었습니다. 즉, 보안 기능이 강화되었으며 서버에서의 클라이언트 상태 관리, 트랜잭션 처리 통합, 원격 관리 및 소수 이름 지정을 위한 자원 풀링과 같은 기능이 포함되어 있습니다. 일반적으로 최신 웹 서버는 업무상 중요하고 확장 가능하며 견고한 응용프로그램과 같은 설계자에게 중요한 문제를 처리합니다.

CGI 스크립트 역할과 관련하여 오늘날의 웹 서버는 세 가지 주요 카테고리(스크립트 페이지, 컴파일 페이지 및 이 두 페이지의 혼성)로 분류할 수 있습니다. 첫 번째 카테고리에서는 클라이언트 브라우저가 요청할 수 있는 각 웹 페이지가 웹 서버의 파일 시스템에서 스크립트 파일로 표시됩니다. 이 파일은 일반적으로 HTML 과 다른 스크립팅 언어가 혼합된 형식입니다. 페이지가 요청되면 웹 서버는 이 페이지 처리를 해당 페이지를 인식하는 엔진에 위임합니다. 이 때 HTML 형식 스트림이 요청 클라이언트로 다시 송신되는 최종 결과도 함께 위임됩니다. 이러한 예는 Microsoft 의 Active Server Pages, Java Server Pages 및 Cold Fusion 입니다.

두 번째 카테고리인 컴파일 페이지에서는 웹 서버가 2 진 컴포넌트를 로드하고 실행합니다. 이 컴포넌트는 스크립트 페이지와 같이, 페이지 요청과 함께 제공된 모든 정보(양식 필드 값과 매개변수값)에 액세스할 수 있습니다. 컴파일 코드는 요청 세부사항을 사용하며, 일반적으로 서버측 자원을 평가하여 클라이언트에 리턴되는 HTML 스트림을 생성합니다. 컴파일 페이지는 일반적으로 스크립트 페이지보다 다양한 기능을 나타낼 수 있습니다. 즉, 컴파일 페이지 요청으로 매개변수를 전달함으로써 다양한 기능을 나타낼 수 있습니다. 단일 컴파일 컴포넌트가 실제로 전체 디렉토리에 있는 스크립트 페이지의 모든 기능을 포함할 수 있습니다. 이러한 유형의 아키텍처를 나타내는 기술은 Microsoft 의 ISAPI 및 Netscape 의 NSAPI 입니다.

세 번째 카테고리는 요청 시 컴파일되는 스크립트 페이지를 나타내며 이후 모든 후속 요청에서는 이 컴파일 버전을 사용합니다. 원래 페이지의 컨텐츠가 변경될 때만 해당 페이지에서 다른 컴파일을 수행합니다. 이 카테고리는 스크립트 페이지의 유연성과 컴파일 페이지의 효율성이 조율된 카테고리입니다.

웹 페이지 모델링

웹 페이지(스크립트 페이지 또는 컴파일 페이지)는 UML 컴포넌트에 일 대 일로 맵핑됩니다. 컴포넌트는 교체 가능한 "실제" 시스템 파트입니다. 모델의 구현 보기(컴포넌트 보기)는 시스템 컴포넌트 및 해당 관계를 나타냅니다. 웹 응용프로그램의 경우 이 보기는 시스템의 모든 웹 페이지와 해당 관계(즉, 하이퍼링크)를 나타냅니다. 특정 레벨에서 웹 시스템의 컴포넌트 다이어그램은 사이트 맵과 같습니다.

컴포넌트는 인터페이스의 실제 패키징만을 나타내므로 페이지 내부의 협업 모델링에는 적합하지 않습니다. 이 추상 레벨은 디자이너와 구현자에게 매우 중요하므로 모델에 포함되어야 합니다. 먼저 각 웹 페이지는 모델의 디자인 보기(논리 보기)에서 UML 클래스이며 다른 페이지에 대한 해당 관계(연관)는 하이퍼링크로 나타냅니다. 그러나 해당 웹 페이지가 잠재적으로 서버에 있는 함수 및 협업 세트만 나타낼 수 있는 경우 또한 클라이언트에 완전히 다른 세트만 있는 경우에는 이러한 추상이 적용되지 않습니다. 동적 HTML(클라이언트측 스크립팅)을 출력의 일부로 채택하는 서버 스크립트 웹 페이지는 이러한 페이지의 한 예입니다. 이러한 문제점에 대한 즉각적인 반응으로, 클래스의 각 속성 또는 오퍼레이션이 서버 또는 클라이언트측에서 유효한지 여부를 나타내도록 스테레오타입을 지정할 수 있습니다. 이러한 경우 원래 의도가 단순화였던 모델이 매우 복잡해지게 됩니다.

문제점에 대한 보다 효과적인 접근 방식은 "관심사항 분리" 프린시펄을 고려하는 것입니다. 논리적으로 볼 때 서버에서의 웹 페이지 동작은 클라이언트에서와 완전히 다릅니다. 서버에서 실행되는 동작은 서버측 자원(중간층 컴포넌트, 데이터베이스, 파일 시스템 등)에 액세스할 수 있습니다. 즉, 해당 동작과 자원이 연관될 수 있습니다. 클라이언트의 동일한 해당 페이지 또는 해당 페이지의 스트림 HTML 출력은 완전히 다른 관계 동작 및 세트를 갖게 됩니다. 클라이언트에서는 스크립트 페이지가 DOM(Document Object Model)을 통해 브라우저와 또한 페이지가 지정하는 Java 애플릿, ActiveX 제어 및 플러그인과 연관됩니다. 신중한 디자이너라면 다른 HTML 프레임 또는 브라우저 인스턴스에 나타나는 클라이언트의 기타 "활성" 페이지와의 관계도 추가할 수 있습니다.

관심사항에 관계 없이, 클래스가 하나인 웹 페이지의 서버측 측면과 다른 클래스를 갖는 클라이언트측 측면을 모델링할 수 있습니다. 이 두 측면은 각각에 대한 스트레오타입 및 아이콘을 정의하는 UML 의 확장 메커니즘을 사용하여 구분할 수 있습니다(서버 페이지 및 클라이언트 페이지). UML 의 스테레오타입을 사용하면 모델링 요소에 대한 새 시맨틱을 정의할 수 있습니다. 스테레오타입화된 클래스는 사용자 정의 아이콘을 사용하거나 guillemet 간 스테레오타입 이름을 표시하여 UML 다이어그램에 렌더링할 수 있습니다. 아이콘은 개요

다이어그램에 유용하게 사용할 수 있습니다. 이 다이어그램에서는 클래스 속성 및 오퍼레이션이 노출될 때 간단한 태그를 사용하는 것이 가장 효과적입니다.

웹 페이지의 경우, 스테레오타입은 클래스가 클라이언트 또는 서버에 있는 웹 페이지의 논리 동작에 대한 추상임을 나타냅니다. 두 가지 추상이 방향성 관계를 통해 서로 연관됩니다. 서버 페이지가 클라이언트 페이지를 빌드한다는 측면에서 이 연관은 빌드로 스트레오타입화됩니다(그림 1). 모든 동적 웹 페이지(즉, 컨텐츠가 런타임에서 결정되는 페이지)는 서버 페이지와 함께 생성됩니다. 모든 클라이언트 페이지는 단일 서버 페이지로 빌드되지만 하나의 서버 페이지에서 여러 클라이언트 페이지를 빌드할 수 있습니다.

웹 페이지 간의 일반 관계는 하이퍼링크입니다. 웹 응용프로그램의 하이퍼링크는 시스템의 탐색 경로를 나타냅니다. 이 관계는 모델에서 링크 스테레오타입의 연관으로 표현됩니다. 이 연관은 항상 클라이언트 페이지에서 시작하여 클라이언트 또는 서버 페이지로 연결됩니다.

하이퍼링크는 시스템에서 웹 페이지에 대한 요청으로 구현되며 웹 페이지는 구현 보기에서 컴포넌트로 모델링됩니다. 클라이언트 페이지에 대한 링크 연관은 대부분 클라이언트 페이지를 빌드하는 서버 페이지에 대한 링크 연관과 같습니다. 이는 링크가 클래스 추상이 아닌 실제 페이지에 대한 요청이기 때문입니다. 웹 페이지 컴포넌트는 두 페이지 추상을 모두 인식하므로 페이지 컴포넌트가 인식하는 클래스에 대한 링크는 동일합니다.

태그 값은 링크 요청과 함께 전달되는 매개변수를 정의하는 데 사용됩니다. 링크 연관 태그 값 "매개변수"는 요청을 처리하는 서버 페이지에서 예상하고 사용하는 매개변수 이름(및 선태적 값)의 목록입니다. 그림 2 에서 SearchResults 페이지에는 그 수가 가변적인 GetProduct 서버 페이지에 대한 하이퍼링크(0..*)가 포함됩니다. 여기서 각 링크는 productId 매개변수에 대해 다른 값을 갖습니다. GetProduct 페이지는 productId 매개변수로 지정된 제품의 ProductDetail 페이지를 빌드합니다.

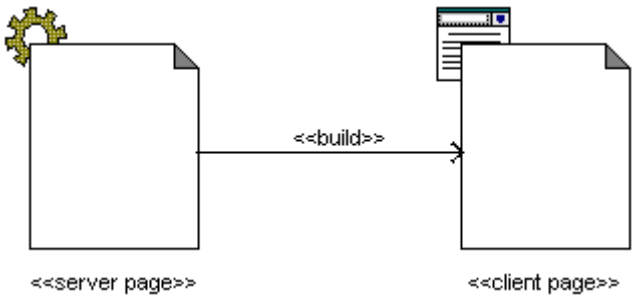

그림 1. 서버 페이지에서 클라이언트 페이지 빌드

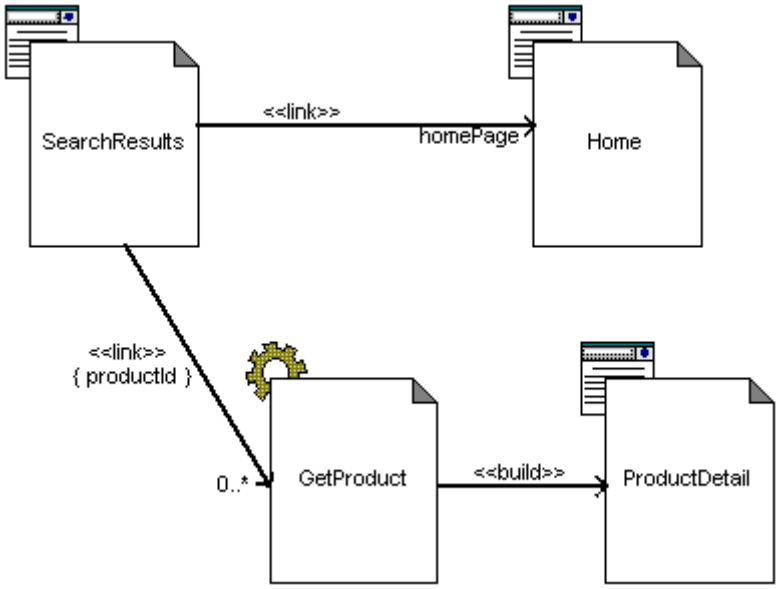

그림 2. 하이퍼링크 매개변수 사용

이 스테레오타입을 사용하면 페이지의 스크립트 및 관계를 보다 쉽게 모델링할 수 있습니다. 서버 페이지 클래스의 오퍼레이션은 페이지의 서버측 스크립트에서 함수가 되며 해당 속성은 페이지 범위의 변수(페이지 기능에서 글로벌 액세스 가능)가 됩니다. 클라이언트 페이지 클래스의 오퍼레이션 및 속성 역시 클라이언트에서 표시 가능한 함수 및 변수가 됩니다. 페이지의 서버 및 클라이언트측 측면을 다른 클래스로 분할하는 데 따른 주요 이점은 시스템의 페이지 및 기타 클래스 간 관계에 있습니다. 클라이언트 페이지는 클라이언트측 자원(DOM, Java 애플릿, ActiveX 제어 및 플러그인)과의 관계로 모델링됩니다(그림 3). 서버 페이지는 그림 4 와 같은 서버측 자원(중간 층 컴포넌트, 데이터베이스 액세스 컴포넌트, 서버 운영 체제 등)과의 관계로 모델링됩니다.

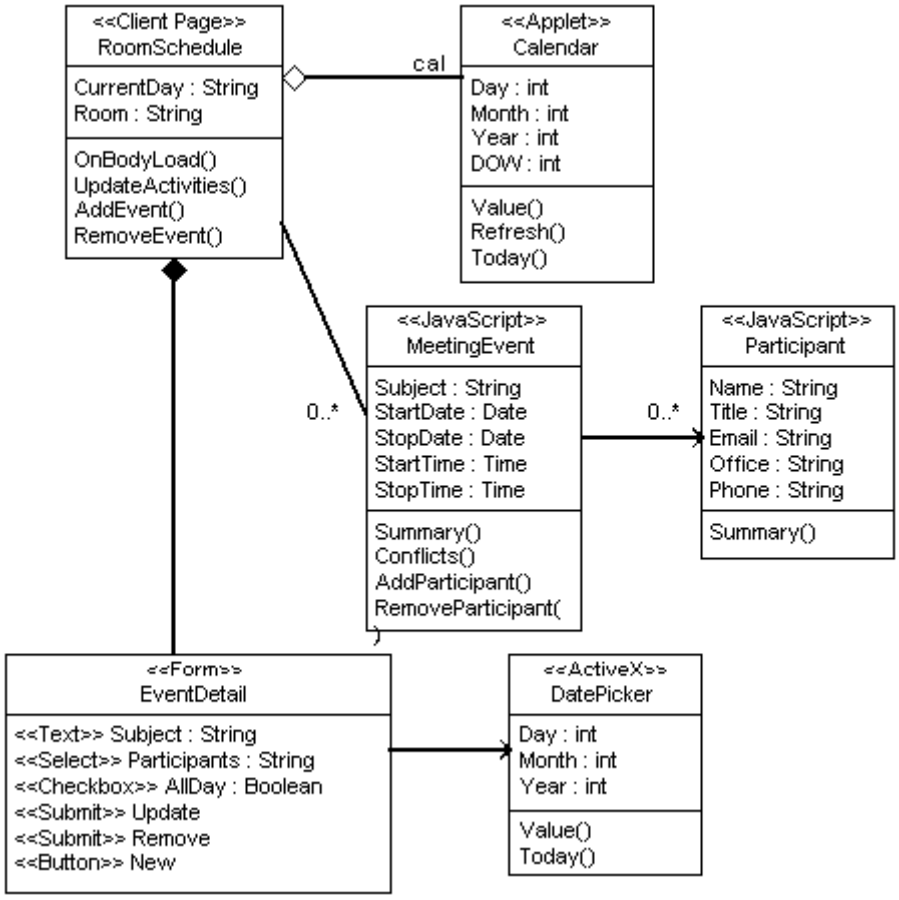
<<Client Page>>
RoomSchedule
CurrentDay : String
Room : String
OnBodyLoad()
UpdateActivities()
AddEvent()
RemoveEvent()
cal
<<Applet>>
Calendar
Day : int
Month : int
Year : int
DOW : int
Value()
Refresh()
Today()
0..*
<<JavaScript>>
MeetingEvent
Subject : String
StartDate : Date
StopDate : Date
StartTime : Time
StopTime : Time
Summary()
Conflicts()
AddParticipant()
RemoveParticipant(
0..*
<<JavaScript>>
Participant
Name : String
Title : String
Email : String
Office : String
Phone : String
Summary()
<<Form>>
EventDetail
<<Text>> Subject : String
<<Select>> Participants : String
<<Checkbox>> AllDay : Boolean
<<Submit>> Update
<<Submit>> Remove
<<Button>> New
<<ActiveX>>
DatePicker
Day : int
Month : int
Year : int
Value()
Today()

그림 3. 클라이언트 협업

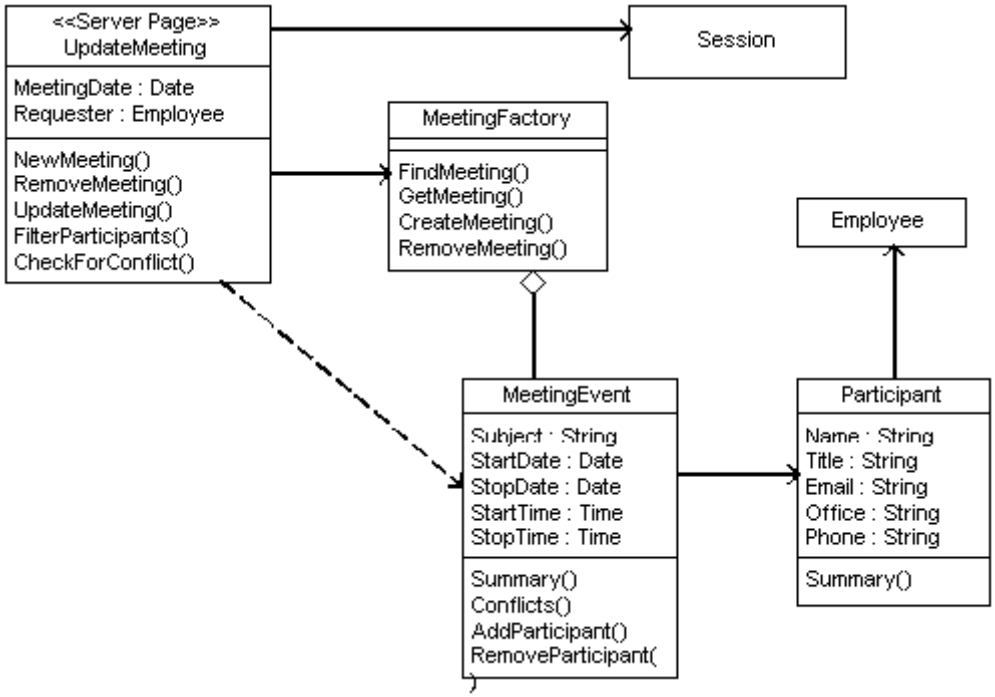

그림 4. 서버 협업

클래스 스테레오타입을 사용하여 웹 페이지의 논리 동작을 모델링하는 데 따른 가장 큰 이점 중 하나는 서버측 컴포넌트와의 협업을 다른 서버측 협업과 동일한 방식으로 표현할 수 있다는 것입니다. 서버 페이지는 시스템의 비즈니스 로직에 참여하는 다른 클래스일뿐입니다. 보다 개념적인 레벨에서 볼 때, 서버 페이지는 일반적으로 제어기 역할을 수행합니다. 즉, 필수 비즈니스 오브젝트 활동을 조율하여 브라우저의 페이지 요청으로 시작한 비즈니스 목적을 달성합니다.

클라이언트측에서 볼 때 협업은 조금 더 복잡해질 수 있습니다. 이는 채택할 수 있는 기술이 다양하기 때문이기도 합니다. 클라이언트 페이지는 가장 간단한 형태라도 컨텐츠 및 프리젠테이션 정보를 모두 포함하는 HTML 문서입니다. 브라우저는 페이지의 형식화 지시사항과 경우에 따라 별도 스타일시트를 사용하여 HTML 페이지를 렌더링합니다. 논리 모델의 경우, 이 관계는 클라이언트 페이지에서 스타일시트 스테레오타입 클래스로의 종속성으로 표현할 수 있습니다. 그러나 스타일시트는 주로 프리젠테이션 문제이고 일반적으로 ADM 의 범위에 포함되지 않습니다.

양식

웹 페이지의 기본 데이터 입력 메커니즘은 양식입니다. 양식은 HTML 문서에서 <form> 태그로 정의됩니다. 각 양식은 제출되는 대상 페이지를 지정합니다. 양식에는 모두 HTML 태그로 표현되는 여러 입력 요소가 포함됩니다. 가장 일반적인 태그는 <input>, <select> 및 <textarea> 태그입니다. 입력 태그는 텍스트 필드, 선택란, 단일 선택 단추, 누름 단추, 이미지, 숨겨진 필드 및 기타 자주 사용되지 않는 몇 가지 유형이 될 수 있으므로 자주 사용됩니다. 모델링 양식은 다른 클래스 스테레오타입의 양식을 의미합니다. <form> 태그에 정의되는 오퍼레이션은 실제로 클라이언트 페이지에서 소유하므로 양식에는 오퍼레이션이 없습니다. 양식의 입력 요소는 모두 양식 클래스의 스테레오타입화된 속성입니다. 양식은 입력 제어 기능을 수행하는 애플릿 또는 ActiveX 제어와 연관될 수 있습니다. 각 양식은 또한 서버 페이지(양식의 제출을 처리하는 페이지)와 연관됩니다. 이 관계는 제출 스트레오타입으로 지정됩니다. 양식은 HTML 문서에 완전히 포함되므로 올바른 집계 양식으로 UML 다이어그램에 표현됩니다. 그림 5 는 양식을 정의하고 처리 서버 페이지로 관계를 제출하는 간단한 장바구니 페이지를 보여줍니다.

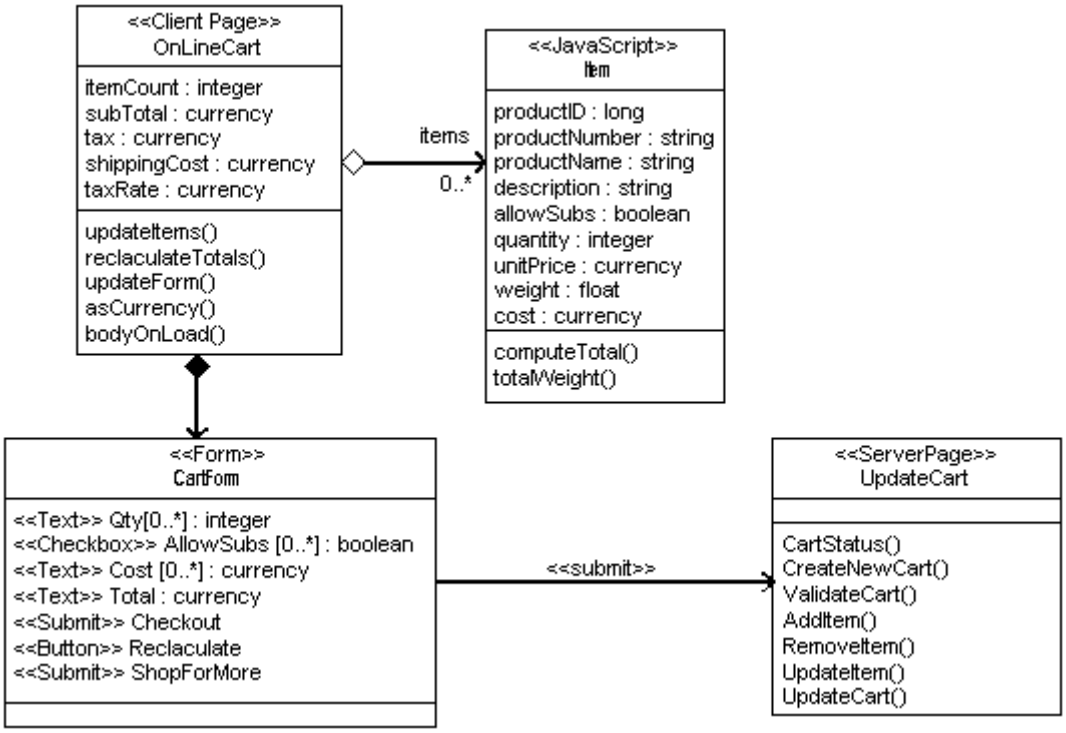

그림 5. 서버 페이지로 양식 제출

그림 5 에서 JavaScript 스테레오타입 클래스는 장바구니 항목을 나타내는 오브젝트입니다. 인스턴스 수가 가변적인 필드의 양식 속성에 대한 설명에는 배열 구문이 사용됩니다. 이 장바구니의 경우, 장바구니에 담을 수 있는 항목 수는 0 개 이상입니다. 각 항목은 Qty, AllowSubs, Cost 및 Total <input> 요소를 갖습니다.

클라이언트 페이지의 모든 활동은 JavaScript 로 실행되고 JavaScript 는 유형을 사용하지 않는 언어이므로 이러한 속성에 지정된 데이터 유형은 구현자 설명의 용도로만 사용됩니다. JavaScript 로 구현되거나 HTML 입력 태그로서 구현되는 경우 해당 유형은 무시됩니다. 이는 모델에 포함되는 함수 매개변수(이 그림에는 일부만 표시)에도 적용됩니다.

프레임

웹 사이트 또는 응용프로그램에서의 HTML 프레임 사용은 도입 시점부터 지금까지 양분화된 논쟁 주제입니다. 프레임을 사용하면 특정 시간에 여러 페이지가 활성화되어 사용자가 볼 수 있습니다. 오늘날 가장 일반적으로 사용되는 브라우저의 최신 기능 세트를 통해 사용자 시스템에서 여러 브라우저 인스턴스가 활성화될 수 있습니다. 동적 HTML 스크립트 및 컴포넌트를 사용하면 이러한 페이지가 서로 상호작용할 수 있습니다. 클라이언트에서 복잡한 상호작용이 발생할 가능성이 높으며 이와 관련된 모델링 요구는 더 높습니다.

특정 응용프로그램에서 프레임 또는 여러 브라우저 인스턴스를 채택하는지 여부는 소프트웨어 설계자가 결정합니다. 이러한 경우 이전과 동일한 이유로 인해 이 클라이언트측 동작의 모델을 ADM 으로 표시해야 합니다. 프레임 사용을 모델링하기 위해 두 가지 추가 클래스 스테레오타입(프레임세트 및 대상)과 연관 스테레오타입인 대상 링크를 정의합니다. 프레임세트 클래스는 컨테이너 오브젝트를 나타내며 HTML <frameset> 태그에 직접 맵핑됩니다. 이 클래스에는 클라이언트 페이지 및 대상이 포함됩니다. 대상 클래스는 다른 클라이언트 페이지에서 참조하는 이름이 지정된 프레임 또는 브라우저 인스턴스입니다. 대상 링크 연관은 다른 페이지에 대한 하이퍼링크이며 특정 대상에서 렌더링됩니다. 그림 6 의 예제에서는 공통 아웃라인 보기가 두 개의 프레임을 사용하는 브라우저에 표시됩니다. 한 프레임은 대상(컨텐츠)에 따라 이름이 지정되고 다른 프레임에는 클라이언트 페이지만 포함됩니다. 이 클라이언트 페이지 프레임은 서적의 목차(TOC)입니다. 이 페이지의

하이퍼링크는 컨텐츠 프레임에서 렌더링될 수 있도록 대상이 지정됩니다. 결과적으로 왼쪽에는 정적 목차가 표시되고 오른쪽 페이지에는 서적 컨텐츠와 장별 내용이 표시됩니다.

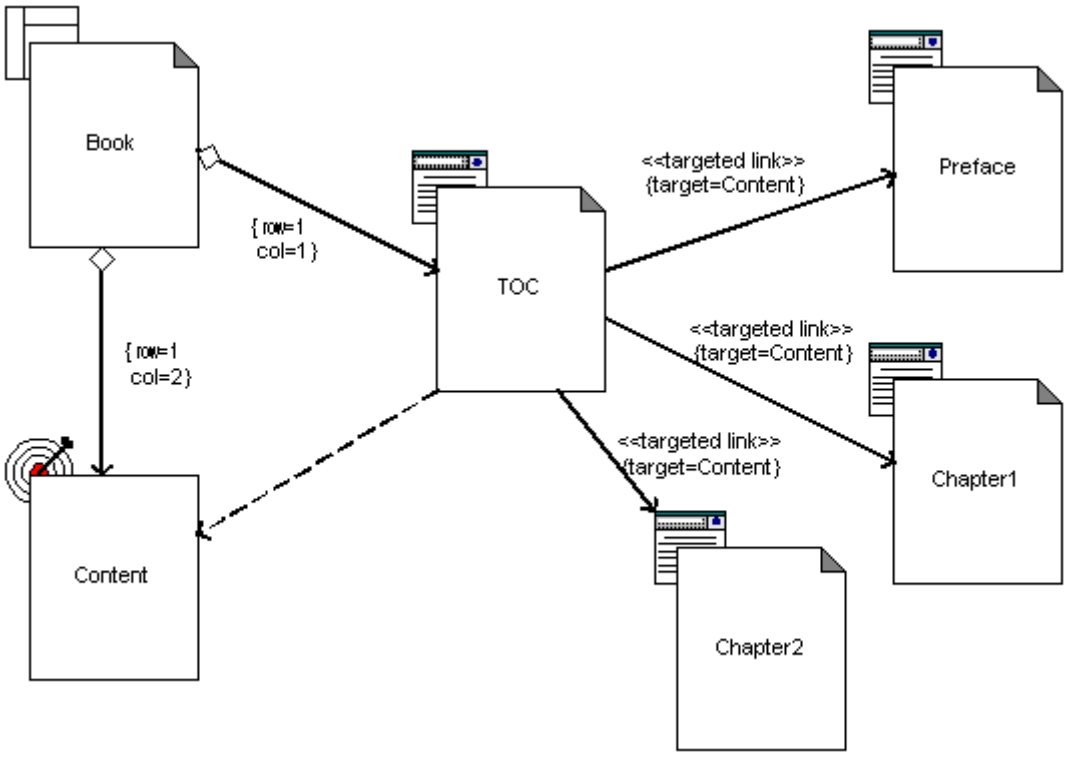

그림 6. 프레임 예제

실제 프리젠테이션 내용의 대부분은 프레임세트 및 연관의 태그 값으로 캡처됩니다. 프레임세트와 대상 또는 클라이언트 페이지 간의 집계 관계에 대한 두 개의 태그 값이 대상 또는 페이지가 속하는 프레임세트 행과 열을 지정합니다. 대상 링크 연관의 "대상" 태그 값은 페이지를 렌더링해야 하는 대상을 식별합니다.

대상이 프레임세트로 집계되지 않는 경우 별도 브라우저 인스턴스를 사용하여 페이지를 렌더링함을 의미합니다. 이 표기법은 클라이언트 시스템의 단일 인스턴스를 표현합니다. 독립적인 여러 대상은 모두 동일한 시스템에서 실행되는 것으로 가정되며 다이어그램은 특정 클라이언트 인스턴스의 클라이언트측 동작을 표현합니다. 다른 배치 구성은 보다 정확한 이해를 위해 모델에 자세히 기록되어야 합니다.

결론

이 문서에서 설명한 아이디어와 개념은 UML 로 웹 응용프로그램 특정 요소를 모델링하는 데 따른 문제와 솔루션에 대한 기본 정보입니다. 또한 이 문서의 목적은 웹 특정 요소 모델링을 나머지 응용프로그램과 통합할 수 있는 일관적이고 완벽한 방법을 제시함으로써 웹 응용프로그램의 디자이너, 구현자 및 설계자에게 적합한 세부사항 및 추상 레벨을 제공하는 것입니다. 웹 응용프로그램에 대한 정규 UML 확장의 첫 번째 버전은 완벽에 가깝게 개발되었습니다. 이 확장은 설계자와 디자이너에게 UML 을 사용하여 웹 응용프로그램 디자인 전체를 표현할 수 있는 일반적인 방법을 제공합니다.

이 확장에 대한 최신 정보는 Rational Rose 웹 사이트 및 Rational Software 웹 사이트의 UML 섹션을 참조하십시오.

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