UML을 사용하여 웹 어플리케이션 구조 모델링

Jim Conallen

Rational Software 백서

TP 157, 6/99
이 자료의 버전은 ACM 커뮤니케이션
1999년 10월(볼륨 42, 번호 10)
발행물에 있습니다.

Rational®
the software development company

목차

요약 1
개요 1
모델링 2
웹 어플리케이션 구조 3
웹 페이지 모델링 4
양식 7
프레임 8
결론 9

요약

웹 어플리케이션은 점점 복잡해지고 임무 결정적으로 되어 가고 있습니다. 이 복잡도를 관리하기 위해 웹 어플리케이션은 모델링되어야 합니다. UML(Unified Modeling Language)은 소프트웨어 집약 시스템을 모델링하는 표준 언어입니다. UML 을 사용하여 웹 어플리케이션을 모델링하면 일부 컴포넌트가 표준 UML 모델링 요소에 잘 맞지 않는다는 것이 명백해집니다. 전체 시스템(웹 컴포넌트 및 일반적인 중간 층 컴포넌트)에 하나의 모델링 표기법을 계속 사용하려면 UML 을 확장해야 합니다. 이 문서에서는 형식 확장자 메커니즘을 사용하여 UML 에 확장자를 표시합니다. 확장자는 웹 특정 컴포넌트를 시스템의 나머지 모델과 통합하여 설계자, 구현자 및 웹 어플리케이션 구조에 맞는 세부사항 및 추상화의 적절한 레벨을 표시하도록 설계됩니다.

개요

지난 몇 해 동안 새 용어가 IT 용어집에 입력되었습니다. 웹 어플리케이션 비즈니스 소프트웨어 시스템에 연관된 모두가 비즈니스와 관련되지 않은 다양한 소프트웨어 노력을 사용하여 웹 어플리케이션을 빌드할 계획이 있는 듯합니다. 이 구조의 다양한 초기 어댑터의 경우 시스템 자체와 같은 용어 웹 어플리케이션은 소규모 웹 사이트 추가에서 견고한 n 층의 어플리케이션으로 발전되었습니다. 전 세계에 분포된 수많은 동시 사용자에게 웹 어플리케이션을 제공하는 것은 드문 일이 아닙니다. 웹 어플리케이션 설계는 중요한 비즈니스입니다.

테스트에 입력시 용어 웹 어플리케이션은 사람마다 그 의미가 조금씩 다릅니다. 일부는 웹 어플리케이션이 Java 를 사용한다고 간주하며 다른 일부는 웹 어플리케이션이 웹 서버를 사용한다고 간주합니다. 일반적인 여론은 그 사이입니다. 따라서 이 문서에서는 웹 어플리케이션을 사용자 입력(탐색 및 데이터 입력)이 비즈니스 상태에 영향을 주는 웹 시스템(웹 서버, 네트워크, HTTP, 브라우저)으로 막연히 정의합니다. 이 정의는 웹 어플리케이션이 비즈니스 상태에서 소프트웨어 시스템이며 "앞 끝"이 웹 시스템을 통해 전달되는 큰 파트에 있다는 것을 확립하기 위한 것입니다.

일반 웹 어플리케이션 구조는 클라이언트 서버 시스템의 구조이며 몇몇 중요한 차이가 있습니다. 웹 어플리케이션의 가장 큰 이점 중 하나는 전개입니다. 웹 어플리케이션 전개는 일반적으로 서버측 컴포넌트를 네트워크에 설정하는 것에 관한 문제입니다. 클라이언트 파트에 특별한 소프트웨어 또는 형상은 필요하지 않습니다. 또 다른 중요한 차이점은 클라이언트와 서버 의사소통에 관한 특성입니다. 웹 어플리케이션의 기본 의사소통 프로토콜은 HTTP 로, 최대 의사소통 처리량 대신 견고성 및 장애 허용을 위해 설계된 연결이 없는 프로토콜입니다. 웹 어플리케이션에서 클라이언트 및 서버 간의 의사소통은 일반적으로 서버측과 클라이언트 측 객체 간의 직접 의사소통이 아닌, 웹 페이지 탐색에서 주기적으로 발생합니다. 웹 어플리케이션에 있는 모든 메시지 작업을 하나의 추상화 레벨에서 웹 페이지 엔티티 요청 및 수신으로 설명할 수 있습니다. 대체적으로 웹 어플리케이션 구조는 동적 웹 사이트의 구조와 크게 다르지 않습니다.

웹 어플리케이션과 웹 사이트의 차이점은 동적인 경우에도 해당 사용법을 수반합니다. 웹 어플리케이션은 비즈니스 논리를 구현하고 이것의 사용은 비즈니스의 상태를 변경합니다(시스템에서 캡처되는 것에 따라). 이는 모델링 노력의 중심을 정의하므로 중요합니다. 웹 어플리케이션은 비즈니스 논리를 실행하므로 가장 중요한 시스템 모델은 프리젠테이션 세부사항이 아닌, 비즈니스 논리 및 비즈니스 상태에 중점을 둡니다. 그러나 프리젠테이션은 중요하며(그렇지 않으면 시스템의 장점이 없음), 비즈니스와 프리젠테이션 고려사항을 명확한 구별하도록 해야 합니다. 프리젠테이션 실행이 중요하거나 복잡한 경우에는 이들 또한 모델링해야 하지만 반드시 비즈니스 논리 모델의 중요한 부분으로 할 필요는 없습니다. 또한 프리젠테이션에서 작업하는 자원은 더욱 예술적이며 비즈니스 규칙 구현에 대해서는 덜 고려하는 경향이 있습니다.

웹 서비스의 개발과 관련된 하나의 방법/표기법이 RMM(Relationship Management Methodology)입니다. RMM 은 인트라넷 및 인터넷 웹 시스템을 설계, 구축 및 유지보수하는 방법입니다. 주요 목적은 동적인 데이터베이스 구동 웹 사이트 유지보수 비용을 줄이는 것입니다. 이는 시스템의 비주얼 표시를 지지하여 설계 토론을 용이하게 합니다. 이는 웹 페이지에 있는 비주얼 요소의 분해 및 데이터베이스 엔티티와의 연관을 포함하는 반복 프로세스입니다. RMM 은 동적 웹 사이트를 작성하고 유지보수하는 " 전체적인 " 방법입니다.

RMM 은 웹 어플리케이션 빌드시에는 적용되지 않습니다. 비즈니스 논리의 중심이 되는 웹 어플리케이션에는 RMM 표기법에서 충분히 다루지 못하는 비즈니스 논리의 구현에 대한 다양한 설명적 메커니즘이 포함되어 있습니다. 클라이언트측 스크립트 작성, 애플릿 및 ActiveX 제어와 같은 설명은 종종 시스템의 비즈니스 규칙을

실행하는 데 상당히 기여합니다. 또한 웹 어플리케이션은 분배된 객체 시스템을 위한 전달 메커니즘으로 사용할 수 있습니다. 애플릿 및 ActiveX 제어는 독립 웹 서버인 DCOM 또는 RMI 를 통해 서버측 컴포넌트와 비동기적으로 상호 작용을 하는 컴포넌트를 포함할 수 있습니다. 정교한 어플리케이션도 자체 의사소통 메커니즘을 확립하고 유지보수하는 다중 브라우저 인스턴스 및 프레임을 클라이언트에서 사용합니다.

해당 메커니즘은 모두 시스템의 비즈니스 논리에 기여하므로 그렇게 모델링해야 합니다. 또한 이들 메커니즘은 비즈니스 논리 파트만을 표시하므로 나머지 시스템 모델과 통합해야 합니다. 대부분의 경우, 비즈니스 논리의 크기는 서버측 층 중 하나에 있는 웹 서버에서 결정됩니다. 모델링 언어 및 표기법 선택은 일반적으로 어플리케이션측의 필요에 의해 결정됩니다. OMG 에서 공식 객체 모델링 언어로 UML 을 채택하면 더 많은 시스템이 UML 표기법으로 표시됩니다. 대부분의 경우 UML 은 소프트웨어 집약 시스템 모델링에 선택되는 언어입니다. 모델링 웹 어플리케이션에 다음과 같은 기본 메시지가 발행됩니다. 나머지 어플리케이션과 함께 웹 특정 컴포넌트에서 실행되는 비즈니스 논리를 어떻게 표현합니까?의 답은 시스템의 비즈니스 논리 실행을 해당 웹 특정 요소 및 UML 이 있는 설명에 표현하는 능력에 따라 다릅니다.

이 책에서는 웹 어플리케이션 모델링에 관한 솔루션 및 실행에 대해 소개할 예정입니다. 웹 어플리케이션에 특정한 구조적으로 중요한 컴포넌트 및 UML 로 해당 컴포넌트를 모델링하는 방법을 중점적으로 다룹니다. 이 책의 독자는 UML, 객체 지향 이론 및 웹 어플리케이션 개발에 친숙해야 합니다. 이 책에 설명된 작업은 사실상 악의가 없는 가정을 기반으로 합니다.

- 웹 어플리케이션은 더욱 복잡해지고 더 많은 임무 임계 역할에 그 자체를 삽입하고 있는 소프트웨어 집약 시스템입니다.
- 소프트웨어 시스템에서 복잡도를 관리하는 한 가지 방법은 추상화하고 모델링하는 것입니다.
- 일반적으로 소프트웨어 시스템에는 각각이 다른 시점, 추상화 및 세부사항 레벨을 표시하는 여러 모델이 있습니다.
- 적절한 추상화 및 세부사항 레벨은 개발 프로세스의 결과물 및 활동에 따라 다릅니다.
- 소프트웨어 집약 시스템의 표준 모델링 언어는 UML 입니다.

이 책에 나온 개념 및 아이디어에 대한 더 충분한 취급 방법은 올 하반기에 Addison Wesley Longman 의 객체 설명 시리즈에 인쇄될 예정인 새로운 책 " UML 을 사용하여 웹 어플리케이션 빌드"에 개발되어 있습니다.

모델링

모델은 세부사항의 일부를 단순화하여 시스템을 이해하는 데 도움을 줍니다. 모델 선택은 문제점 및 솔루션의 형태를 이해하는 데 수많은 영향을 줍니다. 웹 어플리케이션은 다른 소프트웨어 집약 시스템과 같이 일반적으로 모델 세트(유스 케이스 모델, 구현 모델, 전개 모델, 보안 모델 등)로 표시됩니다. 웹 시스템에서 과도하게 사용되는 추가 모델은 사이트 맵, 웹 페이지 요약 및 시스템의 탐색 라우트입니다.

근래에 실행되는 대부분의 모델링 설명은 다양한 웹 어플리케이션 모델 개발에 적합하며 더 이상의 설명이 필요 없습니다. 그러나 매우 중요한 모델인 ADM(Analysis/Design Model)은 웹 페이지 및 이와 연관된 실행 가능한 코드를 모델에 있는 기타 요소와 함께 포함되도록 하는 데 약간의 어려움이 있습니다.

모델링하는 방법을 결정할 때 요약 및 세부사항의 올바른 레벨을 결정하는 것은 모델 사용자에게 이익이 되는 것을 제공하는 데 중요합니다. 대체로 시스템의 결과물(최종 제품을 생산하도록 구성되고 조작될 실생활 엔티티)을 모델링하는 것이 가장 좋습니다. 웹 서버의 내부 또는 웹 브라우저의 세부사항 모델링은 웹 어플리케이션의 설계자 및 아키텍트에게 도움을 주지 않습니다. 페이지 모델링, 서로에 대한 링크, 페이지 작성으로 이동하는 모든 동적 목차 및 클라이언트에 있는 페이지의 동적 목차는 매우 중요합니다. 이는 설계자가 설계하고 구현자가 구현한 결과물입니다. 모델링에는 클라이언트 및 서버의 페이지, 하이퍼링크 및 동적 목차가 필요합니다.

다음 단계는 해당 결과물을 모델링 요소로 맵핑하는 것입니다. 예를 들어, 하이퍼링크는 모델에 있는 연관 요소로 자연스럽게 맵핑됩니다. 하이퍼링크는 페이지 간의 탐색 경로를 표시합니다. 이 개념을 확장하여 페이지는 모델의 논리 보기에 있는 클래스로 맵핑합니다. 웹 페이지가 모델에 있는 클래스인 경우 페이지의

스크립트는 클래스의 조작으로 자연스럽게 맵핑합니다. 스크립트에 있는 페이지 범위의 모든 변수는 클래스 속성으로 맵핑합니다. 서버에서 실행하는 스크립트 세트(페이지의 동적 목차 준비) 및 클라이언트에서만 실행하는 완전히 다른 스크립트 세트(JavaScript)가 웹 페이지에 있다고 간주하면 문제점이 발생합니다. 시나리오에서 모델에 있는 웹 페이지 클래스 검토시 서버에서 사용 중인 조작, 속성 및 관계(페이지 준비 동안) 및 클라이언트에서 사용자가 페이지와 상호 작용시 사용 중인 것에 대한 혼동이 있습니다. 또한 웹 페이지는 웹 어플리케이션에 전달되는 대로 가장 적합하게 시스템 컴포넌트로 모델링됩니다. 웹 페이지를 UML 클래스로 맵핑만 해서는 시스템을 더 잘 이해할 수 없습니다.

UML 작성자는 상자 밖에서 특정 도메인 또는 구조의 관련 의미론을 캡처하는 데 UML 이 충분하지 못한 상황이 항상 있다는 것을 인식해야 합니다. 이를 처리하기 위해 참여자가 UML 의미론을 확장할 수 있도록 형식 확장자 메커니즘을 정의하였습니다. 메커니즘을 사용하여 모델 요소에 적용될 수 있는 *스테레오타입*, *태그 값* 및 *제한조건* 을 정의할 수 있습니다.

스테레오타입 은 모델링 요소에 대한 새 의미를 정의할 수 있는 장식입니다. *태그 값*은 모델링 요소에 값을 " 태그" 하도록 허용하는 모델링 요소와 연관될 수 있는 키 값 짝입니다. *제한조건* 은 모델의 훌륭한 형성을 정의하는 규칙입니다. 이는 자유로운 형식의 텍스트 또는 좀더 형식적인 OCL(Object Constraint Language)로 표현할 수 있습니다.

이 책에서 다루는 작업은 웹 어플리케이션의 UML 에 확장자를 도입합니다. 확장자는 완전히 이 책의 범위 밖이지만 대부분의 개념 및 설명이 여기에 설명되어 있습니다.

마지막 모델링 위치에서 비즈니스 논리 및 프리젠테이션 논리 간에 명확한 구분이 필요합니다. 일반적인 비즈니스 어플리케이션의 경우 비즈니스 논리만 ADM 의 파트이어야 합니다. 프리젠테이션 세부사항(활성 단추, 플라이-오버 도움말 및 기타 UI 개선사항)은 일반적으로 ADM 에 속하지 않습니다. 별도의 UI 모델이 어플리케이션에 구성되는 경우에는 해당 위치에 이러한 사항이 배치됩니다. 솔루션 공간 및 비즈니스 문제점을 표현하는 데 중점을 두도록 ADM 을 유지해야 합니다. 해당 웹 결과물 날짜에 웹 페이지의 룩앤필은 일반적인 개발자가 아닌, 전문가(설명적 그래픽 아티스트)에 의해 가장 잘 설계되고 구현됩니다.

웹 어플리케이션 구조

웹 어플리케이션의 기본 구조에는 브라우저, 네트워크 및 웹 서버가 포함됩니다. 브라우저는 서버로부터 웹 페이지를 요청합니다. 각 페이지는 HTML 로 표시되며 컨텐츠과 형식 지시사항이 혼합되어 있습니다. 일부 페이지에는 브라우저에서 해석되는 클라이언트측 스크립트가 포함됩니다. 이런 스크립트는 표시 페이지에 대한 추가적인 동적 작동을 정의하며 종종 브라우저, 페이지 컨텐츠 및 페이지에 포함된 추가 제어(애플릿, ActiveX 제어 및 플러그인)와 상호 작용합니다. 사용자는 페이지의 컨텐츠를 검토하고 상호 작용합니다. 사용자는 때때로 페이지에 있는 필드 요소에 정보를 입력하고 처리를 위해 서버에 제출합니다. 사용자 는 또한 하이퍼링크를 통해 시스템에 있는 다른 페이지를 탐색하여 시스템과 상호 작용할 수 있습니다. 어느 경우에나 사용자는 시스템의 비즈니스 상태를 변경할 수 있는 입력을 시스템에 제공합니다.

클라이언트의 관점에서는 웹 페이지는 항상 HTML 형식의 문서입니다. 그러나 서버에서 웹 페이지는 수많은 다른 방법의 manifest 파일 자체일 수 있습니다. 최초의 웹 어플리케이션에서 동적 웹 페이지는 CGI(Common Gateway Interface)를 사용하여 빌드되었습니다. CGI 는 스크립트 및 컴파일된 모듈의 인터페이스가 페이지 요청에 따라 전달되는 정보에 대한 액세스를 획득하는 데 사용되도록 정의합니다. CGI 기반 시스템에서 특수 디렉토리는 일반적으로 웹 서버에서 구성되며 페이지 요청에 응답하는 스크립트를 실행할 수 있습니다. CGI 스크립트가 요청되면 서버는 다른 HTML 형식 파일의 경우처럼 파일의 컨텐츠를 리턴만 하는 대신, 파일을 적절한 인터프리터(일반적으로 PERL 쉘)로 처리 또는 실행하고 출력을 요청 클라이언트로 다시 스트림합니다. 최종 처리 결과는 요청 클라이언트로 다시 송신되는 HTML 형식 스트림입니다. 파일이 처리되는 동안 시스템에서 비즈니스 논리가 실행됩니다. 이 시간 동안 서버측 자원(예: 데이터베이스 및 중간 층 컴포넌트)과 상호 작용할 가능성이 있습니다.

현재의 웹 서버는 이 기본 설계에서 향상되었습니다. 지금은 훨씬 더 보안을 인식하며 몇 가지만 예를 들면, 서버에서 클라이언트 상태 관리, 트랜잭션 처리통합, 원격 관리 및 자원 풀과 같은 기능이 포함되어 있습니다. 전체적으로 최근에 생성된 웹 서버는 임무 결정적, 확장 가능하고 견고한 어플리케이션의 아키텍트에게 해당 실행의 중요성을 알리고 있습니다.

CGI 스크립트 역할을 검토해볼 때 현재의 웹 서버는 세 개의 주요 카테고리(스크립트된 페이지, 컴파일된 페이지 및 이 둘의 혼합)로 나눌 수 있습니다. 첫 번째 카테고리에서 클라이언트 브라우저가 요청할 수 있는 모든 웹 페이지는 웹 서버의 파일 시스템에서 스크립트된 파일로 표시됩니다. 일반적으로 이 파일은 HTML 과 다른 스크립트 언어가 섞여 있습니다. 페이지가 요청되면 웹 서버는 페이지 처리를 페이지를 인식하는 엔진에 위임하며, 최종 결과는 HTML 형식 스트림이 요청 클라이언트로 다시 송신됩니다(예: Microsoft’의 Active Server Pages, Java Server Pages 및 Cold Fusion).

두 번째 카테고리, 컴파일된 페이지에서 웹 서버는 2 진 컴포넌트를 로드하여 실행합니다. 스크립트된 페이지에서와 같이 이 구성요소에는 페이지 요청에 수반되는 모든 정보(매개변수 및 양식 필드 값)에 대한 액세스가 있습니다. 컴파일된 코드는 요청 세부사항을 사용하며 일반적으로 서버측 자원에 액세스하여 클라이언트에 리턴되는 HTML 스트림을 작성합니다. 규칙은 아니지만 컴파일된 페이지가 스크립트된 페이지보다 더 큰 기능성으로 처리하는 경향이 있습니다. 매개변수를 컴파일된 페이지 요청에 전달하여 다른 기능성을 얻을 수 있습니다. 하나의 컴파일된 컴포넌트에 실제로는 전체 디렉토리의 스크립트된 페이지 기능이 모두 포함되어 있을 수 있습니다. 이 유형의 구조를 표시하는 설명은 Microsoft ISAPI 및 Netscape NSAPI 입니다.

세 번째 카테고리는 요청되어 컴파일된 스크립트된 페이지를 나타내며 해당 컴파일된 버전은 이후 모든 후속 요청에 사용됩니다. 원래 페이지의 컨텐츠가 변경되는 경우에만입니다. 반면, 페이지는 또 다른 컴파일을 만나게 됩니다. 이 카테고리는 스크립트된 페이지의 유연성과 컴파일된 페이지의 효율성을 절충한 것입니다.

웹 페이지 모델링

스크립트된 또는 컴파일된 웹 페이지를 UML 에 있는 컴포넌트로 1:1 맵핑합니다. 컴포넌트는 시스템의 실제뱲파트이며 대체 가능한 파트입니다. 모델의 구현 보기(컴포넌트 보기)에서는 시스템 컴포넌트 및 이들의 관계에 대해 설명합니다. 웹 어플리케이션에서 구현 보기는 시스템에 있는 모든 웹 페이지 및 서로의 관계(하이퍼링크)를 설명합니다. 하나의 레벨에서 웹 시스템의 컴포넌트 다이어그램은 사이트 맵과 같습니다.

컴포넌트는 인터페이스의 실제 패키지만을 표시하므로 페이지 내부의 협력 모델링에는 적합하지 않습니다. 이 요약 레벨은 설계자 및 구현자에게 매우 중요하며 계속 모델의 파트가 되어야 합니다. 시작하기 위해 모든 웹 페이지는 모델의 설계 보기(논리 보기)에 있는 UML 클래스라고 할 수 있으며, 다른 페이지에 대한 관계(연관)는 하이퍼링크를 나타낸다고 할 수 있습니다. 그러나 제공된 웹 페이지는 서버에만 있는 기능 및 협력 세트를 나타낼 수 있고, 전혀 다른 세트는 클라이언트에만 있다고 간주하면 개념이 분류됩니다. 출력 파트로 동적 HTML(클라이언트측 스크립트 작성)을 채택하는 웹 페이지를 스크립트한 서버는 그러한 페이지의 한 예입니다. 이 문제점에 대한 반사적인 반응은 서버 또는 클라이언트측에서 올바른지 여부를 표시하도록 클래스에 있는 모든 속성 또는 조작을 정형화할 수 있습니다. 이점에서 이 모델은 처음에는 문제를 단순화하도록 하는 경향이 있었으며 꽤 복잡해지고 있습니다.

문제점에 대한 더 좋은 방법은 고려사항 구분의 원리를 고려하는 것입니다. 논리적으로 말하면 서버에서의 웹 페이지 동작은 클라이언트에서와 완전히 다릅니다. 서버에서 실행 중인 동안 서버측 자원(중간 층 컴포넌트, 데이터베이스, 파일 시스템 등)에 대한 액세스(관계)가 있습니다. 같은 페이지 또는 해당 페이지의 스트림된 HTML 출력은 클라이언트에서 전혀 다른 작동 및 관계 세트를 갖습니다. 클라이언트의 스크립트된 페이지에는 DOM(Document Object Model)을 통한 브라우저 자체와의 관계 및 Java 애플릿, ActiveX 제어 또는 페이지가 지정하는 플러그인과의 관계가 있습니다. 주요 설계자의 경우또 다른 HTML 프레임 또는 브라우저 인스턴스에 나타나는 클라이언트에서 다른“ 활성” 페이지와 추가의 관계가 있을 수 있습니다.

고려사항과 별도로 하나의 클래스가 있는 웹 페이지의 서버측 관점 및 다른 하나가 있는 클라이언트측 관점을 모델링할 수 있습니다. UML 의 확장자 메커니즘을 사용하여 이 둘을 구분하며 각각의 «서버 페이지» 및 «클라이언트 페이지»에 대한 스테레오타입 및 아이콘을 정의합니다. UML 에 있는 스테레오타입을 사용하여 모델링 요소에 새 의미론을 정의할 수 있습니다. 사용자 정의 아이콘을 사용하여 스테레오타입된 클래스를 UML 다이어그램에 표현하거나 guillemet(«») 간의 스테레오타입 이름으로 간단하게 장식할 수 있습니다. 아이콘은 개요 다이어그램에 유용하며 클래스 속성 및 조작이 노출되는 경우 단순한 태그를 사용하는 것이 가장 적합합니다.

웹 페이지의 경우, 스테레오타입은 클래스가 클라이언트 또는 서버에서 웹 페이지의 논리적 작동에 대한 요약임을 표시합니다.두 개의 요약은 둘 사이의 방향성 관계로 서로 관련되어 있습니다. 이 연관은 서버 페이지가 클라이언트 페이지를 빌드한다고 할 수 있으므로(그림 1) «빌드»로 정형화됩니다. 모든 동적 웹 페이지(런타임시 컨텐츠가 결정되는 페이지)는 서버 페이지로 구성됩니다. 모든 클라이언트 페이지는 단지 하나의 서버 페이지에서 빌드되나, 서버 페이지에서 다중 클라이언트 페이지를 빌드할 수는 없습니다.

웹 페이지 간의 공통 관계는 하이퍼링크입니다. 웹 어플리케이션의 하이퍼링크는 시스템에서의 탐색 경로를 나타냅니다. 이 관계는 모델에 «링크» 정형화된 연관을 사용하여 모델에 표현됩니다. 연관은 항상 클라이언트 페이지에서 유래되며 클라이언트 또는 서버 페이지를 가리킵니다.

웹 페이지 요청 및 웹 페이지가 구현 보기에서 컴포넌트로 모델링되면 하이퍼링크가 시스템에서 구현됩니다. 클라이언트 페이지에 대한 링크 연관은 대부분의 경우 클라이언트 페이지를 빌드하는 서버 페이지에 대한 링크 연관과 같습니다. 사실상 링크는 클래스 요약 중 하나가 아닌, 페이지에 대한 요청이기 때문입니다. 웹 페이지 컴포넌트는 페이지 요약 모두를 인식하므로 페이지 컴포넌트에서 인식한 모든 클래스의 링크는 같습니다.

태그 값은 링크 요청에 따라 전달되는 매개변수를 정의하는 데 사용됩니다. «링크» 연관 태그 값 " 인수" 는 요청을 처리하는 서버 페이지에서 예측되고 사용되는 매개변수 이름(및 옵션 값)의 목록입니다. 그림 2 에서 SearchResults 페이지는 각 링크마다 productId 매개변수에 대한 다른 값을 가진 GetProduct 서버 페이지에 대한 하이퍼링크(0..*) 변수를 포함합니다. GetProduct 페이지는 productId 매개변수가 지정하는 제품의 ProductDetail 페이지를 빌드합니다.

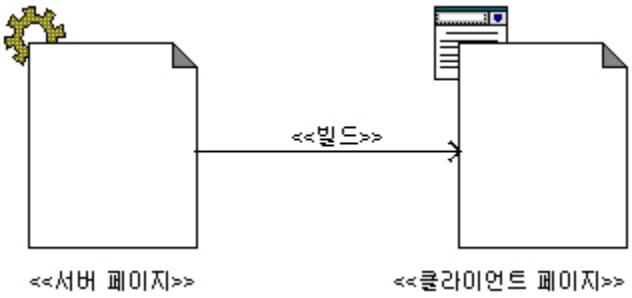

그림 1. 클라이언트 페이지를 빌드하는 서버 페이지

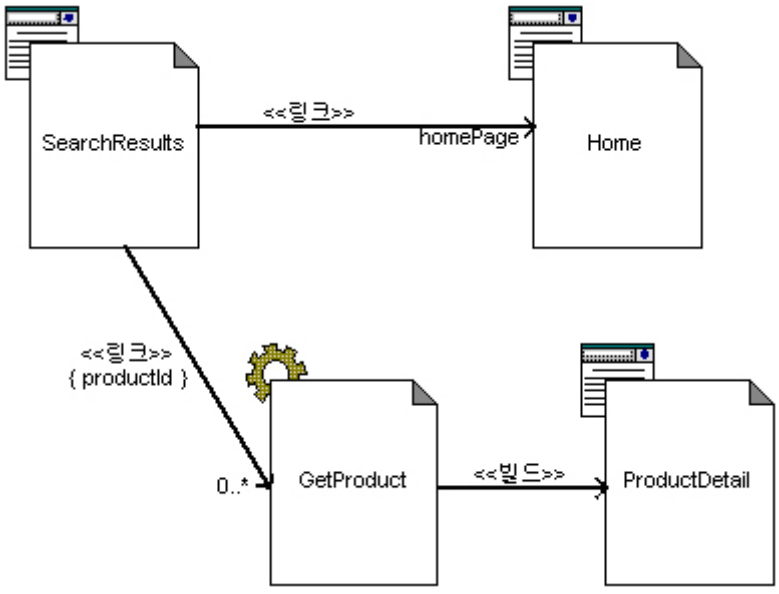

그림 2. 하이퍼링크 매개변수 사용

이러한 스테레오타입을 사용하면 페이지의 스크립트 및 관계 모델링이 쉬워집니다. «서버 페이지» 클래스 조작은 페이지의 서버측 스크립트에서 함수가 되며 해당 속성은 페이지 범위의 변수가 됩니다(페이지 함수로 글로벌 액세스 가능). «클라이언트 페이지» 클래스 조작 및 속성 또한 클라이언트에서 가시적인 함수 및 변수가 됩니다. 페이지의 서버 및 클라이언트측 관점을 다른 클래스로 분리하는 것에 대한 핵심 장점은 페이지 및 기타 시스템 클래스 간의 관계에 있습니다. 클라이언트 페이지는 관계를 사용하여 클라이언트측 자원(DOM, Java 애플릿, ActiveX 제어 및 플러그인)으로 모델링됩니다(그림 3). 서버 페이지는 관계를 사용하여 서버측 자원(중간층 컴포넌트, 데이터베이스 액세스 컴포넌트, 서버 운영 체제 등)으로 모델링됩니다(그림 4).

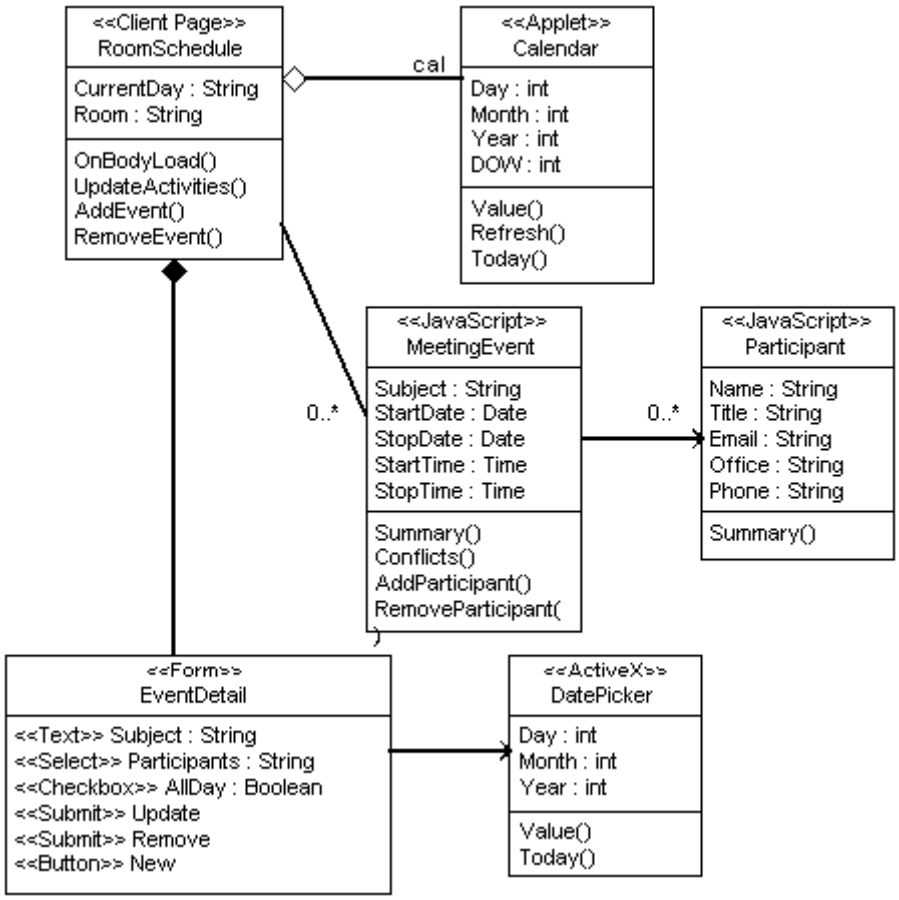

그림 3. 클라이언트 협력

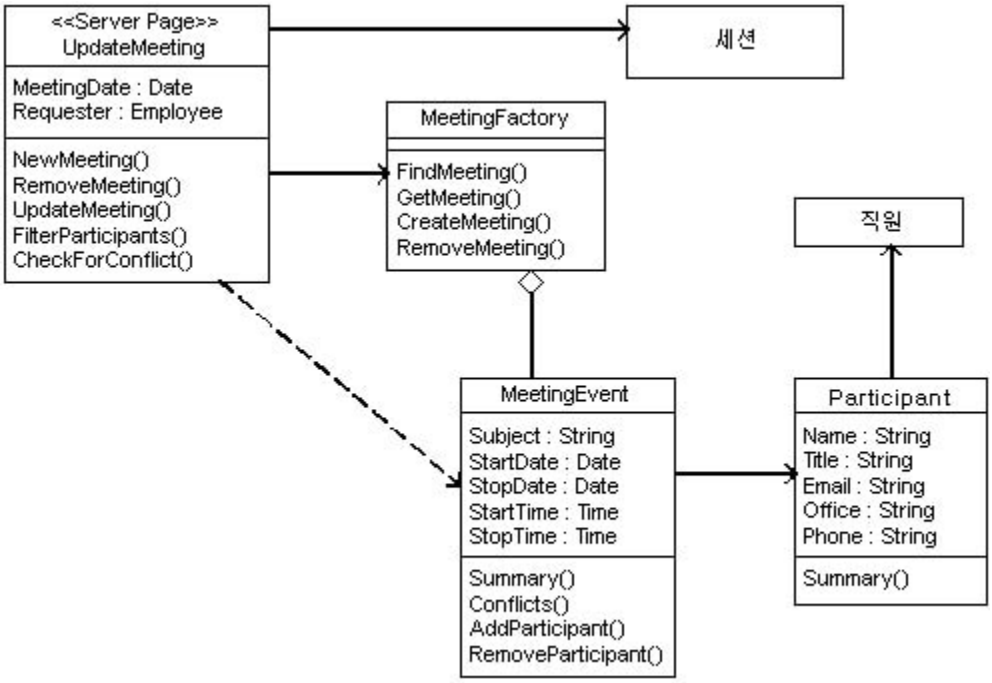

그림 4. 서버 협력

웹 페이지의 논리적 작동을 모델링하는 데 클래스 스테레오타입을 사용하는 가장 큰 장점 중 하나는 다른 서버측 협력과 같은 방법으로 표현할 수 있는 서버측 컴포넌트와의 협력입니다. «ServerPage»는 단지 시스템의 비즈니스 논리에 참여하는 다른 클래스입니다. 더 개념적인 레벨에서 보면, 일반적으로 서버 페이지는 브라우저의 페이지 요청으로 시작된 비즈니스 목표를 달성하도록 필수 비즈니스 객체 활동을 조정하는 제어기 역할을 수행합니다.

클라이언트측에서 협력은 조금 복잡해질 수 있습니다. 이는 채택할 수 있는 기술이 다양하기 때문입니다. 가장 단순한 클라이언트 페이지는 목차 및 프리젠테이션 정보 모두가 들어 있는 HTML 문서입니다. 브라우저는 페이지에서 형식화 지시사항을 사용하여 HTML 페이지를 표현하며 종종 별도의 스타일시트를 사용합니다. 논리 모델에서 이 관계는 클라이언트 페이지에서 «Stylesheet» 스테레오타입된 클래스로의 종속성으로 표현할 수 있습니다. 그러나 스타일시트는 기본적으로 프리젠테이션를 실행하며 종종 ADM 밖에 있습니다.

양식

웹 페이지에 대한 기본 데이터 항목 메커니즘은 양식입니다. 양식은 HTML 문서에 <form> 태그와 함께 정의됩니다. 각각의 양식은 스스로에게로 제출되는 페이지를 지정합니다. 양식은 많은 입력 요소를 포함하며 이런 요소 모두는 HTML 태그로 표시됩니다. 가장 일반적인 태그는 <input>, <select> 및 <textarea> 태그입니다. 입력 태그는 텍스트 필드, 선택란, 단일 선택 단추, 누름 단추, 이미지, 숨겨진 필드는 물론, 기타 일반적이지 않은 몇몇 유형이 될 수 있으므로 다소 과부하됩니다. 모델링 양식은 다른 클래스 스테레오타입 « form»을 의미합니다. <form> 태그에 정의될 수 있는 모든 조작은 실제로 클라이언트 페이지가 소유하므로 « form»에는 조작이 없습니다. 양식의 입력 요소는 모두 « form» 클래스의 스테레오타입된 속성입니다. « form»에는 입력 제어로 활동하는 애플릿 또는 ActiveX 제어와의 관계가 있을 수 있습니다. 또한 모든 양식에는 서버 페이지(양식 제출을 처리하는 페이지)와의 관계가 있습니다. 관계는 «submit»으로 스테레오타입되어 있습니다. 양식이 완전히 HTML 문서에 들어 있으므로 UML 다이어그램에서 강력한 집합 양식과 함께 표시됩니다. 그림 5 에서는 양식을 정의하고 서버 페이지 처리에 대한 제출 관계를 표시하는 단순한 장바구니 페이지를 보여줍니다.

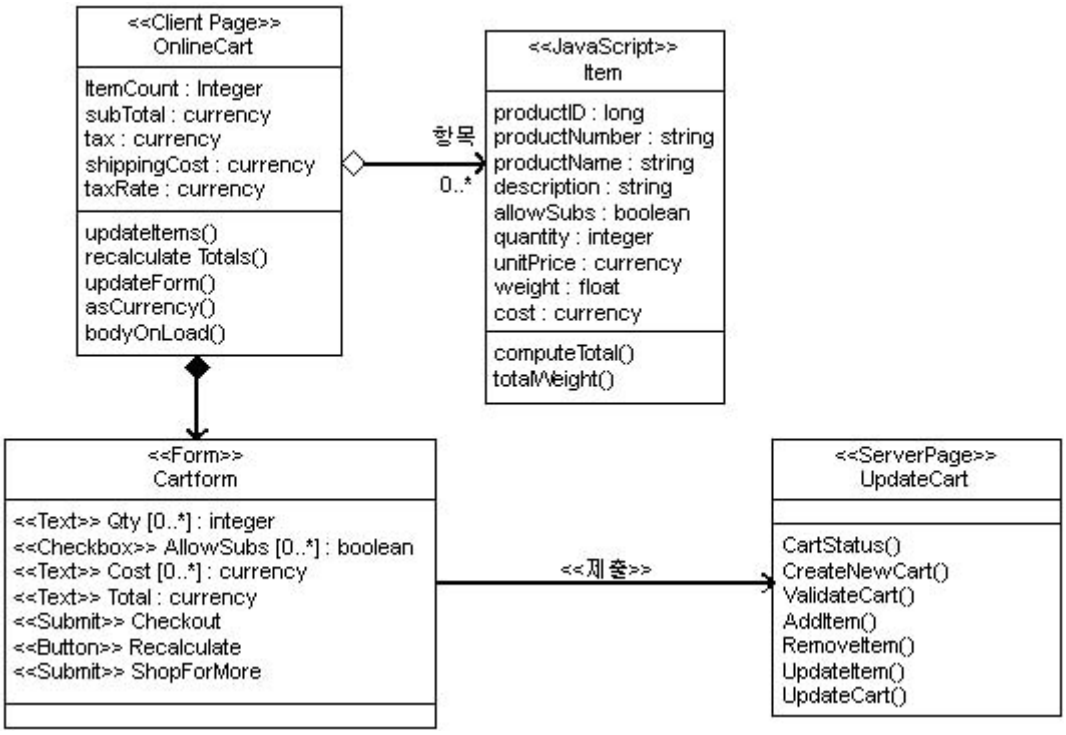

그림 5. 서버 페이지에 양식 제출

그림 5 에서 «JavaScript» 스테레오타입된 클래스는 장바구니에 있는 항목을 표시하는 객체입니다. 배열 구문은 인스턴스 변수가 들어 있는 필드에 대한 양식의등록 정보를 설명하는 데 사용됩니다. 이 장바구니의 경우 카트에는 0 에서 많은 항목까지를 넣을 수 있으며 각각은 Qty, AllowSubs, Cost 및 Total <input> 요소가 있습니다.

클라이언트 페이지에서의 모든 활동은 JavaScript 로 실행되며 JavaScript 는 유형이 없는 언어이므로, 이 속성에 지정되는 데이터 유형은 구현 설명에만 사용됩니다. JavaScript 또는 HTML 입력 태그로 구현되는 경우 유형은 무시됩니다. 이는 이 그림에서 완전히 표시되지는 않았지만 모델의 일부인 함수 매개변수에도 적용됩니다.

프레임

웹 사이트 또는 어플리케이션에서의 HTML 프레임 사용은 도입 이래 양극화된 논쟁의 주제가 되었습니다. 프레임을 사용하면 언제든지 다중 페이지를 활성화하고 사용자에게 표시할 수 있습니다. 요즘 가장 일반적인 브라우저의 최근 기능 세트는 다중 브라우저 인스턴스에서 사용자 시스템에서 사용할 수 있도록 합니다.이런 페이지는 동적 HTML 스크립트 및 컴포넌트를 사용하여 상호 작용합니다. 클라이언트에서의 복잡한 상호 작용 가능성은 상당하며 모델링에 대한 필요성은 더욱 커집니다.

어플리케이션에 프레임 또는 다중 브라우저 인스턴스를 채택할지 여부는 소프트웨어 아키텍트에 대한 결정입니다. 결정 후에는 이전과 같은 이유로 클라이언트측 모델 작동을 ADM 에 표시해야 합니다. 프레임 사용법을 모델링하기 위해 추가로 두 개의 클래스 스테레오타입 «frameset», «target» 및 연관 스테레오타입 «targeted link»에 대해 정의합니다. 프레임 세트 클래스는 컨테이너 객체를 표시하고 HTML <frameset> 태그로 직접 맵핑합니다. 여기에는 클라이언트 페이지 및 대상이 들어 있습니다. 대상 클래스는 기타 클라이언트 페이지에서 참조하는 명명된 프레임 또는 브라우저 인스턴스입니다. 대상 링크 연관은 다른 페이지로의 하이퍼링크이며 특정 대상에서 표현됩니다. 그림 6 에 표시된 예제에서, 일반적인 개요 보기는 두 개의 프레임을 사용하여 브라우저에 표시됩니다. 하나의 프레임은 대상(컨텐츠)으로 명명되는 반면, 다른 프레임에는 클라이언트 페이지만 들어 있습니다. 이 클라이언트 페이지 프레임은 책의 TOC(Table of Contents)입니다.

페이지에 있는 하이퍼링크는 대상이며 컨텐츠 프레임에서 표현됩니다. 결과적으로 왼쪽에는 정적 목차가, 오른쪽 페이지에는 책의 컨텐츠가 장별로 위치합니다.

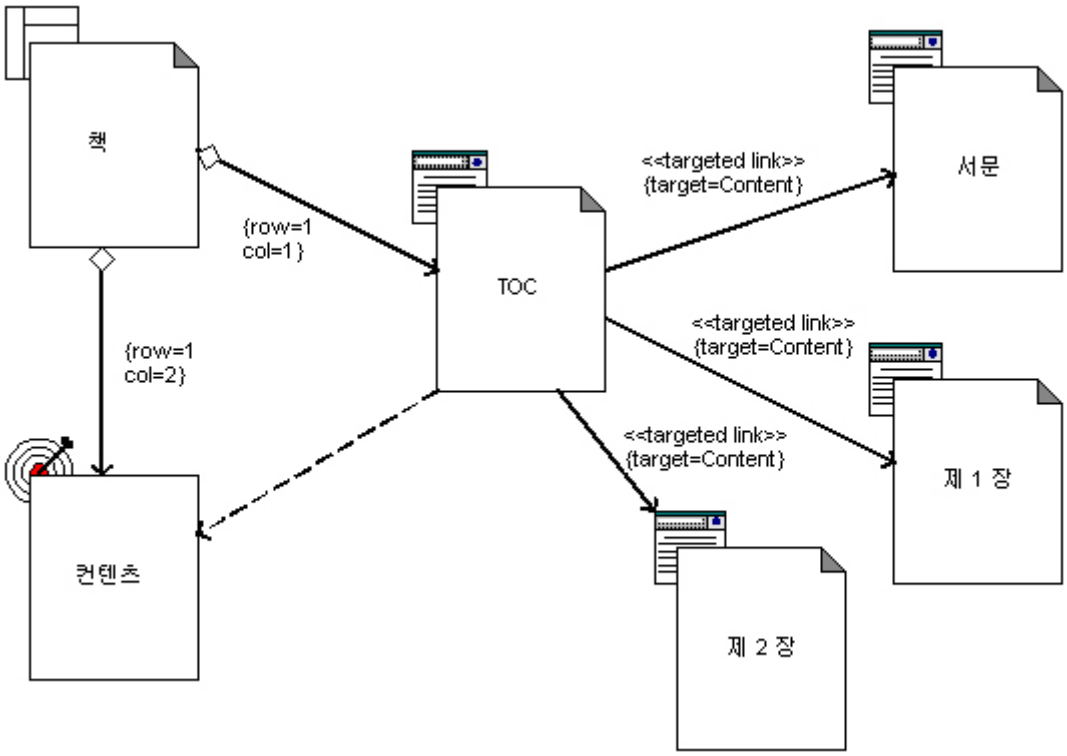

그림 6. 프레임 예

실제 프리젠테이션의 세부사항 대부분은 프레임 세트 및 연관에서 태그 값으로 캡처됩니다. 프레임 세트와 대상 또는 클라이언트 페이지 간의 집합 관계에서 두 개의 태그 값은 대상 또는 페이지가 속한 프레임 세트 행 및 열을 지정합니다. 대상 링크 연관에서 태그 값 " target" 은 페이지를 표현해야 하는 «target»을 식별합니다.

대상이 프레임 세트의 집합이 아니면 페이지를 표현하는 데 별도의 브라우저 인스턴스를 사용하는 것을 의미합니다. 이 표기법은 클라이언트 시스템의 인스턴스 하나를 표시한다는 것을 기억해야 합니다. 다중 독립 대상은 모두 같은 시스템에서 실행되는 것으로 간주되며, 다이어그램은 하나의 클라이언트 인스턴스의 클라이언트측 작동을 표시합니다. 기타 다른 전개 형상은 더 나은 이해를 돕기 위해 모델에서 자세하게 문서화해야 합니다.

결론

이 책에서 논의되는 아이디어 및 개념은 UML 을 사용하는 웹 어플리케이션 특정 요소의 모델링에 대한 실행 및 솔루션에 대한 소개입니다. 이 작업의 목적은 나머지 어플리케이션과 모델링 웹 특정 요소를 통합하는 일관되고 완전한 방법을 제공하며 세부사항 및 요약 레벨은 웹 어플리케이션 설계자, 구현자 및 기획자에게 적절합니다. 웹 어플리케이션용 UML 에 대한 형식 확장자의 첫 번째 버전이 거의 완료되었습니다. 이 확장자는 기획자 및 설계자에게 UML 을 사용하는 웹 어플리케이션 설계의 전체를 표시하는 일반적인 방법을 제공하게 됩니다.

이 확장자에 대한 최신 정보는 Rational Rose 의 인터넷 및 Rational Software 웹 사이트의 UML 절에 있습니다.

Rational®
the software development company

본사:

Rational Software
18880 Homestead Road
Cupertino, CA 95014
전화 번호: (408) 863-9900

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

무료 전화번호: (800) 728-1212
전자 우편: info@rational.com
웹: www.rational.com
월드와이드: www.rational.com/worldwide