Restlet 소개

약력

웹과 자바는 이미 널리 알려진 대로 긴 역사를 가지고 있다. 1994 년에 만들어진 이후로 자바는 java.net 패키지와 HttpURLConnection 클래스를 이용하는 특정 HTTP 클라이언트를 포함하고 있다. 당시 애플릿은 기존 웹 서버로 콜백을 수행하기 위하여 웹 브라우저상에서 실행되는 인기있는 기술이었다. HTTP 클라이언트를 이용하는 것은 이러한 작업을 하기에 좋은 방법이었으며, 방화벽과 관련된 보안 문제를 제한하였다.

그 이후로 1998 년 즈음, HTTP 서버상의 동적 컨텐츠 생성을 위한 한 방법으로서 서블릿 API 가 소개되었다. 서블릿 API 는 HTTP 요청/응답 주기를 객체지향적인 모델로 표현하려고 하였다. 자매 스펙인 자바 서버 페이지(JSP)와 함께, 서블릿 API 는 자바 테크놀로지를 기업환경으로 가져가도록 하는데 크게 이바지하였다. 자바 테크놀로지를 사용하는 애플리케이션에서는 개발자는 보통 "모델 2" 접근법이라 알려져 있기도 한 객체지향적인 MVC (Model-View-Controller) 디자인 패턴을 도입하였다. 이 모델에서 서블릿은 컨트롤러로서, JSP 페이지는 뷰로서, 그리고 자바빈(JavaBean) 객체는 모델로서 수행된다.

같은 시기에 XML 표준이 W3C 워킹 그룹에서 출현하였다. XSLT, XSL-FO, XPATH 표준과 함께, XML 은 서블릿과 JSP 페이지에 필적하는 동적 페이지를 생성하는 새로운 방법을 제공해 주었다. 그 결과 XML 은 모든 자바 플랫폼과 JSP 에 받아들여져 XML 문서를 생성할 수 있도록 발전되었다. 또 다른 방향은 아파치 코쿤 프로젝트를 시작한 Stefano Mazzocchi 에 의해 주도된 것인데, 관심의 분리(separation of concern)과 컴포넌트 지향(component-oriented) 디자인 개념에 따라 구축된 XML 퍼블리싱 프레임워크(XML publishing framework)였다.

2000 년, 스트러츠(Struts) 프로젝트에서는 기존 모델 2/MVC 패턴에 따라 액션(Action)이라 불리는 표준 컨트롤러를 정의하였다. 스트러츠에서는 ActionForms 를 사용하여 모델과 뷰 간의 애플리케이션 상태를 교환한다. 스트러츠는 특히 폼 처리에 있어 순수 서블릿이 제공하는 것보다 더 높은 수준의 추상화를 제공하여 빠르게 인기를 얻었다. 그 후 수많은 다른 프레임워크들이 비슷한 문제를 해결하기 위해 나타났다. 가장 주목할 만한 것은 Spring 인데, 스트러츠보다 더 포괄적인 자바/J2EE 애플리케이션 프레임워크이며 MVC 접근법 또한 구현하고 있다.

같은 해 HTTP 와 URI 스펙의 공저자이며 아파치 HTTP 서버의 핵심 기여자였던 Roy T. Fielding 은 소프트웨어 아키텍처에 관한 논문을 작성하였다. 이 논문의 5 장에서 그는 공식적으로 웹을 지원하는 아키텍처 스타일을 정의하였고 그것을 REST(REpresentational State Transfer)라 이름지었다. REST 는 객체지향과는 대조적으로 자원이 도메인의 식별가능한 개념(객체에 필적하는)들을 표현하는 자원지향(resource-oriented)이라는 새로운 패러다임을 정의했다. 자원은 표준 URI(URL 이나 URN)를 이용하여 참조되며 통일된 인터페이스를 통해 컴포넌트(브라우저, 서버, 프록시, 게이트웨이 등)에 의해 조작된다. 이러한 인터페이스는 한정된 동사의 목록을

가지는데, 본질적으로는 HTTP 메소드들이다. 또한 자원은 절대로 직접적으로 교환되지 않으며 오직 그 상태에 대한 표현에 의해 교환된다. 커넥터(Connector)는 예를 들자면 HTTP 프로토콜의 클라이언트 측을 구현하는 컴포넌트들간의 표현들의 통신을 가능하게 해주는 아키텍처 요소이다.

서블릿의 제한점

2003 년 말, Jetty 웹 컨테이너의 저자이며 서블릿 스펙의 기여자인 Greg Wilins 는 서블릿에 관한 몇 가지 문제들을 블로그에 올렸다:

- 프로토콜상의 관심사항과 애플리케이션의 관심사항간의 명확한 분리가 없다.
- 블록 IO 상의 가정에 기인한 비블록 NIO 의 완전한 이점을 누릴 수 없다.
- 완전한 서블릿 웹 컨테이너는 오히려 많은 애플리케이션들을 지나치게 제한(overkill)한다.

그는 정말로 프로토콜에 독립적이며 컨텐츠와 그것의 메타데이터를 노출하는 컨텐틀릿(contentlet)을 정의해줄 새로운 API 를 제정하자고 제안하였다. 이러한 생각은 Restlet 프로젝트를 만드는데 많은 부분을 시사하고 또 영감을 불어넣었다. 이후 블로그 포스팅에서 그는 왜 컨텐틀릿과 같은 개념을 가지지 않는 현 서블릿 API 가 비블록 NIO API 를 효율적으로 사용하는 것을 제한하는지에 관해 상세하게 설명했다. 이는 전통적으로 처리해야 할 각각의 HTTP 요청을 별도의 쓰레드를 이용하도록 되어있기 때문이다. 그는 더 나아가 이 포스팅에서 차세대 서블릿에 관해서도 설명하고 있다.

또 다른 주요 이슈는 API 가 애플리케이션 개발자들로 하여금 애플리케이션이나 사용자 세션 수준의 세션 상태를 메모리에 직접 저장하도록 조장한다는 점이다. 이것은 좋은 기능처럼 보일지는 모르겠지만, 서블릿 컨테이너의 확장성과 고가용성에 대한 주요 이슈가 되었다. 이러한 문제를 보상하기 위해 복잡한 로드 밸런싱, 세션 복제 및 퍼시스턴스 메커니즘이 구현되어야만 한다. 그렇지만 결국 확장성은 부득이하게 겪게 된다.

Restlet 의 시작

새로운 웹 사이트 개발을 시작할 때, 나는 기술적으로 가능한 만큼 REST 아키텍처 스타일을 따르기를 원했다. 많은 연구를 거쳐 나는 자바 REST 프레임워크가 부족하다는 것을 깨달았다. 그것에 가장 근접한 프로젝트는 1060 Research 에 의해 개발된 1060 NetKernel 뿐이었는데, 나에게 필요한 것에 비해 너무 많은 기능을 가지고 있었고 REST 개념에 대한 매핑이 내가 기대한 것만큼 직접적이지는 않았다.

이러한 연유로 나는 나만의 REST 프레임워크를 서블릿 API 를 기반으로 개발하게 되었다. 그 프레임워크가 어느정도까지 잘 작동했던 것은 서블릿 API 를 완전히 감췄기 때문이었다. 나는

Greg Wilkins 이 제안했던 것을 기억해 냈고 서블릿 API 를 완전히 생략하기로 마음먹었다. 운 좋게도 Jetty 가 자체적인 HTTP 프로토콜 구현과 서블릿 API 을 잘 분리하고 있었다. 결국 나는 직접적으로 REST uniform 호출을 수행하는 최초의 Restlet 커넥터인 HTTP 서버 커넥터를 개발할 수 있었다.

게다가 나는 이 블로그 포스팅의 Benjamin Carlyle 의 좋은 조언에 따라 자바를 이용하여 웹 상의 클라이언트측과 서버측 뷰 사이의 자연스럽지 못한 분리를 없애고 싶었다. 오늘날의 네트워크 환경에서는 그러한 차이점이 만들어지지 말아야 하는데, 누구라도, 동일한 시간에, 웹 클라이언트와 웹 서버로서 액션을 취할 수 있어야 한다. REST 에서 모든 컴포넌트는 필요한 만큼의 클라이언트와 서버 커넥터를 가질 수 있으므로 나는 위에서 언급한 HttpURLConnection 클래스에 기반하여 간단하게 클라이언트 HTTP 커넥터를 개발하였다. 물론 최근에 추가된 Jakarta Commons HTTP Client 에 기반한 것과 같은 다른 구현체들도 제공될 수 있다.

몇 번을 반복하고 나서 Restlet 프로젝트를 두 부분으로 분리하는 것이 개발자에게 더 이로울 것이라는 것이 확실해 졌다. 첫 번째 부분은 Restlet API 라 불리는 일반적인 클래스 집합이며 몇 가지 헬퍼 클래스와 Restlet 구현체를 등록하는 메커니즘을 포함하고 있다. 두 번째 부분은 Noelios Restlet 엔진이라 불리는 참조 구현체로서 몇 가지 서버 및 클라이언트 커넥터(HTTP, JDBC, SMTP 등)와 표현(문자열, 파일, XML 문서, 템플릿, 스트림 등에 기반한) 집합, 그리고 파일 확장자에 기반하여 automatic content negotiation 을 통해 디렉터리 트리로부터 정적 파일들을 처리할 수 있는 디렉터리(Directory)를 포함한다. 2005 년 11 월에 처음 일반에 공개된 이후 프로젝트는 활발한 커뮤니티 사용자와 개발자가 도움을 주는 덕택에 성숙해지고 있다.

결론

복잡하게 집중화된 모델의 강력함에도 불구하고 객체지향적인 패러다임이 웹 개발에 있어 언제나 최적의 선택은 아니다. 자바 개발자들은 이것을 깨닫고 새로운 웹 서버나 새로운 AJAX 기반의 웹 클라이언트를 개발할 때 좀 더 RESTfully 하게 생각하는 것을 시작할 필요가 있다. Restlet 프로젝트는 여러분이 곧바로 웹 2.0 에 올라탈 수 있도록 해주는 단순하지만 견고한 기반을 제공해 준다.