ASP.NET 기술

차재호* · 김정인**

ASP.NET은 ASP의 차세대 버전이며, 웹 Application을 구축하는데 필요한 서비스를 제공하는 웹 개발 플랫폼이다.

1. 문제의 배경

웹이 등장 했을 때, 처음에는 텍스트와 그림을 정적인 페이지로 보여주기 위해 사용되었다. 이 정도의 웹 서버를 작성하는 것은 비교적 쉬운 일이었다. 파일을 가리키는 URL을 받아 들여 서버의 하드 드라이브로부터 해당 파일을 읽어 들이고, 다시 이 파일의 내용을 Client에게 보여주면 되었다. 이런 간단한 구조를 가지고도 그 당시에는 많은 일을 할 수 있었다. 웹에 있는 데이터가 정적이면서 개방되었던 그 당시에는 이러한 방법이 제대로 작동하였다. 이 당시에는 사용자 입력이나 프로그래밍 Logic은 없는 상태였고, 그냥 단순하게 웹 사이트에서 보여주는 내용만을 보는 것이 전부였다.

그러나, 오늘날에는 사용자가 급격히 증가 하면서 웹 페이지와의 상호 작용하려고 하는 경향이 급격히 증가 하였고, 이에 따라 웹 프로그래머는 사용자의 입력을 받아 들여 응답하는 HTML을 동적으로 생성하도록 프로그램을 작성하여야 하며, 이러한 이유로 새로운 접근법을 생각해야 했다.

Client에 따라서 페이지를 동적으로 생성하는 웹 서버 Application에서는 몇 가지의 작업이 필요하다.

첫째: 프로그램을 페이지 요청과 연결 시키는 방법이 필요하다. 사용자가 페이지를 요청할 때, 서버는 단순히 디스크로부터 파일을 읽어 오는 것이 아니며, 해당 페이지는 사용자의 요청이 들어오기 전에는 존재하지도 않는다. 대신 웹 서버는 페이지를 생성하는 프로그램을 실행 시키게 된다.

둘째: 웹 서버는 사용자가 입력한 내용을 서버 쪽 프로그램으로 전달 해 주고, 서버 프로그램의 실행 결과를 사용자에게 다시 전달해 주는 방법이 필요하게 된다.

셋째: 사용자가 요청하는 데이터에는 사적인 것도 있기 때문에 웹 서버는 사용자가 누구인지를 알아내서 해당 사용자가 가지고 있는 권한만큼의 행위를 할 수 있게 해야 한다. 또한 다른 사용자가 볼 수 없도록 해야 한다.

마지막으로 웹 서버는 사용자의 세션 관리 메커니즘이 필요하다. 사용자는 웹 사이트와 상호작용을 개별적인 페이지의 요청으로 보는 것이 아니기 때문에 적정 기간 동안 지속되는 대화의 개념, 즉 세션의 개념이 필요하다. 이런 작업을 하는 것

*㈜필라넷 수석컨설턴트, ㈜웹타임 전임강사
**동명정보대학교 컴퓨터공학과 교수

은 쉽지 않으며, 별도의 코드를 필요로 하게 된다.

웹 서버는 공통적으로 해결해야 하는 문제를 해결할 Runtime 환경이 필요하다. 이러한 환경은 프로그램을 쉽게 작성하게 해주고, 관리 및 배치하기 위해서도 필요하다.

웹이 보편화되고 사용자의 요구가 증가함에 따라 웹 프로그래머는 두 가지 분야에서 더욱 복잡한 작업을 해야만 했다. 그 중 하나는 웹 프로그래밍을 보다 쉽게 해야 하는 것과, 이러한 웹 Application이 보다 잘 수행 되도록 하는 것이었다.

ASP.NET은 이런 두 분야에서 많은 개선을 가져왔다. ASP.NET은 초기 ASP와 비슷하게 생겼고 별도로 포팅하지 않아도 된다. 그러나, 내부적으로는 ASP.NET은 .NET Framework을 활용하기 위해 완전히 개조되었다. 이전의 코드는 HTML 요소와 스크립트가 섞여 있어서 프로그램하기도 어려웠고, 관리하기도 어려웠다.

ASP.NET은 코드 비하인드(Code-behind)라는 특징을 이용해서 프로그램과 HTML 출력을 분리시킨다. 이러한 이유로 코드 가독성이 증대하고 디버깅 환경이 개선 되었다. ASP.NET은 이전에 윈도우 컨트롤을 이용하여 윈도우 Application을 쉽게 작성하였던 것처럼 ASP.NET은 HTML 페이지를 쉽게 작성할 수 있도록 도와주는 컨트롤이 미리 구현되어 있다. 이 컨트롤을 이용해서 HTML의 세부적인 문법을 몰라도 속성을 설정함으로써 Application을 작성할 수 있다

또한 ASP.NET은 보안 프로그램을 작성하는 것이 훨씬 쉬워졌다. ASP.NET은 자신의 Application을 지키기 위한 많은 방법을 제공하고 있으며, 범 세계적인 인증기법으로 Passport를 지원하기도 한다.

ASP.NET은 기존의 ASP에서 세션의 상태관리 기능 중에서 여러 대의 서버에서 할 수 없었던 단점을 확장시켰다.

어떤 의미에서 아주 중요한 내용 중에 하나가 ASP.NET은 .NET Framework에서 돌아가기 때문에 컴포넌트나 페이지를 바꾸기 위해 서버를 중지 시킬 필요가 없다는 것이다. 그뿐만 아니라 프로세스를 재활용할 수도 있도록 했으며, 프로세스가 영원히 수행되도록 할 수도 있고, 서버를 자동으로 중지 시켰다가 설정시간이 되면 다시 자동으로 시작하도록 설정할 수가 있다. 이렇게 하면 사용자 코드에서 발생하는 메모리 누수문제도 해결할 수가 있다.

2. Web Control에 대하여

웹 컨트롤은 오래 전 윈도우 사용자 인터페이스를 만들 때와 같은 생각으로 만들어 졌다. 컨트롤은 사용자 인터페이스를 다루는 프로그램 Logic을 캡슐화하기 위해 존재한다.

입력과 출력이 빈번하게 일어나는 상황인 경우, 입/출력을 프로그래머가 사용할 수 있도록 미리 묶어 두는 것도 매우 좋은 생각이다. 막 프로그램을 시작한 사람이라면 버튼과 같은 입/출력 프로그램을 작성할 필요가 없다는 것이 얼마나 많은 시간을 줄여주는지 금방 알 수가 있을 것이다. 뿐만 아니라 사용자 인터페이스의 일관성을 유지할 수도 있을 것이다.

ASP.NET의 관점에서 웹 컨트롤은

• Visual Studio.NET과 같은 개발환경에 표준 프로그래밍 모델을 보여주며

• 윈도우 컨트롤이 자신의 겉모습을 윈도우 GDI함수로 만들었던 것처럼 표준 브라우저에서 디스플레이 할 수 있도록 HTML을 만들어 낼 수 있는 프로그래밍 Logic과 관련된 사용자 인터페이스의 한 형태이다.

웹 컨트롤은 HTML 컨트롤 보다 더 풍부하고 다양한 기능을 가지고 있어서 보다 세부적으로

구현이 가능하며, 프로그램 작성도 쉽다. 웹 컨트롤은 각각 개별적으로 다른 기능을 수행하며,그 수행엔진은 서버 쪽에서 컨트롤을 만들어 내고 그 컨트롤을 사용한다. 컨트롤은 Client를 위해 자신과 관련된 HTML을 만들어 낸다.

웹 컨트롤 중 입력 컨트롤의 경우, 멋진 기능 중 하나가 바로 PostBack 데이터라고 부르는 것이다. 사용자가 입력을 마친 뒤에 페이지를 떠났다가 다시 돌아오면 이전에 입력했던 내용을 그대로 기억하고 있다가 보여준다.

웹 컨트롤에 들어있는 디스플레이 컨트롤도 view state라고 부르는 기능이 있어서 각 컨트롤의 속성을 이용해서 상태를 유지하는 기능이 있다. 사용자가 이 컨트롤을 특정 값으로 설정하지는 않지만 이전에 페이지를 떠날 때 어떤 상태였는지 기억한다. .aspx 페이지 환경은 각 페이지마다 __VIEWSTATE라는 Hidden 입력 컨트롤을 자동으로 넣어둔다. 페이지가 없어 질 때 자동으로 자신의 상태를 __VIEWSTATE에 차례로 넣어둔다. 그리고 나서 페이지가 다시 생성될 때 저장해 둔 상태를 찾아낸다. 만약 이런 기능을 원하지 않을 경우 해당 컨트롤의 MaintainState 속성을 False로 설정하면 된다.

웹 컨트롤의 뛰어난 기능 중 또 하나는 웹 컨트롤이 사용자가 사용하고 있는 브라우저의 능력을 감지하여 각 브라우저를 충분히 활용할 수 있도록 서로 다른 코드를 만들어 낸다.

마지막으로 웹 컨트롤은 확장 가능하게 되어 있어서 그다지 어려움 없이 자신만의 웹 컨트롤을 만들어 낼 수가 있다.

3. 상태관리

웹 페이지에 대한 요청은 기본적으로 각각 독립적으로 구성 된다. 즉, 사용자가 페이지 A를 방문했다가 몇 분 후에 페이지 B를 방문했을 경우 페이지 B의 내용은 사용자가 페이지 A를 방문했는지 안 했는지 모른다는 것이다. 즉 HTTP는 웹 서버가 연속적으로 들어오는 요청이 동일한 사용자에게서 왔는지 분간할 방법이 본질적으로 없다는 것이다. 웹 서버가 상태를 유지하고 그것을 개별 사용자와 연결시키기가 어렵다는 것이다.

기존 ASP에서 이 제약은 Session 내장 객체의 키-값 쌍으로 되어 있는 In-Memory 컬렉션을 통해서 극복했었다. 그러나 Session 컬렉션 자체가 상당한 한계를 가지고 있었다. 또한 많은 Application은 개별 사용자 상태뿐 아니라 웹 Application의 모든 사용자에게 공유되는 Application 수준의 상태 정보를 저장하고 조회해야 하는 경우가 있는데 기존 ASP에서는 Application 내장 객체에서 Session 컬렉션과 비슷한 컬렉션을 제공했다.

Application 상태란 Application을 사용하는 여러 사용자 간에 공유되어야 하는 Data를 말한다. 데이터베이스의 연결 문자열 정보라든지, 공유변수, Cache DataSet이 좋은 예라고 할 수 있다. Application 상태 관리의 어려움 중 하나가 상태를 저장하는 다양한 방법 중에서 하나를 선택해야 한다는 것이다.

기존 ASP처럼 ASP.NET은 개발자들이 값과 개체 인스턴스를 저장하는 데 사용할 수 있는 키-값의 쌍으로 되어 있는 컬렉션을 제공한다. 기존 ASP와 비슷한 방법으로 해당 컬렉션에 접근할 수 있다. ASP.NET에서 Application 상태 저장 기능은 HttpApplicationState 클래스의 인스턴스에 의해 제공되며, 해당 클래스의 인스턴스는 Page 클래스의 Application 속성으로 노출된다. 모든 ASP.NET 페이지는 Page 클래스에서 상속하기 때문에 Application 속성을 해당 Page의 속성인 것처럼 접근할 수 있다.

여러 사용자가 공유하는 상태 정보를 관리하는 데 있어서 어려운 점은 두 사용자가 동시에 상태 정보를 수정하여 잘못된 데이터가 저장되는 일이 일어나지 않게 보장 하는 것이다. 기존 ASP에서 처럼 ASP.NET에서 Application 객체는 Lock과 Unlock 메소드를 제공해서 개발자들이 오직 한 사용자만이 지정된 시간에 Application 상태 정보를 수정할 수 있게 한다.

Application 상태에는 상태 정보를 관리하는 방법을 정할 때 고려해야 하는 몇 가지 제약 조건이 있다

첫째: 지속성

Application상태는 웹 Application이 가동하는 동안만 존재한다. 웹 Application 혹은 웹 서버가 종료되거나 충돌하면 Application 수준의 저장된 모든 상태 정보는 소멸된다. Application이 재 시작까지 유지되어야 할 상태 정보는 데이터베이스나 다른 영속적인 저장소에 저장해야 한다.

둘째: 웹 농장(Web Farm)

Application 상태는 웹 농장의 여러 서버 간에 공유되지 못한다. 이런 시나리오에서 모든 사용자가 이용할 수 있는 값을 저장해야 한다면, Application 상태는 적절한 선택이 아니다.

셋째: 메모리

서버에 이용 가능한 물리적 메모리 양은 항상 한정되기 마련이다. Application 상태를 과도하게 사용하면 상태 정보가 가상 메모리에 저장되는 결과가 초래되어 성능을 심각하게 감소시킬 수 있다.

Application 상태 정보를 관리하는 것도 힘들지만, 상태 정보가 지속되지 않는 Stateless HTTP에서 각 사용자의 상태를 유지하기는 훨씬 더 힘들다. 웹 서버는 일련의 요청이 같은 사용자에 대한 상태 정보인지 알아내야 하고, 웹 서버는 이 요청을 해당 특정 사용자에 대한 상태 정보와 연결해야 한다.

기존 ASP에서는 이러한 요구를 Session 내장 객체를 사용하였다. 각 사용자는 세션 ID에 의해 식별되는 자기만의 세션 인스턴스를 가지고 있었고, 여기에 사용자 각각의 상태 정보를 저장할 수 있었다. 세션 ID는 Application이 가동되는 동안 고유하게 생성된 값으로 Client 컴퓨터의 쿠키에 저장되었다. 그 다음 사용자의 연속된 요청마다 해당 ID값이 다시 전달되어 서버가 그 사용자의 세션을 파악할 수 있게 했다.

그러나, 기존 ASP의 세션 상태는 몇 가지의 본질적인 한계를 가지고 있었다. 특히 높은 확장성을 요구하는 Application에게 이상적인 방법은 아니었다.

그 한계를 알아보면 다음과 같다.

• 웹 농장(Web Farm)

기존 ASP의 세션 상태는 앞에서 언급했듯이 웹 농장의 서버들간에 확장되어질 수 없다.

• 지속성

기존 ASP의 세션은 서버가 재 시작하거나, 충돌이 발생하면 소멸된다. 장바구니 같은 경우 이러한 한계 때문에 해당 방법은 적절하지 않다.

• 쿠키 의존성

기존 ASP는 쿠키를 받아 들일 수 없는, 또는 받아 들이지 않으려고 하는 브라우저에서 세션 상태를 지원할 방법이 없었다. 다른 외부 솔류션을 이용하면 가능하지만 이러한 경우 성능 면에서 수용하기 어려운 결과가 종종 있었다.

ASP.NET은 쿠키를 지원하지 않는 브라우저에서도 확장 가능하고 신뢰할 수 있는 가용성 있는 상태 저장소를 사용자 마다 제공하며, 이러한 한계를 해결한다.

ASP.NET은 세션 상태를 저장할 수 있는 새로운 설정 몇 가지를 제공한다.

이용 가능한 설정 값을 보면

• In-Process(InProc)

이것은 기본 설정 값이다. 기존 ASP와 동일하게 수행된다.

◦ Out-of-Process(StateServer)

이 설정은 세션 상태가 ASP.NET state NT 서비스를 가동하는 서버에 의해 저장되도록 한다. 연결된 State Server를 Attribute에 지정하면 된다.

◦ SQL Server

세션 상태를 SQL Server에 저장한다. SQL Server를 Attribute에 지정하면 된다.

• Cookieless Sessions

이 설정은 쿠키를 지원하지 않는 브라우저의 사용자들에 대해서 조차도 세션 상태를 유지할 수 있도록 해준다.

다음으로 일시적으로 나마 상태를 저장하기 위한 방법으로 Cache를 사용할 수 있다. Cache의 설계 목표는 개발자에게 Application의 이점을 제공하면서 파일이 변경된 경우 Cache에서 Item을 복구하는 등의 Application이 제공하는 것 이상의 추가 기능을 제공하려는 것이다. 원한다면 Item을 다시 Cache에 추가하는 코드를 통해 사용할 수 있다.

Cache는 System.Web.Cache 네임스페이스에서 찾을 수 있는 Cache 클래스의 인스턴스이다. 이것은 Application과 같은 간단한 키/값 쌍 Hashtable이며 아래의 내용을 지원한다.

◦ 의존성 기반 만료

의존성은 또 다른 Cache 키, 파일, 타임스탬프가 될 수 있다. 이 의존성 중 하나가 변경되거나 만료되면 Cache Item은 무효가 되고 Cache에서 삭제된다.

◦ 잠금관리

Application과 유사하게, 동시에 발생하는 요구가 Cache를 수정하려고 할 수 있다. Application은 Lock()과 Unlock()메소드를 사용하지만, Cache 클래스는 Application과는 달리 자신만의 내부 잠금 관리 기능을 사용한다. 따라서 Application은 Update할 때 Lock()과 Unlock()메소드를 사용하도록 요구하는 반면 Cache는 그럴 필요가 없다.

◦ 리소스 관리

사용되고 있는 Cache Item을 자동으로 제거하고 가용 메모리를 늘인다. 따라서 Item을 요청하기 전에 항상 해당 Item이 존재하는지 검사할 필요가 있다.

◦ CallBack

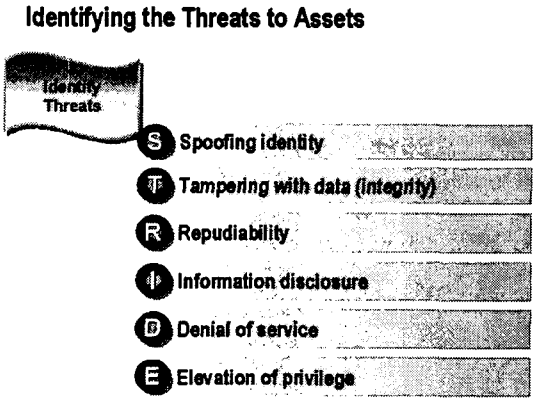
Cache는 Item이 Cache에서 제거될 때 코드를 실행시킬 수 있게 하는 기능을 지원한다.

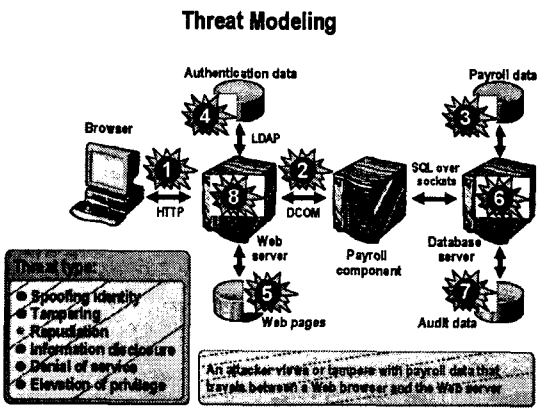
4. 보 안

보안은 웹 개발자가 Application을 설계하고 구현할 때 고려해야 할 첫째 사안 중 하나이다. 여러 가지 측면에서 보안을 고려하지 않고 Application을 설계하는 것은 보안이 없는 Application을 설계하는 것과 마찬가지이다.

보안의 수준과 유형은 다양하다. 어떤 보안 유형과 수준을 적용해야 하는가는 해당 Application이 하는 일과 저장될 데이터의 유형과 비용, 수용할 수 있는 위험의 크기, 안전한 Application을 만드는데 드는 시간과 노력, 비용에 따라 다르다고 볼 수 있다. 개인의 홈페이지와 기업의 인트라넷이나 전자 상거래 사이트의 보안 요구 사항은 많이 다를 것이다. 다음은 실제 공격의 유형을 살펴본 것이다.

웹 Application을 만드는데 이렇게 많은 잠재적인 공격을 생각하면 공격에 안전한 Application 설계를 어디서부터 시작해야 할 지 안다는 것 자체가 어려운 일이다. 여기서 몇 가지 전략을 살펴보면





• 서버 설치와 Application 설계

공격에 안전하지 않은 서버 설정과 잘못된 Application 설계 때문에 발생하는 웹 서버 및 데이터 손상 방지하기

• 패치하기

서버에 있는 소프트웨어의 취약점으로 인한 웹 서버 손상 방지하기

• 엑세스 제한

부적절한 엑세스 설정으로 인한 웹 서버 및 데이터 손상 방지하기

• 감사와 로그 남기기

누가 사이트를 방문하고 있는지, 무엇을 하는지, 언제인지 추적하기

• SSL과 기타 암호화 보안 도구

데이터 손상 막기

이러한 전략을 통해 Application의 Logic 상에서 어떤 종류의 보안의 위협이 존재하는지 파악하여 적절한 정책을 적용하여야 한다. 예를 들면 웹 Application의 Logic 상에서 어떤 부분들에서 보안 위협을 고려해야 하는지를 아래의 그림과 같이 보여 줄 수 있다.

각각의 프로세스에서 STRIDE 보안 위협 중 어떠한 위협이 존재하는지 파악하여 적절한 정책을 수립하게 된다. 이러한 보안 위협을 제거하기 위한 정책을 수립하는 과정에서 몇 가지 보안을 구현하는 기술적인 방법을 살펴봐야 할 것이다.

우선 기본적인 개념의 파악이 따라야 할 것이다.

Application 보안의 기본 개념은 네 가지의 주제로 구성된다.

• 인증(Authentication)

• 권한부여(Authorization)

• 인격화(Impersonation)

• Data 및 기능적 보안은 물리적인 방법, 운영체제 업데이트, 안정적인 소프트웨어 사용 등을 통해 시스템을 보안하는 과정

위에서 열거한 주제 중에서 앞의 세 가지 주제를 구현하는데 필요한 기능을 제공하기 위해 운영체제, IIS, .NET Framework의 여러 요소들이 결합될 수 있다. 전체 보안 과정이 어떻게 동작하는지 알기 위해서 세 가지 주제를 각각 살펴보고자 한다.

첫째: 인증(Authentication) :

웹 Application 혹은 특정 폴더, 자원, 컴포넌트에 대한 액세스를 특정 사용자에게 제한하려면 사용자를 구별할 수 있어야 한다. 아주 세부적인 사항까지 구별하지는 못하지만 최소한 액세스 여부를 판단할 수 있을 정도는 알아야 한다. 인증은 사용자가 자기 자신임을 입증하도록 요청하는 것

을 말한다.

둘째: 권한부여(Authorization)

일단 사용자가 누구인지 알게 되면 사용자가 요청한 자원에 대해 액세스하도록 할지 결정한다. 자원의 종류에 따라 몇 가지 방법으로 나누어 진다. 윈도우 기반에서는 ACL(Access Control List)를 가지고 있는데, 이는 해당 자원에 액세스 할 사용자의 목록이다. 이 목록으로 각 사용자 별로 액세스의 종류를 결정할 수도 있다.

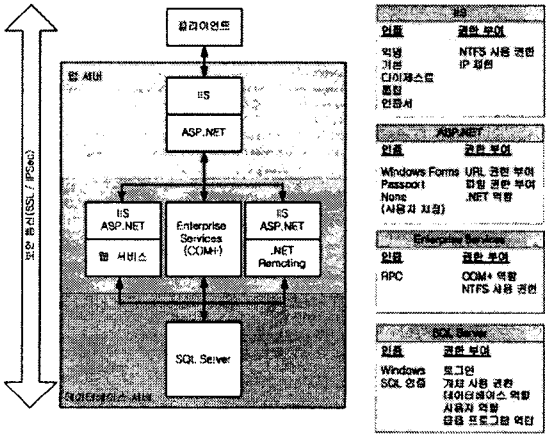
셋째: 인격화(Impersonation)

윈도우는 기본적으로 익명 액세스를 허용하지 않는다. 모든 사용자가 자신의 계정으로 인증되고 권한을 부여 받게 된다. IIS는 어떤 페이지나 자원에 대해 익명으로 액세스 하도록 요청을 받았을 때 IIS는 IUSR_서버이름 계정을 사용하여 익명 사용자를 대표한다. 만약 요청 받은 자원이 COM 또는 COM+ 컴포넌트를 사용하는 ASP 페이지라면 그 컴포넌트는 IWAM_서버이름 이라는 계정으로 실행 된다. 그러나, ASP.NET 페이지라면 자원에 대한 모든 액세스를 로컬 시스템 계정을 사용해서 실행되어 질 것이다. 만약 인격화가 작동하고 있다면 ASP.NET은 인증된 사용자 계정으로 모든 자원을 실행하게 된다.

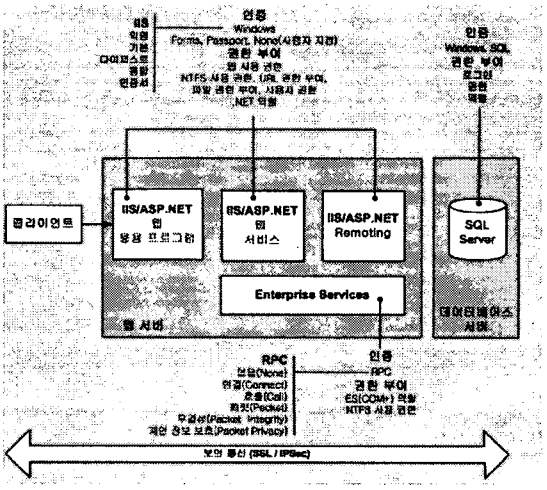
ASP.NET에서의 보안

기존의 ASP에서 IIS에서 보안에 관련된 설정을 했던 것은 아직도 어떤 경우에는 효과적이다. Application 설정이나 자주 발생하는 에러 등의 설정 등에서 IIS는 아직도 운영체제와 함께 기본적인 보안 절차를 수행하고 요청을 관리하기 때문이다. 사실 아직도 ASP.NET 페이지에 대한 요청은 IIS가 받아서 해당 페이지가 포함된 사이트에 대한 설정이 정의 되어 있는 Application 매핑을 사용해서 ASP.NET에 전달한다. IIS에서 사이트나 디렉토리의 등록정보 대화상자로 홈 디렉토리 페이지에 있는 응용프로그램 구성 대화상자를 열면 그 매핑을 볼 수 있다. 모든 ASP.NET 자원의 유형별 스크립트 매핑은 .NET Framework 폴더에 저장되어 있는 aspnet_isapi.dll이라는 파일로 향해 있다. Application 매핑은 아직 파일의 확장자에 의존한다. 이것이 ASP.NET이 작동하고 있는 서버에서 기존의 ASP 3.0 페이지들이 계속 실행될 수 있는 이유이다.

IIS는 먼저 사용자를 인증하고, 보안 프로세스를 실행하는 동안 ASP.NET으로 그 응답을 넘긴다. 여기서 모든 보안 사항을 설명할 수는 없기



.NET 웹 응용프로그램 보안



때문에 전체적인 보안 아키텍처를 살펴보고자 한다. 응용 프로그램 서버와 원격 응용 프로그램 서버의 경우로 나누어 볼 수도 있지만 기본적으로 Internet 응용프로그램은 원격 응용프로그램의 더 복잡한 형태로 보고 원격 응용 프로그램의 아키텍처를 살펴 볼 것이다.

위의 그림은 원격 응용 프로그램 계층 모델과 다양한 기술이 제공하는 보안 서비스를 보여 준다. 인증과 권한 부여는 전 계층의 많은 개별 지점에서 발생한다. 이러한 서비스는 주로 IIS, ASP.NET, Enterprise Services 및 SQL Server에서 제공된다. 보안 통신 채널도 계층 전체에서 적용되며 클라이언트 브라우저나 장치에서 데이터베이스까지 연결되어 있다. 채널의 보안은 SSL (Secure Sockets Layer)과 IPSec을 함께 사용하여 이루어진다.

이러한 다양한 계층 모델과 다양한 보안 기술이 제공하는 인증, 권한 부여 및 보안 통신 기능을 요약하면 다음과 같다.

위의 그림에서 볼 수 있듯이 복잡한 과정을 거쳐서 보안 설정이 이루어진다.

처음 인증에 관하여 Windows 2000의 .NET

보안 기능

기 술	인 증	권 한 부 여	보 안 통 신
IIS	익명 인증 기본 인증 다이제스트 인증 Windows 통합 인증 (Kerberos/NTLM) 인증서 인증	IP/DNS 주소 제한 웹 사용 권한 NTFS 사용 권한, 요청된 파일의 Windows 액세스 제어 목록(ACL)	SSL
ASP.NET	None(사용자 지정) Windows 인증 Forms 인증 Passport 인증	파일 권한 부여 URL 권한 부여 사용자 권한 .NET 역할	
웹 서비스	Windows 인증 None(사용자 지정) 메시지 수준 인증	파일 권한 부여 URL 권한 부여 사용자 권한 .NET 역할	SSL 및 메시지 수준 암호화
원격 서비스	Windows 인증	파일 권한 부여 URL 권한 부여 사용자 권한 .NET 역할	SSL 및 메시지 수준 암호화
Enterprise Services	Windows 인증	Enterprise Services(COM+) 역할 NTFS 사용 권한	원격 프로시저 호출(RPC)
SQL Server 2000	Windows 인증 (Kerberos/NTLM) SQL 인증	서버 로그인 데이터베이스 로그인 고정 데이터베이스 역할 사용자 정의 역할 응용 프로그램 역할 개체 사용 권한	SSL
Windows 2000	Kerberos NTLM	Windows ACLs	IPSec

Framework에서 제공하는 인증 옵션을 살펴보자. 크게 세 가지의 인증 관련 모드를 살펴 볼 수가 있다.

첫째: ASP.NET 인증모드
둘째: Enterprise Services 인증
셋째: SQL Server 인증

ASP.NET 인증 모드

ASP.NET 인증 모드에는 Windows, Forms, Passport 및 None이 있다.

Windows 인증.

이 인증 모드에서 ASP.NET은 IIS를 통해 사용자를 인증하고 인증된 ID를 나타내는 Windows 액세스 토큰을 만든다. IIS에서는 다음 인증 메커니즘을 제공한다. :

기본 인증.

기본 인증에서 사용자는 신분을 증명하기 위해 사용자 이름과 암호의 형식으로 자격 증명을 제공해야 한다. 기본 인증은 RFC 2617을 기반으로 제안된 인터넷 표준이다. Netscape Navigator와 Microsoft Internet Explorer는 모두 기본 인증을 지원한다. 사용자의 자격 증명은 암호화되지 않은 Base64로 인코드 되어 브라우저에서 웹 서버로 전송된다. 웹 서버는 암호화되지 않은 사용자 자격 증명을 받기 때문에 사용자의 자격 증명을 사용하여 원격 호출(예: 원격 컴퓨터 및 리소스에 액세스)을 할 수 있다. 기본 인증은 일반적으로 SSL을 사용하여 설정되는 보안 채널과 함께 사용되어야만 한다. 그렇지 않으면 네트워크 모니터링 소프트웨어를 사용하여 사용자 이름과 암호를 쉽게 도난 당할 수 있다. 기본 인증을 사용하는 경우 자격 증명이 로그 온 이후의 모든 요청에서 전달되므로 로그 온 페이지뿐 아니라 모든 페이지에서 SSL을 사용해야 한다.

다이제스트 인증.

IIS 5.0에서 도입된 다이제스트 인증은 기본 인증과 비슷하지만 암호화되지 않은 사용자의 자격 증명을 브라우저에서 웹 서버로 전송하는 대신 자격 증명의 해시를 전송하므로 기본 인증보다 안전하다. 단, 다이제스트 인증에는 Internet Explorer 5.0 이상의 클라이언트 및 특정 서버 구성이 필요하다.

통합된 Windows 인증.

통합된 Windows 인증(클라이언트와 서버 구성에 따라 Kerberos 또는 NTLM)에서는 사용자의 Internet Explorer 웹 브라우저와 암호화 교환을 사용하여 사용자 ID를 확인한다. 통합된 Windows 인증은 Internet Explorer에서만 지원하며 Netscape Navigator에서는 지원하지 않는다. 따라서 클라이언트 소프트웨어를 제어할 수 있는 인트라넷 시나리오에서만 사용되는 경우가 많다. 익명 액세스를 사용하지 않거나 익명 액세스가 Windows 파일 시스템 권한을 통해 거부된 경우 웹 서버에서만 통합된 Windows 인증이 사용된다.

인증서 인증.

인증서 인증에서는 클라이언트 인증서를 사용하여 사용자를 명확히 식별한다. 클라이언트 인증서는 사용자의 브라우저 또는 클라이언트 응용 프로그램에서 웹 서버로 전달된다. 웹 서비스의 경우 웹 서비스 클라이언트가 HttpWebRequest 개체의 ClientCertificates 속성을 사용하여 인증서를 전달한다. 인증서를 받은 웹 서버는 인증서에서 사용자의 ID를 추출한다. 이 방법은 사용자 컴퓨터에 설치된 클라이언트 인증서를 이용하므로 인트라넷이나 사용자 수가 확실히 파악되고 제어되는 익스트라넷에서 주로 사용된다. 클라이언트 인증서를 받은 IIS는 인증서를 Windows 계

정으로 매핑한다.

익명 인증.

클라이언트를 인증할 필요가 없거나 사용자 지정 인증 스키마를 구현하는 경우 익명 인증을 사용하도록 IIS를 구성할 수 있다. 이 경우 웹 서버는 Windows 액세스 토큰을 만들어 모든 익명 사용자를 동일한 익명(또는 게스트) 계정으로 나타낸다. 기본 익명 계정은 IUSR_서버 명이며, 여기에서 서버 명은 설치할 때 지정한 컴퓨터의 NetBIOS 이름이다.

Passport 인증.

이 인증 모드에서 ASP.NET은 Microsoft Passport의 중앙 집중화된 인증 서비스를 사용한다. ASP.NET에서는 웹 서버에 설치되어야 하는 Microsoft Passport 소프트웨어 개발 키트(SDK)에서 제공하는 기능과 관련된 편리한 래퍼를 제공한다.

Forms 인증.

이 방법은 클라이언트측 리디렉션을 사용하여 사용자의 자격 증명(일반적으로 사용자 이름과 암호)을 입력할 수 있는 지정된 HTML 폼에 인증되지 않은 사용자를 전달한다. 이러한 자격 증명은 유효성이 검사되고 난 후 인증 티켓이 생성되어 클라이언트로 반환된다. 인증 티켓은 사용자 ID를 유지하고 사용자가 구성원인 역할의 목록을 해당 사용자 세션 동안 선택적으로 유지한다. Forms 인증은 웹 사이트 개인화에만 사용되는 경우도 있다. 이 경우 ASP.NET이 단순한 구성을 사용하여 자동으로 프로세스를 상당 부분 처리하기 때문에 사용자 지정 코드는 거의 작성할 필요가 없다. 개인화 시나리오에서 쿠키는 사용자 이름만 갖고 있어야 한다. Forms 인증에서는 사용자 이름과 암호를 웹 서버에 일반 텍스트로 보낸다. 따라서 Forms 인증은 SSL로 보안되는 채널과 함께 사용해야 한다. 이후 요청에서 전송되는 인증 쿠키를 계속 보호하려면 로그 온 페이지뿐 아니라 응용 프로그램의 모든 페이지에서 SSL을 사용하는 것을 고려해야 한다.

None.

None은 사용자를 인증하지 않겠다거나 사용자 지정 인증 프로토콜을 사용하고 있다는 것을 나타낸다.

Enterprise Services 인증

Enterprise Services 인증은 기본 원격 프로시저 호출(RPC) 전송 인프라를 사용하여 수행되며 이 인프라에서는 운영 체제의 SSPI(Security Service Provider Interface)를 사용한다. Enterprise Services 응용 프로그램의 클라이언트는 Kerberos 또는 NTLM 인증을 사용하여 인증될 수 있다.

서비스되는 구성 요소는 라이브러리 응용 프로그램이나 서버 응용 프로그램에 호스팅될 수 있다. 라이브러리 응용 프로그램은 클라이언트 프로세스에 호스팅되므로 클라이언트의 ID를 갖게 된다. 서버 응용 프로그램은 고유 ID로 별도의 서버 프로세스에서 실행된다.

서비스된 구성 요소로 들어오는 호출은 다음 수준으로 인증될 수 있다. :

• **기본값(Default)** : 보안 패키지에 대한 기본 인증 수준이 사용된다.

• **없음(None)** : 인증을 하지 않는다.

• **연결(Connect)** : 연결이 이루어질 때만 인증한다. 인증방법은 사용자 ID를 확인하며, Client와 Server의 최초 연결을 인증 대상으로 한다.

• **호출(Call)** : 원격 프로시저 호출이 시작될 때마다 인증한다. 인증방법은 사용자 ID와 일련번호의 암호화를 통해서 이루어지며, 모든 메소드 호출을 인증 대상으로 한다. 인증되지 않은 사용

자와 Network 전송의 기록과 재생을 방어 대상으로 한다.

• **패킷(Packet)** : 모든 호출 데이터를 받았는지를 인증하고 확인한다. 인증방법은 사용자 ID 확인과 일련번호의 암호화를 통해서 이루어 지며, 모든 메소드의 호출과 모든 패킷에 대해 이루어지고, Call과 같은 방어 대상을 가진다.

• **패킷 무결성(Integrity)** : 데이터가 전송 중에 변경되지 않았는지를 인증하고 확인한다. 인증방법은 사용자 ID와 일련번호의 암호화 및 Data CheckSum을 암호화해서 이루어 지며 모든 패킷을 대상으로 하고 인증되지 않은 사용자, Network 전송의 기록과 재생 및 Data Tempering(데이터 조작)에 대해서 방어 대상으로 한다.

• **패킷 개인 정보 보호(Privacy)** : 데이터와 보낸 사람의 ID 및 서명을 포함하여 패킷을 인증하고 암호화한다. 상기의 수준을 모두 포함하며 가장 강력한 방법으로 모든 Data를 암호화 하며, 도청까지 방어 대상으로 한다.

SQL Server 인증

SQL Server에서는 Windows 인증(NTLM 또는 Kerberos)을 사용하여 사용자를 인증하거나 SQL 인증이라고 하는 고유한 기본 제공 인증 스키마를 사용할 수 있다. 사용할 수 있는 옵션에는 두 가지가 있다. :

• **SQL Server 및 Windows.**

• 클라이언트는 SQL Server 인증이나 Windows 인증을 사용하여 Microsoft SQL Server의 인스턴스에 연결할 수 있다. 이것은 혼합 모드 인증이라고도 한다.

• **Windows만.**

• 사용자는 Windows 인증을 사용하여 Microsoft SQL Server의 인스턴스에 연결해야 한다.

두 번째로 권한 부여 에 관하여 Windows 2000의 .NET Framework에서 제공하는 권한 부여 옵션을 살펴보자. 크게 세 가지의 권한 부여 관련 모드를 살펴 볼 수가 있다.

첫째: ASP.NET 권한 부여 옵션

둘째: Enterprise Services 권한 부여 옵션

셋째: SQL Server 권한 부여 옵션

ASP.NET 권한 부여 옵션

ASP.NET 권한 부여 옵션은 ASP.NET 웹 응용 프로그램, 웹 서비스 및 원격 구성 요소에서 사용될 수 있다. ASP.NET에서는 다음과 같은 권한 부여 옵션을 제공한다. :

• **URL 권한 부여.**

• 이것은 시스템과 응용 프로그램 구성 파일 내의 설정으로 구성되는 권한 부여 메커니즘이다. URL 권한 부여를 통해 응용 프로그램의 URI (Uniform Resource Identifier) 네임스페이스에 있는 특정 파일과 폴더에 대한 액세스를 제한할 수 있다. 예를 들어, 지정된 사용자에 대해 URL로 표시되는 특정 파일 또는 폴더의 액세스 권한을 선택적으로 거부하거나 허용할 수 있다. 또한 요청을 발행하는 데 사용되는 HTTP 동사의 유형 (GET, POST 등)과 사용자의 역할 구성원 자격을 기반으로 액세스를 제한할 수도 있다. URL 권한 부여에는 인증된 ID가 필요하다. 인증된 ID는 Windows 인증이나 티켓 기반 인증 스키마를 통해 얻을 수 있다.

• **파일 권한 부여.**

• 파일 권한 부여는 IIS에서 제공되는 Windows 인증 메커니즘 중 하나를 사용하여 호출자를 인증하고 ASP.NET이 Windows 인증을 사용하도록 구성되어 있는 경우에만 적용된다.

파일 권한 부여를 사용하여 웹 서버에서 특정 파일에 대한 액세스를 제한할 수 있다. 액세스 권한은 파일에 연결된 Windows ACL에 의해 결정

된다.

• **사용자 권한 요구.**

• 사용자 권한 요구는 선언적 방식이나 프로그래밍 방식을 통해 추가적인 고급 액세스 제어 메커니즘으로 사용될 수 있다. 사용자 권한 요구를 사용하면 개별 사용자의 ID와 그룹 구성원 자격을 기반으로 클래스, 메서드 또는 개별 코드 블록에 대한 액세스를 제어할 수 있다.

• **.NET 역할.**

• .NET 역할은 응용 프로그램에서 같은 사용 권한을 가진 여러 사용자를 그룹화하는 데 사용된다. .NET 역할은 이전의 역할 기반 구현인 Windows 그룹과 COM+ 역할 등과 개념적으로 유사하다. 그러나 이전 방법과 달리 .NET 역할은 인증된 Windows ID를 필요로 하지 않으며 Forms 인증과 같은 티켓 기반 인증 스키마와 함께 사용할 수 있다.

.NET 역할은 리소스와 작업에 대한 액세스를 제어하는 데 사용할 수 있으며 선언적 방식과 프로그래밍 방식으로 구성할 수 있다.

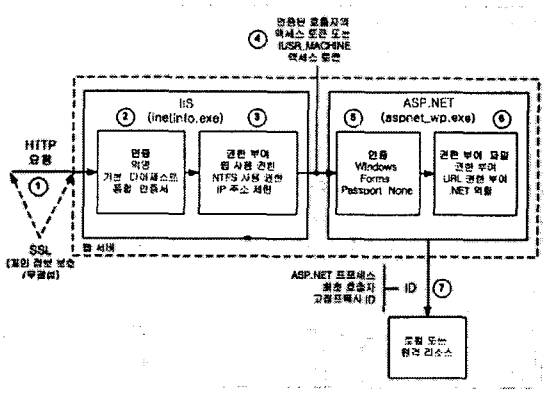
Enterprise Services 권한 부여

Enterprise Services 응용 프로그램에 있는 서비스되는 구성 요소의 기능에 대한 액세스는 Enterprise Services 역할 구성원 자격에 의해 관리된다. 이러한 역할 구성원 자격은 .NET 역할과는 다르며 Windows 그룹이나 사용자 계정을 포함할 수 있다. 역할 구성원 자격은 COM+ 카탈로그에서 정의되며 구성 요소 서비스 도구를 사용하여 관리된다.

SQL Server 권한 부여

SQL Server에서는 개별 데이터베이스 개체에 적용할 수 있는 세부적인 사용 권한을 제공한다. 사용 권한은 역할 구성원 자격(SQL Server에서는 고정 데이터베이스 역할, 사용자 정의 역할 및 응용 프로그램 역할을 제공)을 기반으로 부여되거나 개별 Windows 사용자나 그룹 계정에 부여될 수 있다.

이제까지 인증과 권한부여에 관하여 살펴 보았다. 이를 정리해서 ASP.NET의 보안 아키텍처를 구성해서 보면 다음과 같은 그림을 그릴 수 있다.



IIS 및 ASP.NET이 제공하는 인증 및 권한 부여 메커니즘을 보여 준다. 클라이언트가 웹 요청을 발행하면 다음과 같은 순서로 인증 및 권한 부여 이벤트가 발생한다. :

1. 네트워크에서 HTTP(S) 웹 요청을 수신하고. SSL을 사용하여 서버 ID(서버 인증서 사용)와 클라이언트 ID(선택적)를 보안할 수 있다.

2. IIS가 기본, 다이제스트, 통합(NTLM 또는 Kerberos) 또는 인증서 인증을 사용하여 호출자를 인증하면 사이트 전체 또는 일부에서 인증된 액세스가 필요하지 않은 경우 익명 인증을 사용하도록 IIS를 구성할 수 있다. IIS는 인증된 각 사용자에 대한 Windows 액세스 토큰을 만든다. 익명 인증을 선택하면 IIS는 익명 인터넷 사용자 계정(기본적으로 IUSR_MACHINE)에 대한 액세스 토큰을 만든다.

3. IIS가 요청된 리소스를 액세스하는 권한을 호출자에게 부여하고 요청된 리소스에 첨부된

ACL에 의해 정의되는 NTFS 사용 권한 액세스 권한을 부여하는 데 사용된다. 특정 IP 주소를 가진 클라이언트 컴퓨터의 요청만 받도록 IIS를 구성할 수도 있다.

4. IIS가 인증된 호출자의 Windows 액세스 토큰을 ASP.NET에 전달하고 익명 인증을 사용할 경우에는 익명 인터넷 사용자의 액세스 토큰이 전달될 수 있다.

5. ASP.NET이 호출자를 인증한다.

Windows 인증을 사용하도록 ASP.NET을 구성한 경우 이 부분에서 추가 인증이 수행되지 않는다. 대신 ASP.NET은 IIS로부터 받은 모든 토큰을 수락한다.

Forms 인증을 사용하도록 ASP.NET을 구성한 경우 호출자가 HTML 폼을 사용하여 제공한 자격 증명이 데이터 저장소(일반적으로 Microsoft SQL Server 데이터베이스 또는 Active Directory 디렉터리 서비스)에 대해 인증된다. Passport 인증을 사용하도록 ASP.NET을 구성한 경우에는 사용자가 Passport 사이트로 리디렉션된 다음 Passport 인증 서비스가 사용자를 인증한다.

6. ASP.NET이 요청된 리소스나 작업에 대한 액세스 권한을 부여한다.

또한 .NET 역할을 선언적 방식 또는 프로그래밍 방식으로 사용하여 요청된 리소스에 액세스하거나 요청된 작업을 수행하는 권한 호출자에게 부여되었는지 확인할 수 있다.

7. 응용 프로그램 내의 코드가 특정 ID를 사용하여 로컬 및/또는 원격 리소스를 액세스하고 기본적으로 ASP.NET은 가장을 사용하지 않으므로 구성된 ASP.NET 프로세스 계정이 ID를 제공하게 된다. 이외에도 최초 호출자의 ID(가장을 사용하도록 설정한 경우)나 구성된 서비스 ID가 대체 옵션이 될 수 있다.

지금까지 인증과 권한 부여에 관하여 알아 보았다. 보안은 상당히 중요한 부분이기 때문에 비중을 조금 두었다. 이외에도 많은 고려 사항이 있지만 중략하고자 한다.

차 재 호

◦ 1993년 울산대학교 행정학과 학사 졸업
◦ 전 MBC, 경실련, 매경디지탈캠퍼스 전임강사
◦ 현재 : ㈜필라넷 수석컨설턴트, ㈜웹타임 전임강사
◦ 자격사항
Microsoft Certified Trainer (MCT)
Microsoft Certified System Engineer (MCSE)
Microsoft Certified Solution Developer (MCSD)
Microsoft Certified Database Administrator (MCDBA)
Microsoft Certified Application Developer (MCAD)

김 정 인

• 1986년 2월 계명대학교 통계학과 이학사
• 1993년 3월 게이오대학 계산기과학전공 공학석사
• 1996년 3월 게이오대학 계산기과학전공 공학박사
• 1986년 3월~1988년 11월 (주)국제컴퓨터엔지니어링 개발부
• 1988년 12월~1990년 2년 (주)미래정보시스템 개발부 과장
• 1996년 5월~1998년 2월 포항공과대학교 정보통신연구소 연구원
• 1998년 3월~현재 동명정보대학교 컴퓨터공학과 교수
• 관심분야 : 자연언어처리, 정보검색, 기계번역