22 데이터베이스

모던 웹을 위한 Javascript jQuery 입문

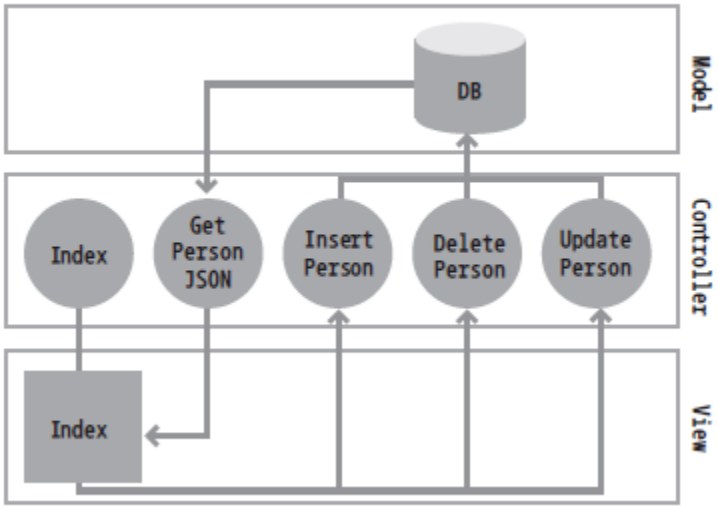
한빛미디어 Hanbit Media, Inc.

0. 프로젝트 구성

❖ 데이터 베이스 활용

- 뷰에서 직접 서버의 데이터베이스에 접근할 수 없음
- 웹 서비스를 사용해 데이터베이스의 내용을 가져오고, 내용을 추가, 삭제하고, 수정

그림 22-1 프로젝트 구성

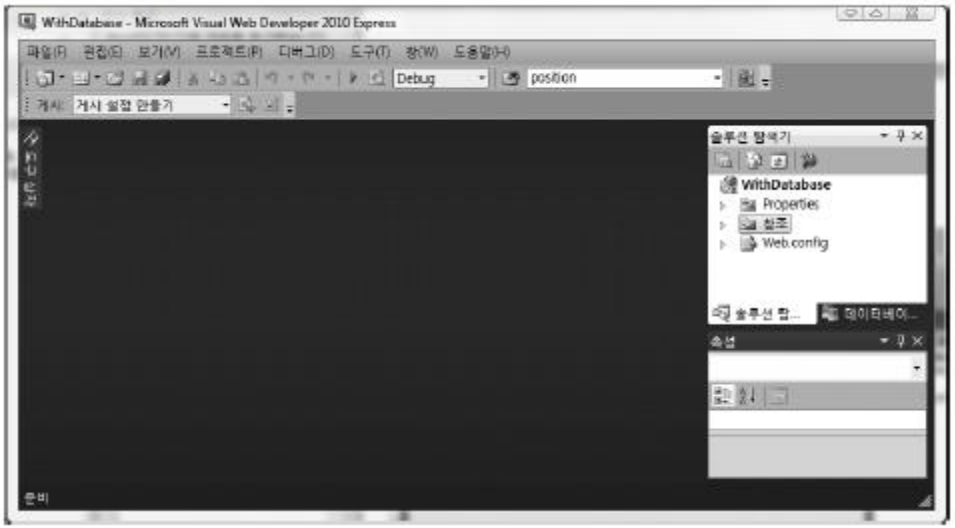


22.1 데이터베이스 만들기

❖ 데이터베이스를 생성하는 방법

- ASP.NET MVC 3 프로젝트 새로 생성
  - 프로젝트의 제목 - 'WithDatabase'

그림 22-2 프로젝트 생성



22.1 데이터베이스 만들기

❖ 데이터베이스를 추가하는 방법

- [추가]→[새 항목] 클릭
  - 'SQL Server 데이터베이스' 선택
  - 파일의 이름은 'Database.mdf'로 설정

그림 22-3 새 항목 추가

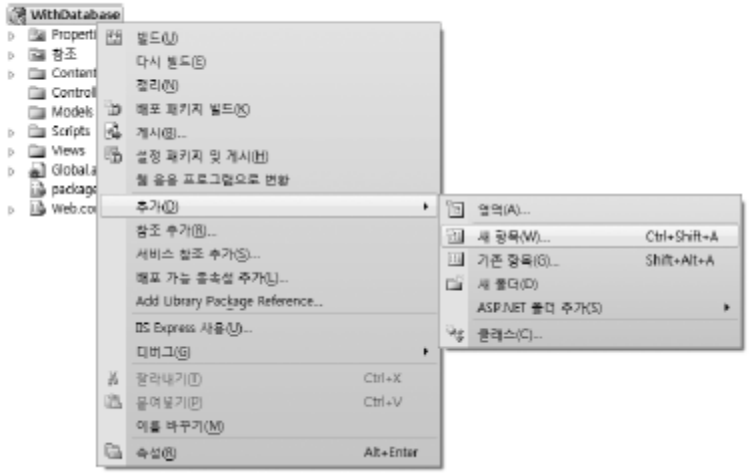


그림 22-4 SQL Server 데이터베이스 항목

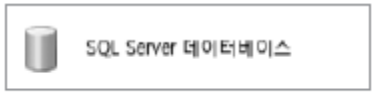
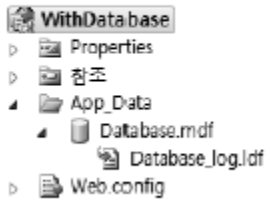


그림 22-5 현재 프로젝트 파일 구성



22.2 테이블 만들기

❖ 데이터베이스의 가장 중요한 요소는 테이블Table

표 22-1 도서관과 데이터베이스

도서관	데이터베이스
책	테이블
정보	데이터

그림 22-6 데이터베이스 구성

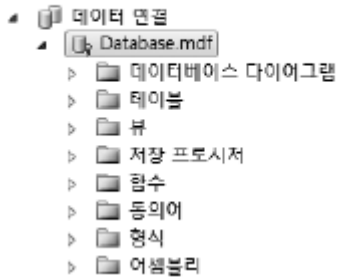
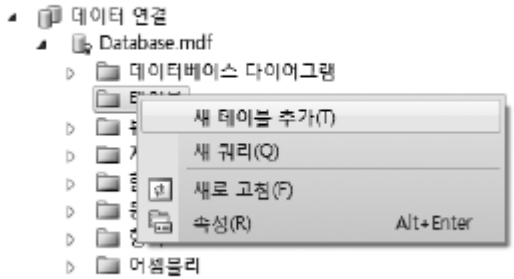


그림 22-7 새 테이블 추가



22.2 테이블 만들기

❖ 테이블 추가

- 테이블 추가 버튼을 누르면 간단한 스프레드시트 실행
- 열 이름 입력 - 객체의 속성
- 데이터 형식 – 어떠한 데이터를 저장할 것인지 의미
  - int는 정수Integer
  - Nvarchar (national character varying) 는 문자열 의미
    - **nvarchar(MAX)라고 입력한 것은 저장할 수 있는 최대 길이까지 문자열 저장**

그림 22-8 테이블 디자인

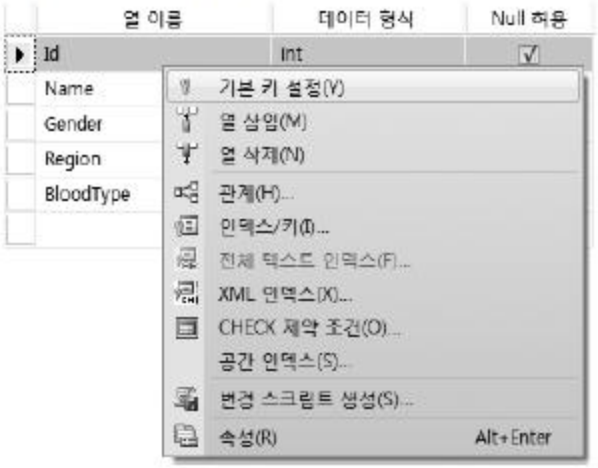
열 이름	데이터 형식	Null 허용
Id	int	☑
Name	nvarchar(MAX)	☑
Gender	nvarchar(MAX)	☑
Region	nvarchar(MAX)	☑
BloodType	nvarchar(MAX)	☑
		☐

22.2 테이블 만들기

❖ 기본 키 설정

- 작성한 테이블의 Id 행에 기본 키 설정
- ASP.NET MVC 3에서 데이터를 가져오려면 기본 키 반드시 설정해야

그림 22-9 기본 키 설정

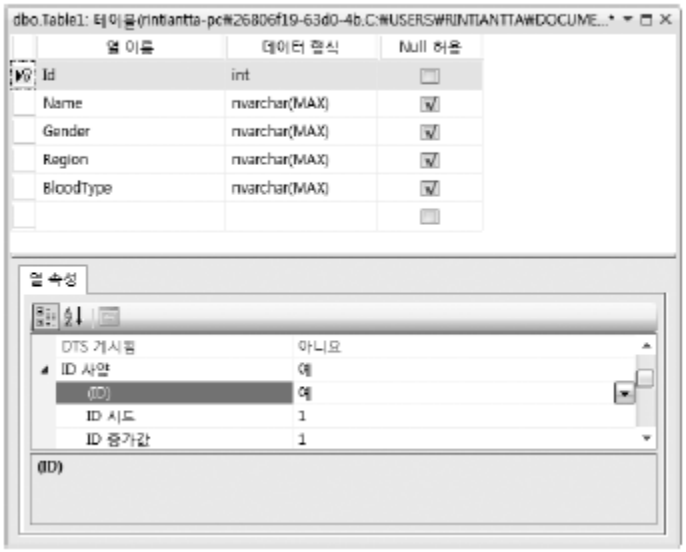


22.2 테이블 만들기

❖ 기본 키 중복 방지 설정

- Id를 선택 후 열 속성에서 'ID 사양' 속성을 '예'로 바꿈
  - 테이블에 데이터를 추가할 때 기본 키 자동 설정

그림 22-10 ID 시드 설정



22.2 테이블 만들기

❖ 테이블 저장과 데이터 입력

그림 22-11 테이블 저장

그림 22-12 테이블 데이터 표시

그림 22-13 테이블에 데이터 입력

22.3 LINQ To SQL

❖ 코드에서 데이터베이스로 접근하는 방법

- 가장 간단한 LINQ To SQL

그림 22-14 Models 폴더에 새 항목 만들기

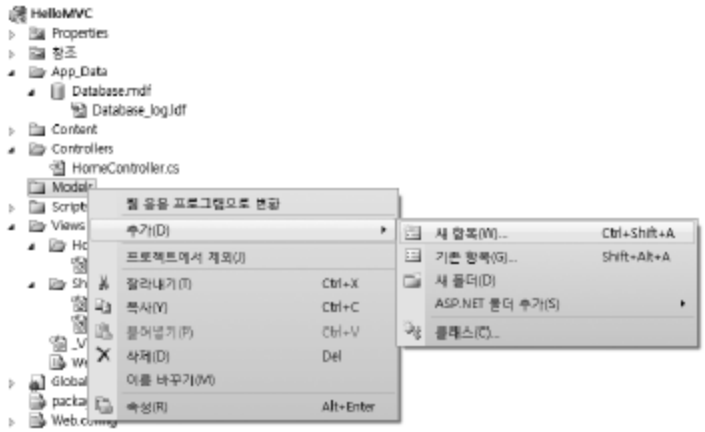


그림 22-15 LINQ to SQL 클래스

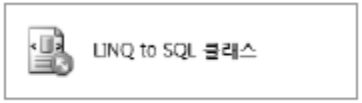
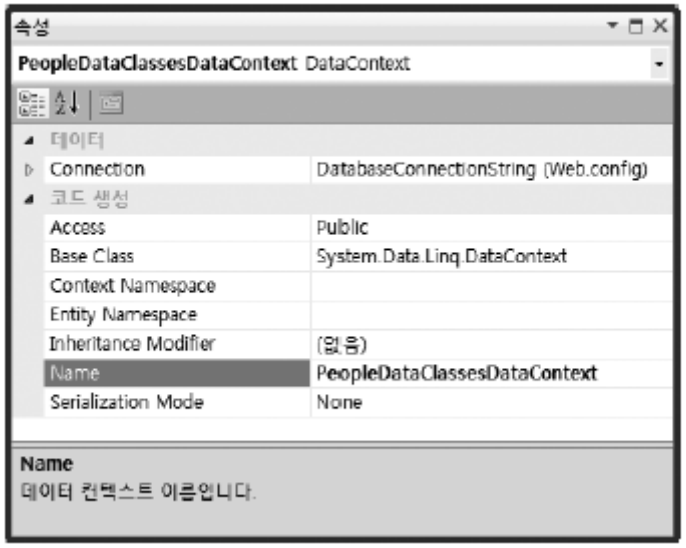


그림 22-16 People 테이블을 드래그합니다.

❖ 코드에서 데이터베이스로 접근하는 방법

- 테이블 속성 보기
- 모든 과정이 끝나면 Ctrl + Shift + B 키 눌러 빌드

그림 22-17 PeopleDataClassesDataContext의 속성



22.4 데이터베이스에서 데이터 가져오기

❖ 데이터를 JSON 형태로 제공하는 웹 서비스 생성

- 액션 GetPeopleJSON 생성

코드 22-1 HomeController.cs : 액션 GetPeopleJSON 생성

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }

    public ActionResult GetPeopleJSON()
    {

    }
}
```

❖ 데이터를 JSON 형태로 제공하는 웹 서비스 생성 (2)

코드 22-2 HomeController.cs : PeopleDB 객체 생성

```
public ActionResult GetPeopleJSON()
{
    var peopleDB = new PeopleDB();
}
```

그림 22-18 파란색 상자

```
public ActionResult GetPeopleJSON()
{
    var peopleDB = new PeopleDB();
}
```

그림 22-19 'using WithDatabase.Models;' 선택

```
public ActionResult GetPeopleJSON()
{
    var peopleDB = new PeopleDB();
}
```

using WithDatabase.Models;

WithDatabase.Models.PeopleDB

'PeopleDB'에 대한 클래스 생성(C)

새 형식 생성(T)...

22.4 데이터베이스에서 데이터 가져오기

❖ Using Database

코드 22-3 HomeController.cs : using 구문 추가

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using WithDatabase.Models;
```

코드 22-4 HomeController.cs : Json() 메서드의 사용

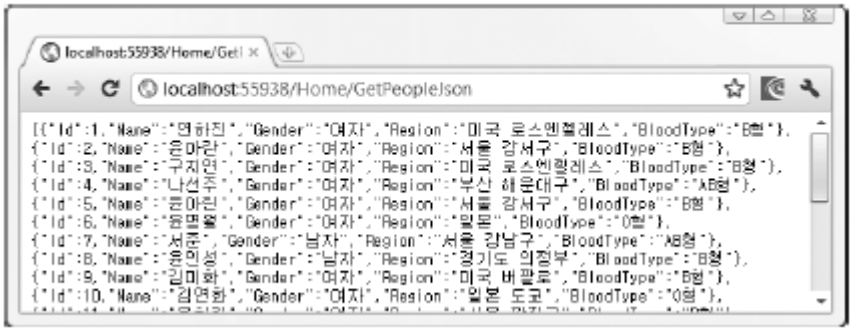
```
public ActionResult GetPeopleJSON()
{
    var peopleDB = new PeopleDB();
    return Json(peopleDB.People, JsonRequestBehavior.AllowGet);
}
```

22.4 데이터베이스에서 데이터 가져오기

❖ 출력 결과 확인

- JsonRequestBehavior.AllowGet
  - Get 방식으로 쉽게 데이터 확인
- 프로젝트를 실행 후 http://localhost:Port/Home/GetPeopleJSON 접속
- 브라우저가 JSON 파일 다운로드 한다면 메모장에서 열어 결과 확인
- 크롬을 사용한다면 간단하게 그림 22-20처럼 JSON 출력

그림 22-20 데이터베이스의 모든 데이터를 JSON으로 출력



22.5 데이터베이스에 데이터 추가

❖ 데이터 추가 액션 InsertPerson

- 매개 변수로 name, gender, region, bloodType 받음
- 위 매개 변수로 데이터 만들어 테이블에 추가

코드 22-5 HomeController.cs : 액션 InsertPerson 생성

```
public ActionResult InsertPerson(string name, string gender,
    string region, string bloodType)
{

}
```

코드 22-6 HomeController.cs: People 객체 생성 및 초기화

```
public ActionResult InsertPerson(string name, string gender,
    string region, string bloodType)
{
    // 테이블에 입력할 데이터를 만듭니다.
    var willAdd = new People();
    willAdd.Name = name;
    willAdd.Gender = gender;
    willAdd.Region = region;
    willAdd.BloodType = bloodType;
}
```

22.5 데이터베이스에 데이터 추가

❖ 생성한 객체 테이블에 넣기

- PeopleDB 클래스의 객체 생성
- InsertOnSubmit() 메서드와 SubmitChanges() 메서드 순서대로 실행
- 자료가 문제 없이 넣어지면 문자열 'INSERT SUCCESS' 출력

코드 22-7 HomeController.cs: 테이블에 데이터 추가

```
public ActionResult InsertPerson(string name, string gender,
    string region, string bloodType)
{
    // 테이블에 입력할 데이터를 만듭니다.
    var willAdd = new People();
    willAdd.Name = name;
    willAdd.Gender = gender;
    willAdd.Region = region;
    willAdd.BloodType = bloodType;

    // 테이블에 데이터를 넣습니다.
    var peopleDB = new PeopleDB();
    peopleDB.People.InsertOnSubmit(willAdd);
    peopleDB.SubmitChanges();

    return Content("INSERT SUCCESS");
}
```

22.5 데이터베이스에 데이터 추가

❖ HttpNotFound() 메서드

- 예외가 발생하면 Page Not Found 에러 페이지 출력

코드 22-8 HomeController.cs: 액션 InsertPerson 완성

```
public ActionResult InsertPerson(string name, string gender,
    string region, string bloodType)
{
    // 테이블에 입력할 데이터를 만듭니다.
    var willAdd = new People();
    willAdd.Name = name;
    willAdd.Gender = gender;
    willAdd.Region = region;
    willAdd.BloodType = bloodType;
    try
    {
        // 테이블에 데이터를 넣습니다.
        var peopleDB = new PeopleDB();
        peopleDB.People.InsertOnSubmit(willAdd);
        peopleDB.SubmitChanges();
        return Content("INSERT SUCCESS");
    }
    catch (Exception exception)
    {
        // 데이터 입력에 실패할 경우
        return HttpNotFound();
    }
}
```

22.6 데이터베이스의 데이터 삭제

❖ 데이터 삭제하는 방법

- 대부분 기본 키 입력 받아 지움
- 액션 DeletePerson
  - 테이블에서 해당 데이터를 가져와야 함
  - Single() 메서드 사용하면 People 테이블에서 조건에 일치하는 데이터 추출
    - **x => x.Id = = id ' 는 C#에서 람다식이라고 부르며, "데이터를 x라 했을 때 x객체의 Id 속성이 id와 같은 것을 추출한다" 는 의미**

코드 22-10 HomeController.cs: Single() 메서드의 사용

```
public ActionResult DeletePerson(int id)
{
    // 삭제할 데이터를 선택합니다.
    var peopleDB = new PeopleDB();
    var willDelete = peopleDB.People.Single(x => x.Id == id);
}
```

22.6 데이터베이스의 데이터 삭제

❖ 데이터 삭제

- DeleteOnSubmit() 메서드와 SubmitChanges() 메서드 사용
- DeleteOnSubmit() 메서드의 첫 번째 변수에 삭제할 데이터 넣어줌
- 데이터 성공적으로 제거되면 문자열 'DELETE SUCCESS ' 출력

코드 22-11 HomeController.cs: 테이블의 데이터 삭제

```
public ActionResult DeletePerson(int id)
{
    // 삭제할 데이터를 선택합니다.
    var peopleDB = new PeopleDB();
    var willDelete = peopleDB.People.Single(x => x.Id == id);

    // 삭제합니다.
    peopleDB.People.DeleteOnSubmit(willDelete);
    peopleDB.SubmitChanges();

    return Content("DELETE SUCCESS");
}
```

22.6 데이터베이스의 데이터 삭제

❖ 예외 처리

코드 22-12 HomeController.cs: 액션 DeletePerson 완성

```
public ActionResult DeletePerson(int id)
{
    try
    {
        // 삭제할 데이터를 선택합니다.
        var peopleDB = new PeopleDB();
        var willDelete = peopleDB.People.Single(x => x.Id == id);


        // 삭제합니다.
        peopleDB.People.DeleteOnSubmit(willDelete);
        peopleDB.SubmitChanges();
        return Content("DELETE SUCCESS");
    }
    catch (Exception excception)
    {
        return HttpNotFound();
    }
}
```

22.6 데이터베이스의 데이터 삭제

❖ 실행 결과

- 실행 후 http://localhost:Port/Home/DeletePerson?id=1 접속
  - 'DELETE SUCCESS'라는 문자열 출력
  - 데이터베이스 탐색기에서 테이블 살펴보면 데이터가 제거된 것 확인

❖ 존재하지 않는 데이터 찾아 에러 처리 확인

- http://localhost:Port/Home/DeletePerson?id=9999 에 접속
  - 데이터베이스에 9999번 데이터 없으므로 예외가 발생
  - HttpNotFound() 메서드 실행한 결과

그림 22-21 HttpNotFound() 메서드

22.7 데이터베이스의 데이터 수정

❖ 데이터 수정하는 액션 UpdatePerson

- 매개 변수로 id와 region 입력 받음
- 수정할 데이터 가져옴
  - 데이터 제거할 때와 마찬가지로 Single() 메서드 사용

코드 22-14 HomeController.cs: Single() 메서드의 사용

```
public ActionResult UpdatePerson(int id, string region)
{
    // 수정할 데이터를 선택합니다.
    var peopleDB = new PeopleDB();
    var willUpdate = peopleDB.People.Single(x => x.Id == id);
}
```

코드 22-15 HomeController.cs: 테이블의 데이터 수정

```
public ActionResult UpdatePerson(int id, string region)
{
    // 수정할 데이터를 선택합니다.
    var peopleDB = new PeopleDB();
    var willUpdate = peopleDB.People.Single(x => x.Id == id);

    // 수정합니다.
    willUpdate.Region = region;
    peopleDB.SubmitChanges();

    return Content("UPDATE SUCCESS");
}
```

22.7 데이터베이스의 데이터 수정

❖ 수정 시 예외 처리

- 프로젝트 실행 후 "http://localhost:22/Home/UpdatePerson?id=1&Region=한국 " 접속
- 'UPDATE SUCCESS'라는 문자열 출력
- 데이터 수정 확인

코드 22-16 HomeController.cs : 액션 UpdatePerson 완성

```
public ActionResult UpdatePerson(int id, string region)
{
    try
    {
        // 수정할 데이터를 선택합니다.
        var peopleDB = new PeopleDB();
        var willUpdate = peopleDB.People.Single(x => x.Id == id);

        // 수정합니다.
        willUpdate.Region = region;
        peopleDB.SubmitChanges();
        return Content("UPDATE SUCCESS");
    }
    catch (Exception exception)
    {
        return HttpNotFound();
    }
}
```

22.8 Ajax를 사용한 데이터 추가와 삭제

❖ Index 뷰 생성

- 생성한 뷰에 코드 22-17처럼 body 태그 구성

코드 22-17 Index.cshtml: body 태그 구성

```
<body>
    <div>
        <form id="insert_form">
        <fieldset>
            <legend>데이터 추가</legend>
            <table>
                <tr>
                    <td><label>이름</label></td>
                    <td><input type="text" name="name"/></td>
                </tr>
                <tr>
                    <td><label>성별</label></td>
                    <td><input type="text" name="gender"/></td>
                </tr>
                <tr>
                    <td><label>지역</label></td>
                    <td><input type="text" name="region"/></td>
```

22.8 Ajax를 사용한 데이터 추가와 삭제

❖ Index 뷰 생성 (2)

```
                </tr>
                <tr>
                    <td><label>혈액형</label></td>
                    <td><input type="text" name="bloodType"/></td>
                </tr>
            </table>
            <input type="submit" value="추가"/>
        </fieldset>
        </form>
    </div>
    <table id="output">

    </table>
</body>
```

그림 22-22 생성한 body 태그

22.8 Ajax를 사용한 데이터 추가와 삭제

❖ Head 태그 구성

코드 22-18 Index.cshtml: body 태그구성

```
<head>
    <title>Index</title>
    <script src="http://code.jquery.com/jquery-1.7.js"></script>
    <script>
        $(document).ready(function () {
            // 데이터를 보여주는 함수입니다.
            function selectData() {
                // #output 내부의 내용물을 제거합니다.
                $('#output').empty();

                // Ajax를 수행합니다.
            }

            // 데이터를 추가합니다.
            $('#insert_form').submit(function (event) {
                // 기본 이벤트를 제거합니다.
                event.preventDefault();
            });

            // 초기 화면에 데이터를 표시합니다.
            selectData();
        });
</script>
</head>
```

22.8 Ajax를 사용한 데이터 추가와 삭제

❖ 데이터 입력 페이지

코드 22-19 Index.cshtml : 완성

```
$(document).ready(function () {
    // 데이터를 보여주는 함수
    function selectData() {
        // #output 내부의 내용을 제거합니다.
        $('#output').empty();

        // Ajax를 수행합니다.
        $.getJSON('/Home/GetPeopleJSON', function (data) {
            $(data).each(function (index, item) {
                var output = '';
                output += '<tr>';
                output += '    <td>' + item.Id + '</td>';
                output += '    <td>' + item.Name + '</td>';
                output += '    <td>' + item.Gender + '</td>';
                output += '    <td>' + item.BloodType + '</td>';
                output += '</tr>';

                $('#output').append(output);
            });
        });
    }
```

22.8 Ajax를 사용한 데이터 추가와 삭제

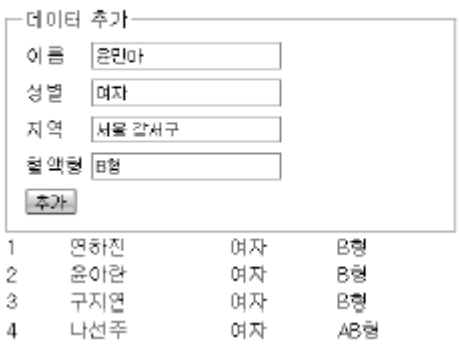
❖ 데이터 입력 페이지 (2)

```
// 데이터를 추가합니다.
    $('#insert_form').submit(function (event) {
        // Ajax를 수행합니다.
        var data = $(this).serialize();
        $.post('/Home/InsertPerson', data, selectData);

        // 기본 이벤트를 제거합니다.
        event.preventDefault();
    });

    // 초기 화면에 데이터를 표시합니다.
    selectData();
});
```

그림 22-23 실행 결과



Thank You !

모던 웹을 위한 Javascript jQuery 입문

한빛미디어 Hanbit Media, Inc.