## **DATABASE**

### **1. user TABLE**

1. userID : 로그인 하기 위한 사용자 ID
2. userPW : 로그인 하기 위한 사용자 Password
3. userName : 사용자 이름
4. userGender : 사용자 성별
5. userEmail : 사용자 이메일

### **2. communitylist Table**

1. communityCode : 각 커뮤니티를 고유하게 구분하기 위한 식별자 (Auto Increase)
2. communityName : 커뮤니티의 이름
3. communityDes : 커뮤니티의 설명

### **3. communitypost Table**

1. communityPostCode : 각 게시글을 고유하게 구분하기 위한 식별자 (Auto Increase)
2. communityPostName : 게시글의 제목
3. communityPostContent : 게시글의 내용
4. communityPostUser : 게시글을 작성한 사용자 ID ( Foreign Key – user.userID / Guest라면 임의의 이름 )
5. communityPostTime : 게시글을 작성한 날짜 (GETDATE)
6. communityPostView : 게시글을 조회한 횟수
7. communityPostRecommand : 게시글을 추천한 횟수
8. communityCode : 게시글을 게시한 커뮤니티의 고유한 코드 ( Foreign Key – communitylist.communityCode )
9. communityPostAuth : 게시글을 게시한 사용자가 사용자인지, 게스트인지
10. communityPostPassword : 게시글을 게시한 사용자가 게스트라면 수정, 삭제에 필요한 Password

### **4. comment Table**

1. commentCode : 각 댓글을 고유하게 구분하기 위한 식별자 (Auto Increase)
2. commentPostCode : 댓글이 작성된 게시글의 고유한 코드 ( Foreign Key – communitypost.communityPostCode )
3. commentUser : 댓글을 작성한 사용자 ID ( Foreign Key – user.userID / Guest라면 임의의 이름 )
4. commentPassword : 댓글을 작성한 사용자가 게스트라면 수정, 삭제에 필요한 Password
5. comment : 댓글의 내용
6. commentAuth : 댓글을 작성한 사용자가 사용자인지, 게스트인지

## **DATABASE SQL**

Create table [user] (

userID varchar(20) primary key,

userPW varchar(16) not null,

userName nvarchar(20) not null,

userGender nvarchar(10) not null,

userEmail varchar(30) not null

)

insert into [user] values('root', 'root', N'관리자', N'선택 안함', 'myaspcommunity@gmail.com')

Create table communitylist (

communityCode int IDENTITY (1, 1) primary key,

communityName nvarchar(20) not null unique,

communityDes nvarchar(100) not null,

)

insert into communitylist (communityName, communityDes) values(N'ASP 프로그래밍', N'ASP 프로그래밍에 대해 이야기를 나누는 커뮤니티입니다')

insert into communitylist (communityName, communityDes) values(N'ASP 프로그래밍 2', N'ASP 프로그래밍 커뮤니티 대피소')

insert into communitylist (communityName, communityDes) values(N'개발자들의 쉼터', N'모든 종류의 개발자들을 환영합니다')

insert into communitylist (communityName, communityDes) values(N'로스트아크', N'"여러분들의 근사한 모험이야말로 저희가 간절하게 꿈꾸는 추억입니다."')

insert into communitylist (communityName, communityDes) values(N'에픽세븐', N'스마일게이트의 모바일 게임 에픽세븐에 대한 이야기를 나눕니다')

insert into communitylist (communityName, communityDes) values(N'블루아카이브', N'몰?루')

insert into communitylist (communityName, communityDes) values(N'리그 오브 레전드', N'롤 좋아하시는 모든 분들 환영~')

Create table communitypost (

communityPostCode int IDENTITY (1, 1) primary key,

communityPostName nvarchar(50) not null,

communityPostContent nvarchar(500) not null,

communityPostUser nvarchar(20) not null,

communityPostTime datetime DEFAULT GETDATE(),

communityPostView int default 0,

communityPostRecommand int default 0,

communityCode int not null,

communityPostAuth nvarchar(20),

communityPostPassword nvarchar(20)

)

Create table comment (

commentCode int IDENTITY (1, 1) primary key,

commentPostCode int not null,

commentUser nvarchar(20) not null,

commentPassword nvarchar(20) not null,

comment nvarchar(300) not null,

commentAuth nvarchar(20) not null

)

## **페이지 기능**

### **1. Authentication**

1. Login
   * **로그인 기능을 수행하는 페이지**
   * 커뮤니티에서 회원으로 활동하기 위해 가입된 아이디와 비밀번호로 로그인 할 수 있는 페이지
   * 확인 버튼을 누를 시 Action
     + ID TextBox 혹은 PW TextBox 가 입력되지 않았다면 Alert 후 Return
     + [user] TABLE 접근하여 userID = ID TextBox 인 Data 를 Select
     + 해당 Data의 userPW 가 PW TextBox 와 일치한다면 FormsAuthentication 을 사용하여 로그인 처리
       - 일치하지 않는다면 Alert
   * 회원가입 버튼을 누를 시 Action
     + PostBackUrl 사용하여 회원가입 페이지로 리다이렉션
2. Register
   * **회원가입 기능을 수행하는 페이지**
   * 커뮤니티에 회원으로 가입할 수 있는 페이지
   * 아이디, 비밀번호, 이름, 성별, 이메일, 이메일 인증 코드를 TextBox 로 입력 받음
   * RequiredField 와 RegularExpression Validation 컴포넌트를 사용하여 유효성 검사
   * 이메일 인증 구현 **( MailEnable 사용 )**
     + GMAIL 으로 가상 계정인 [myaspcommunity@gmail.com](mailto:myaspcommunity@gmail.com) 계정 생성
     + 입력한 이메일로 인증 메일 전송
       - randomString 함수를 선언하여 10자의 랜덤한 문자열 생성 후 전송
     + Session에 해당 랜덤한 문자열을 저장
     + 이메일 인증 코드 TextBox 에 Session과 동일한 문자열이 입력되었다면 통과
   * 유효성 검사를 전부 통과하면 user DATABASE 에 정보 Insert
   * 회원가입이 정상적으로 완료되었다면 Login Page 로 리다이렉션

### **2. Page**

1. indexPage
   * **기본 페이지**
   * 가장 처음 보여지는 페이지이자 오류가 발생했을 때 리다이렉션되는 페이지
2. communitySearchPage
   * **커뮤니티를 검색하는 페이지**
   * 마스터 페이지 (MasterPG) 의 검색 TextBox 에 원하는 커뮤니티를 입력하면 리다이렉션 되는 페이지
   * Page에는 DIV 태그로 출력하고자 하는 위치를 지정
   * 쿼리스트링의 Keyword를 받아서 해당 문자열이 포함된 커뮤니티 리스트를 데이터베이스를 활용하여 조회
     + cmd.Parameters.AddWithValue ("@communityName", "%" + keyword + "%");
   * While을 사용하여 조회된 모든 결과를 Loop
     + HtmlGenericControl 객체를 사용하여 새로운 DIV 태그 생성
       - 해당 DIV 태그에는 Attributes.Add 메서드를 사용하여 출력하고자 하는 모양을 잡음
     + 커뮤니티 리스트를 만들기 위해 Image 태그, LinkButton 태그, Label 태그 를 만들어진 DIV 태그에 부착
     + Page의 DIV 태그에 만들어진 DIV 태그 Add
   * 출력된 커뮤니티 리스트의 커뮤니티 이름 LinkButton을 클릭하면 해당 커뮤니티로 이동 (QueryString 사용)
3. postListPage
   * **특정 커뮤니티 페이지에서 해당 페이지의 게시글을 출력하는 페이지**
   * QueryString이 없거나 비정상적이면 indexPage로 리다이렉션
   * 쿼리 스트링의 Page를 받아서 해당 커뮤니티를 데이터베이스를 활용하여 조회
   * IsPostBack 속성을 사용하여 만약 방문한적이 없는 커뮤니티라면 최근 방문한 커뮤니티에 추가
   * 커뮤니티 리스트에서 해당 커뮤니티의 코드(기본키)를 받아서 그 코드로 커뮤니티 게시글 리스트 데이터베이스를 조회하여 Page의 Table에 TableCell을 추가
     + 해당 게시글로 리다이렉션 하기 위해 LinkButton에 Page와 PostCode 쿼리 스트링 이용
   * 전체글 버튼을 클릭하면 해당 커뮤니티의 메인 페이지로 이동
   * 글쓰기 버튼을 클릭하면 postWritePage로 이동
4. postWritePage
   * **특정 커뮤니티 페이지에서 게시글을 새로 작성하는 페이지**
   * IsPostBack 속성을 사용하여 만약 방문한적이 없는 커뮤니티라면 최근 방문한 커뮤니티에 추가
   * 비 로그인 상태와 로그인 상태 두가지로 구분 (User.Identity.IsAuthenticated)
     + 이 두 상태는 Auth 컬럼을 사용하여 DB에서 정보를 찾을 때 확인
   * 비 로그인 상태
     + 작성자, 비밀번호, 제목, 내용을 입력할 수 있음
   * 로그인 상태
     + 작성자는 데이터베이스에서 자동으로 받아와서 PageLoad 시 자동으로 작성됨
     + 작성자와 비밀번호 TextBox는 Readonly 속성을 부여하여 사용자가 값을 넣을 수 없음
     + 제목과 내용은 임의로 입력 가능
   * 임시저장 버튼 클릭 시 Session에 제목과 내용이 저장됨
     + 값이 없을 시 Alert 후 저장되지 않음
   * 불러오기 버튼 클릭 시 Session에서 임시 저장된 글을 불러옴
     + 값이 없을 시 Alert 후 불러와지지 않음
   * 취소 버튼 클릭 시 해당 커뮤니티의 메인 페이지로 리다이렉션
   * 글쓰기 버튼 클릭 시 커뮤니티 게시글 데이터베이스에 해당 글이 저장
5. postEditPage
   * **특정 커뮤니티 페이지에서 특정 게시글을 수정하는 페이지**
   * postWritPage와 동일
   * 차이점은 Write 페이지는 Insert 사용, Edit 페이지는 Update 사용
6. PostReadPage
   * **특정 커뮤니티 페이지에서 게시글을 새로 작성하는 페이지**
   * IsPostBack 속성을 사용하여 만약 방문한적이 없는 커뮤니티라면 최근 방문한 커뮤니티에 추가
   * 쿼리 스트링의 PostCode를 받아서 게시글 데이터베이스에서 검색
   * 데이터베이스에서 정보를 받아와서 조회수 칼럼을 1 증가 후 출력
   * 댓글 데이터베이스에서 정보를 받아와서 HtmlGenericControl 객체를 사용하여 출력
     + 댓글의 삭제 버튼의 PostBackUrl에 댓글의 정보를 담아서 삭제할 때 정보를 추출 가능
   * 추천 버튼 클릭 시 쿼리 스트링의 PostCode를 받아와서 게시글 데이터베이스에서 검색 후 추천 컬럼 1 증가
   * 전체글 버튼 클릭 시 해당 커뮤니티의 메인 페이지로 리다이렉션
   * 수정, 삭제 버튼 클릭 시 OnCommand를 사용하여 CommandArgument를 쿼리 스트링을 사용하여 erPass 페이지로 리다이렉션
   * 댓글 입력 버튼을 클릭하면 해당 내용이 댓글 데이터베이스에 저장
     + WritePage와 동일하게 회원 비회원 구분
7. myInfoPage
   * **내 정보와 내가 작성한 글, 댓글을 확인하는 페이지**
   * 로그인 하면 LoginView에 의해 로그인 버튼이 있던 위치에 존재하는 내정보 버튼에 의해 이동
   * 사용자 이름, ID, 내가 쓴 게시글, 내가 쓴 댓글 확인 가능
   * 회원 탈퇴 가능

### **3. erPass**

1. erPass
   * **글 수정, 삭제를 위해 권한 확인하고 리다이렉션하는 페이지**
   * 쿼리 스트링을 사용하여 Post, PostCode, Mode 값을 추출
   * 만약 Mode가 commentRemove 라면
     + QueryString의 CommentCode를 추출
     + 해당 댓글을 데이터베이스에서 조회하여
     + 해당 작성자의 권한을 조회하여 회원이라면 ID가 일치한다면 바로 삭제
     + 비회원이라면 비밀번호 입력창을 페이지에 띄운 뒤 비밀번호가 일치한다면 삭제
   * 만약 Mode가 commetRemove가 아니라면
     + 권한이 회원이라면
       - 해당 글의 권한이 비회원이라면 바로 해당 게시글 페이지로 리턴
       - 해당 글의 권한이 회원이라면 ID가 일치한다면 바로 삭제 혹은 postEditPage로 리다이렉션
     + 권한이 비회원이라면
       - 해당 글의 권한이 회원이라면 바로 해당 게시글 페이지로 리턴
       - 해당 글의 권한이 비회원이라면 비밀번호 입력창을 페이지에 띄운 뒤 비밀번호가 일치한다면 삭제

### **4. MasterPage**

1. MasterPG.Master
   * **UI 구성하는 마스터 페이지**
   * BootStrap Import
   * NavBar 설정
     + 사이트 로고, 검색 TextBox, LoginView
   * Page 설정
     + 좌단 페이지
       - 최근 접속한 커뮤니티 리스트
     + 중앙 페이지
       - ContentPlaceHolder 사용하여 Page의 내용 출력