一组相关对象的集合，数据修改的单元

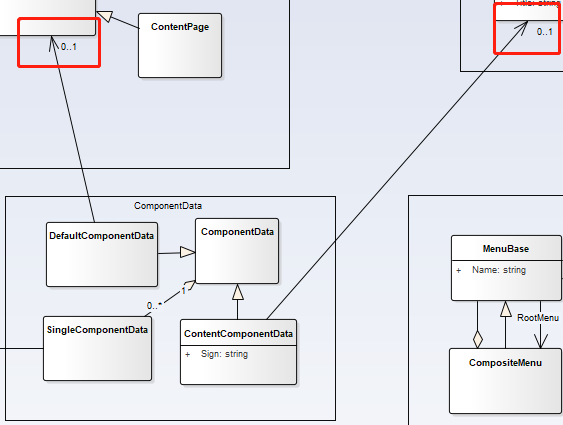
**聚合根**

访问聚和修改都从聚合根开始，聚合根是一张表，一个聚合只有一个聚合根

**聚合内部**

聚合内可以引用其他聚合根，外部对象不能引用聚合内部实体，可以将聚合内部实体引用传给外部，但只能是临时引用，聚合内部实体的声明周期由聚合根决定

如图：



聚合根是ComponentData（这里有一个疑惑？DefaultComponentData和ContentComponentData到底算不算聚合根）

**聚合根与服务的依赖关系**

服务应直接依赖与聚合根，而不要依赖与聚合内部实体

// 聚合内部实体

public class Entity1{

}

// 聚合根

public class Root {

    public Entity1 Entity1{ get; set; }

}

public class Service {

    // 不推荐这种方式

public Service(Entity1 e){}

    // 推荐使用这种方式

    public Service(Root e){

}

**聚合示例**

    // 文章，这里文章属于聚合根

    public class PageData:Entity, IHasCreationTime

    {

        public string Name { get; }

        ...

        // 文章标签

        public ICollection<Tag> Tags { get; set; }

        // 点击文章

        public void ToClick() {

            Click++;

        }

        // 文章评分

        public void ToScore(int score, int userId)

        {

            if (score < 0 || score > 10) {

                throw new UserFriendlyException("无效的评分范围");

            }

            string userIdStr = userId.ToString();

            List<string> userIds = ScoreUser == null ? new List<string>() : ScoreUser.Split("|").ToList();

            if (userIds.Any(e=>e == userIdStr)) {

                throw new UserFriendlyException("已评分，无法再次评分");

            }

            if (userIds.Count > 5000) {

                throw new UserFriendlyException("评分人数已达到上限");

            }

            double sum = Score \* ScoreNum;

            sum = sum + score;

            ScoreNum++;

            Score = sum / ScoreNum;

            userIds.Add(userIdStr);

            ScoreUser = userIds.JoinAsString("|");

        }

}

    // 标签，这里标签属于聚合内部实体

    public class Tag : Entity

    {

        public string Name { get; set; }

        public string DisplayName { get; set; }

        [ForeignKey("PageDataId")]

        public PageData PageData { get; set; }

        public int PageDataId { get; set; }

    }