عنوان: **توصیه‌هایی در مورد overloading متدها**
نویسنده: وحید نصیری
تاریخ: ۱۷:۳۵:۰۸ ۱۳۸۷/۱۱/۲۷
آدرس: www.dotnettips.info
برچسب‌ها: Tips

مطلب زیر چکیده‌ای است از کتاب Framework Design Guidelines در مورد سربارگذاری توابع

الف) از یک نوع خروجی استفاده کنید

مثال زیر را در نظر بگیرید:

```
public User[] GetGroupMembers(int groupId)
public List<User> GetGroupMembers(string groupName)
```

هر دوی این متدها گروهی از کاربران را بازگشت می‌دهند. خروجی متد اول از نوع آرایه است و خروجی متد دوم از نوع یک لیست جنریک می‌باشد.
مشکل اینجا است که هنگام فراخوانی هر کدام، نحوه‌ی استفاده متفاوت خواهد بود. بنابراین باید از این نوع سربارگذاری پرهیز کرد.

ب) نام‌های آرگومان‌های توابع سربارگذاری شده شما باید یکسان باشند

در مثال زیر، از اولین آرگومان جهت دریافت شناسه‌ی یک گروه استفاده می‌شود:

```
public List<User> GetGroupMembers(int groupId)
public List<User> GetGroupMembers(int id, int pageIndex, int pageSize)
```

اما همانطور که ملاحظه می‌کنید در یکی از groupId استفاده شده و در دیگری از نام id . این روش مزموم است! از آن پرهیز کنید! (هر دو یا باید groupId باشند و یا id . این هماهنگی و یکسان بودن باید در توابع سربارگذاری شده‌ی شما حفظ شود)

ج) ترتیب آرگومان‌ها را در توابع سربارگذاری شده حفظ و رعایت نمائید

لطفا به مثال زیر دقت کنید:

```
public List<User> GetGroupMembers(int groupId)
public List<User> GetGroupMembers(int pageIndex, int pageSize, int groupId)
```

در اینجا شناسه‌ی گروه یکبار به عنوان آرگومان اول معرفی شده و بار دیگر به عنوان آرگومان سوم. این کار نیز مزموم است، زیرا برنامه نویس‌ها از روی عادت به اولین آرگومان توابع سربارگذاری شده‌ی شما، همان شناسه‌ی گروه را ارسال خواهند کرد و این مورد سبب بروز باگ‌هایی خواهد شد که ردیابی آن مشکل است. بنابراین اگر شناسه‌ی گروه به عنوان اولین آرگومان معرفی شده، در تمامی overload های این تابع نیز اولین آرگومان باید همان شناسه‌ی گروه باشد.

د) از call forwarding استفاده کنید

جهت توضیح بهتر call forwarding به مثال زیر دقت نمائید:

```
public List<User> GetGroupMembers(int groupId)
{
   return GetGroupMembers(groupId, 0, 10);
}

public List<User> GetGroupMembers(int groupId, int pageIndex, int pageSize)
{
   return GetGroupMembers(groupId, pageIndex, pageSize, SortOrder.Ascending);
}

public List<User> GetGroupMembers(int groupId, int pageIndex, int pageSize, SortOrder sortOrder)
{
   var query = new GroupQuery();
   query.GroupID = groupId;
   query.PageIndex = pageIndex;
   query.PageSize = pageSize;
   query.SortOrder = sortOrder;

   return GetGroupMembers(query);
}

public List<User> GetGroupMembers(GroupQuery groupQuery)
{
   // Actual implementation to get group members goes here
}
```

معنای call forwarding این است که هنگام پیاده سازی توابعی از این دست، کوچکترین تابع سربارگذاری شونده باید تابع بزرگتر بعدی خود را صدا بزند و همین‌طور الی آخر که در مثال فوق به خوبی آشکار است. در این مثال چهارمین تابع سربارگذاری شده، بیشترین حوضه‌ی تمرکز را در خود دارد و توابع دیگر می‌توانند از آن بهره جویند بدون اینکه پیاده سازی خاصی را ارائه دهند.

ه) زیاده روی نکنید!

آخرین موردی که توصیه شده این است که در تهیه توابع سربارگذاری شده زیاده روی نکنید. در این نوع موارد باید قانون 20/80 را در نظر گرفت. 2 تا 3 تابع سربارگذاری شده ارائه دهید که 80 درصد کار را انجام می‌دهند و سپس یک تابع سربارگذاری شده‌ی دیگر ارائه دهید که 20 درصد باقیمانده را پوشش دهد.

پ.ن.
در سی شارپ 4 ، با معرفی optional parameters ، شاید کمتر به سربارگذاری نیاز باشد.