13. Organizing Your Model: Packages

Shaoning Zeng, http://zsn.cc

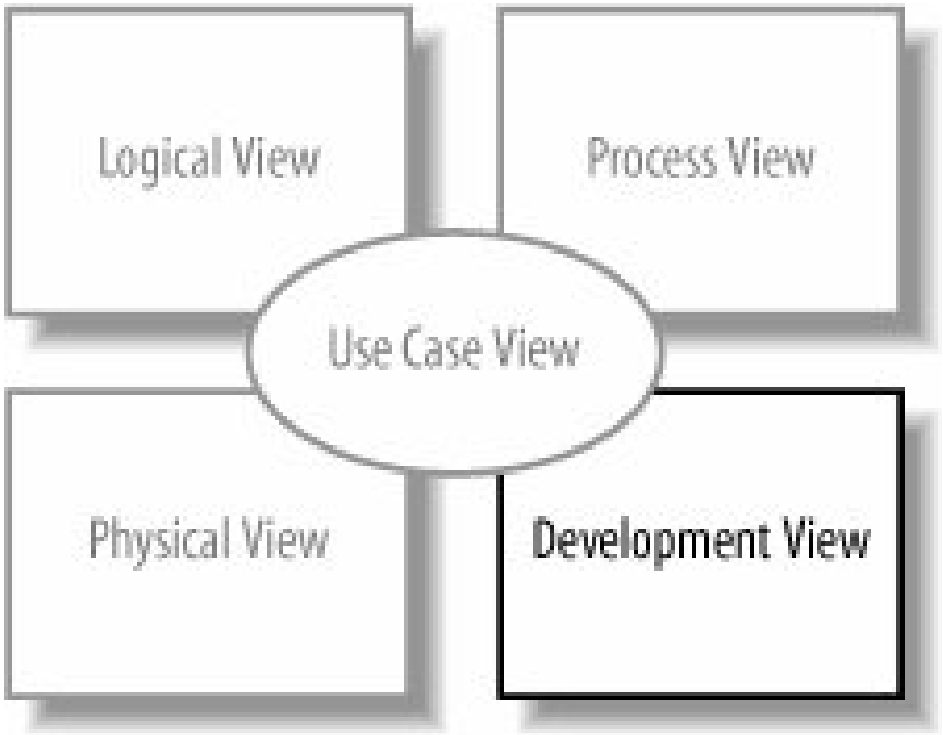
What we learnt?

- 11. Modeling a Class's Internal Structure: Composite Structures
- 12. Managing and Reusing Your System's Parts: Component Diagrams

13. Organizing Your Model: Packages

- 13.1. Packages 包
- 13.2. Namespaces and Classes Referring to Each Other
- 13.3. Element Visibility 可见性
- 13.4. Package Dependency 依赖关系
- 13.5. Importing and Accessing Packages 导入访问
- 13.6. Managing Package Dependencies
- 13.7. Using Packages to Organize Use Cases

Figure 13-1. The Development View describes how your system's parts are organized into modules, which are represented as packages in UML



13.1. Packages

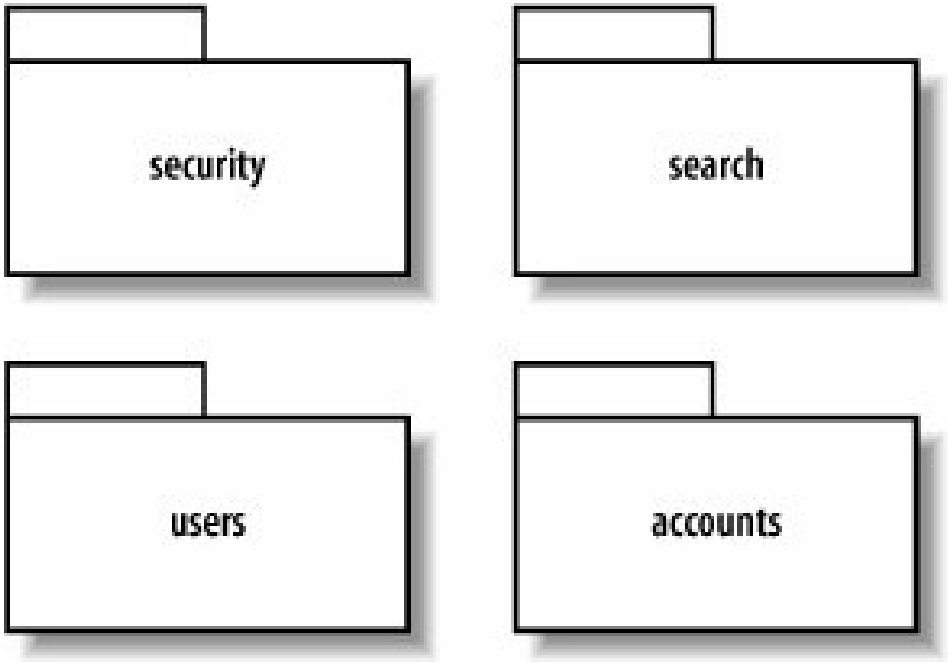


Figure 13-2. Packages in a CMS; each package corresponds to a specific system concern

Figure 13-3. Two ways to show that the Credentials and IdentityVerifier classes are contained in the security package

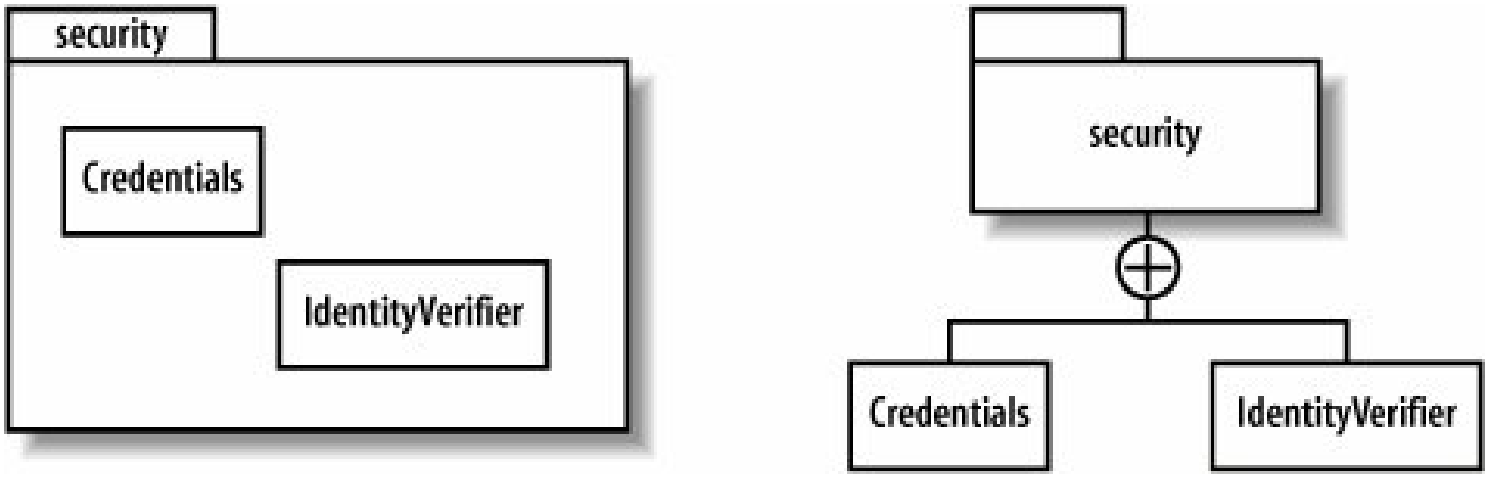


Figure 13-4. A package that contains another package

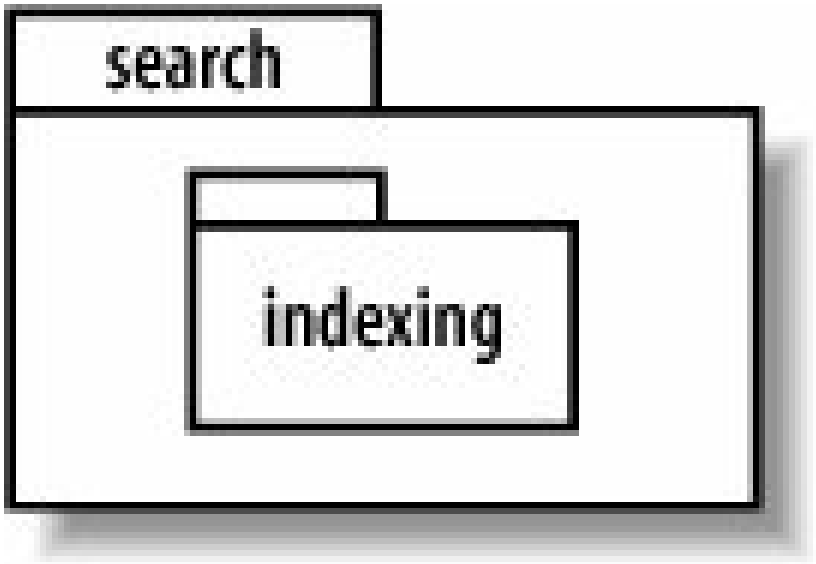


Figure 13-5. Deeply nested packages are common in enterprise applications: the search and indexing packages are shown in a typical package structure for the ACME company

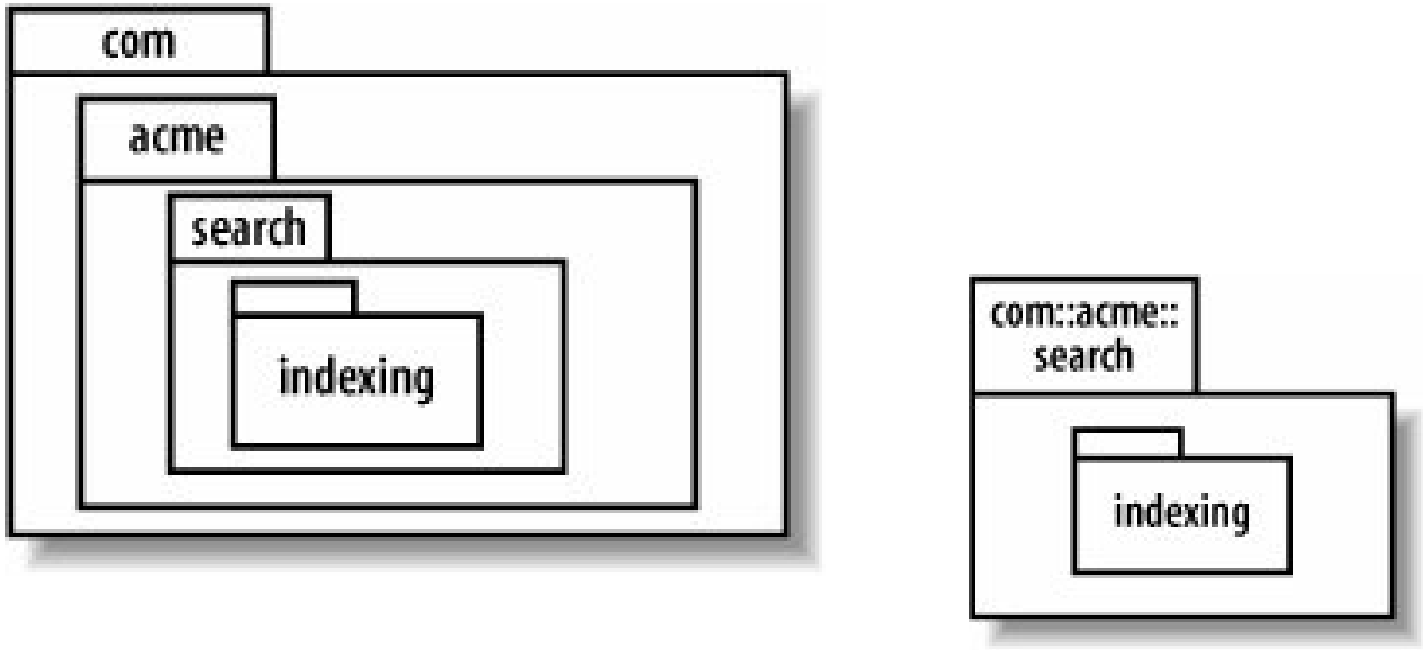


Figure 13-6. Flattening nested packages

Figure 13-7. Common ways UML tools show that a class belongs to a package

13.2. Namespaces and Classes Referring to Each Other 命名空间与类引用

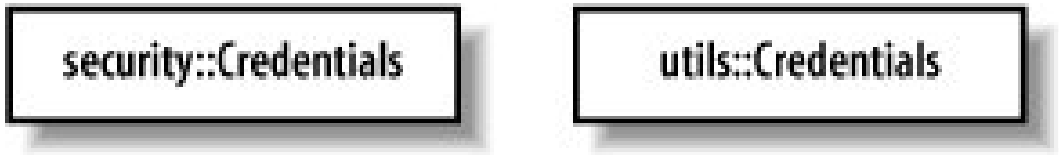


Figure 13-8. Representing a class with its fully-scoped name: both the security and utils packages have a class named Credentials

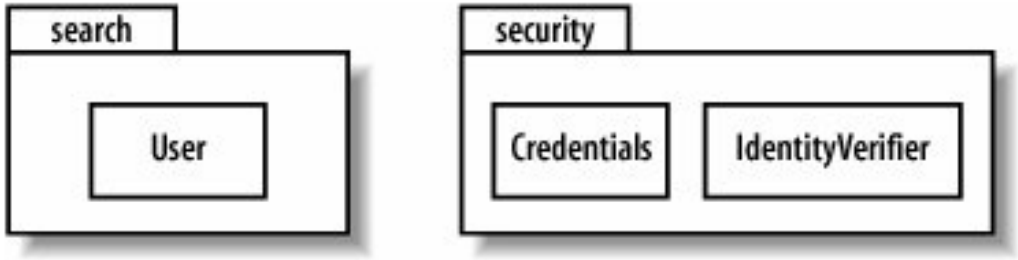


Figure 13-9. Classes in different packages have to provide name scope

13.3. Element Visibility

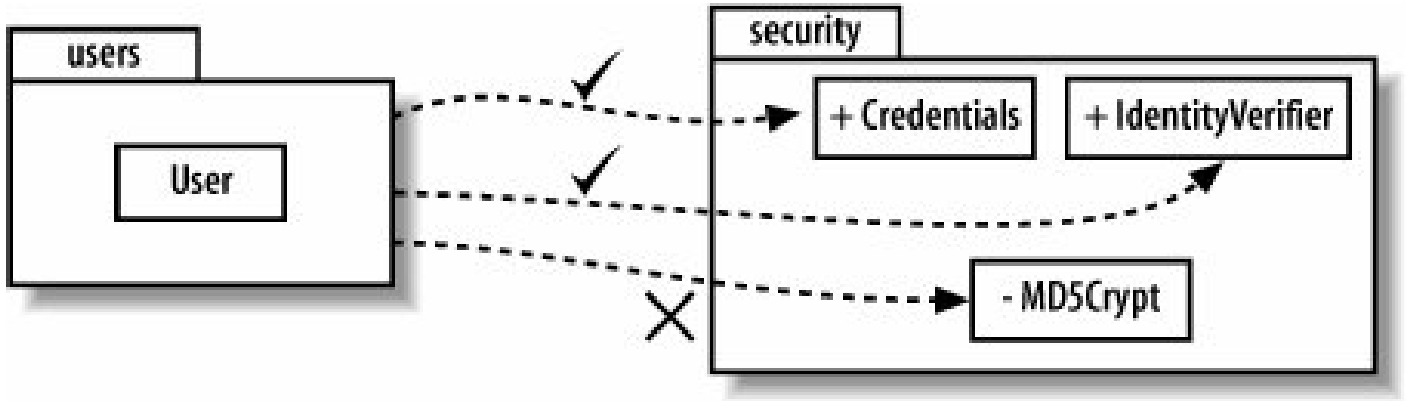


Figure 13-11. Since MD5Crypt has private visibility, it isn't accessible outside the security package

```
public class Credentials {}
```

13.4. Package Dependency

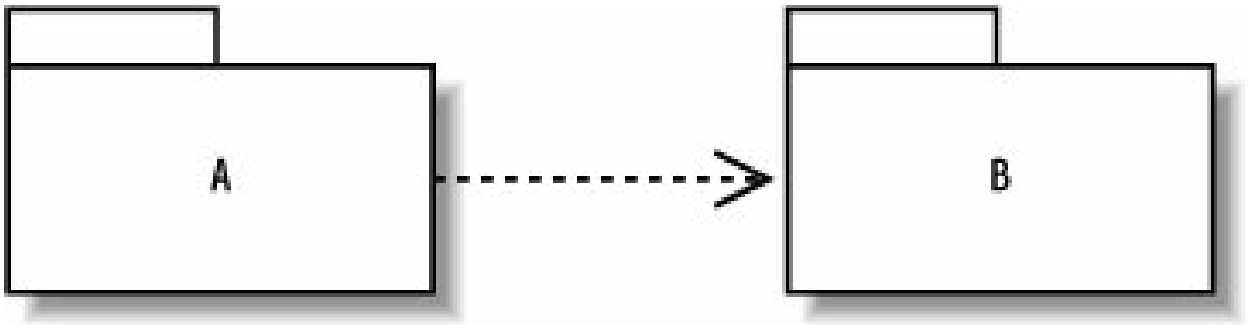
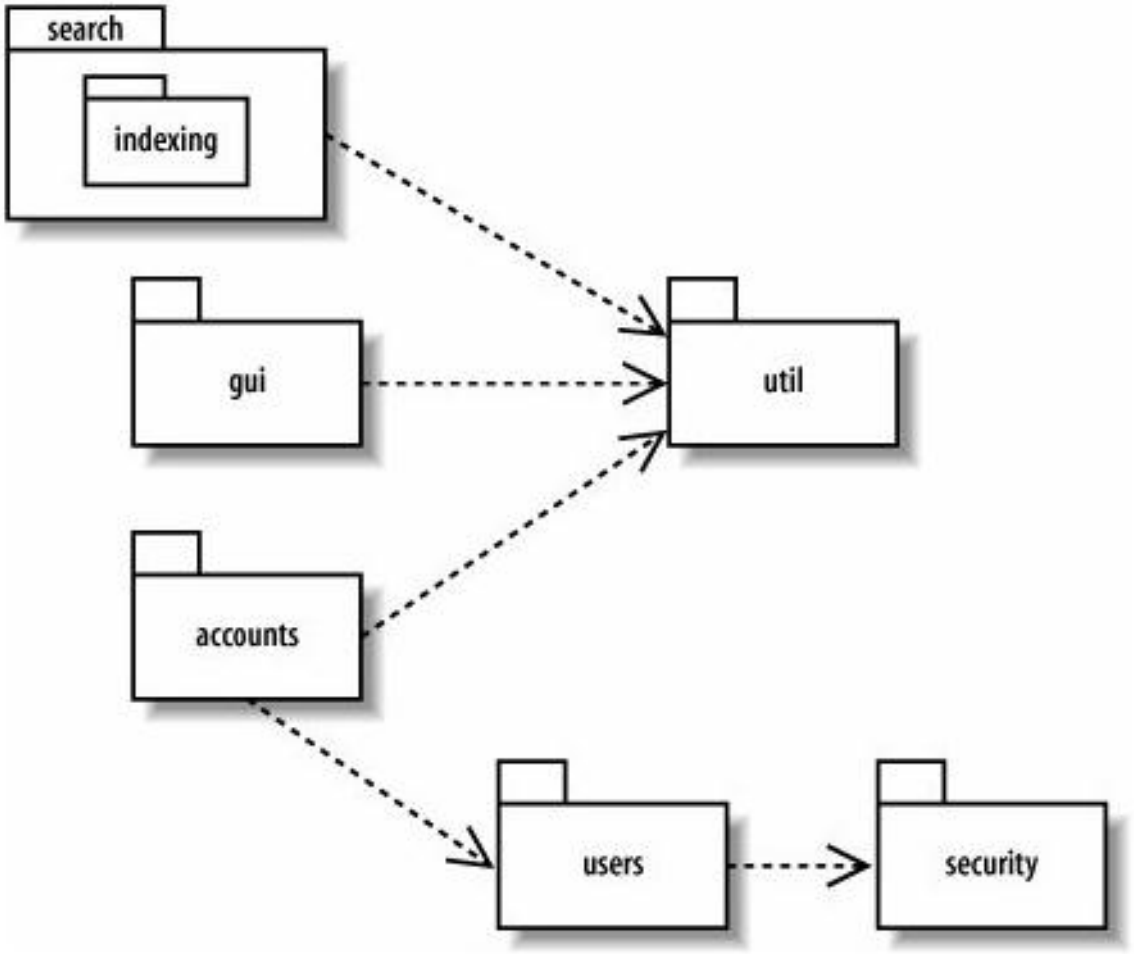
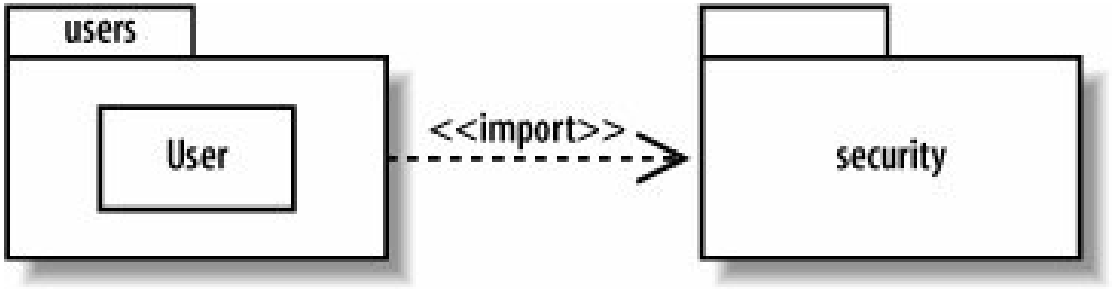


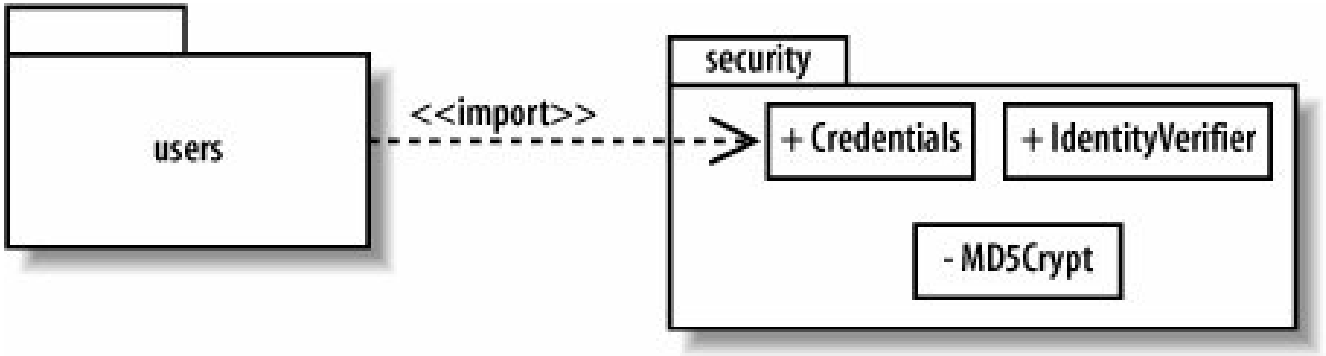
Figure 13-12. Package A depends on package B

Figure 13-13. A typical package diagram featuring core packages and dependencies



13.5. Importing and Accessing Packages





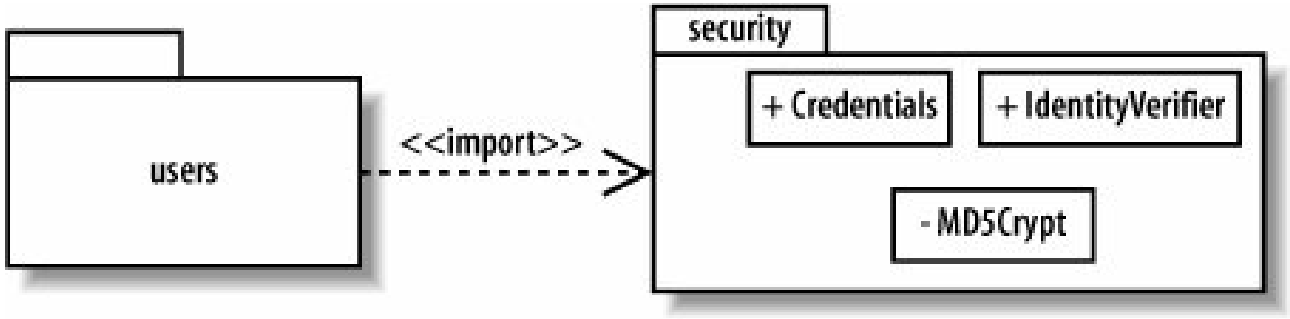


Figure 13-16. Private visibility causes a class not to be seen even though its package is imported

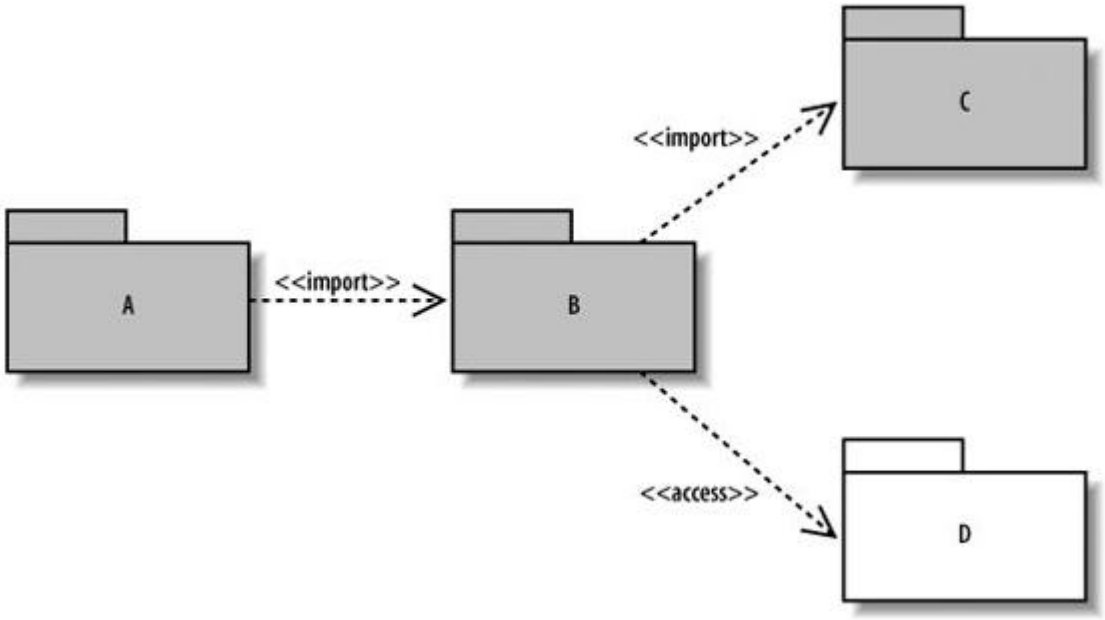


Figure 13-17. Package A can see public elements in C but not D

13.6. Managing Package Dependencies

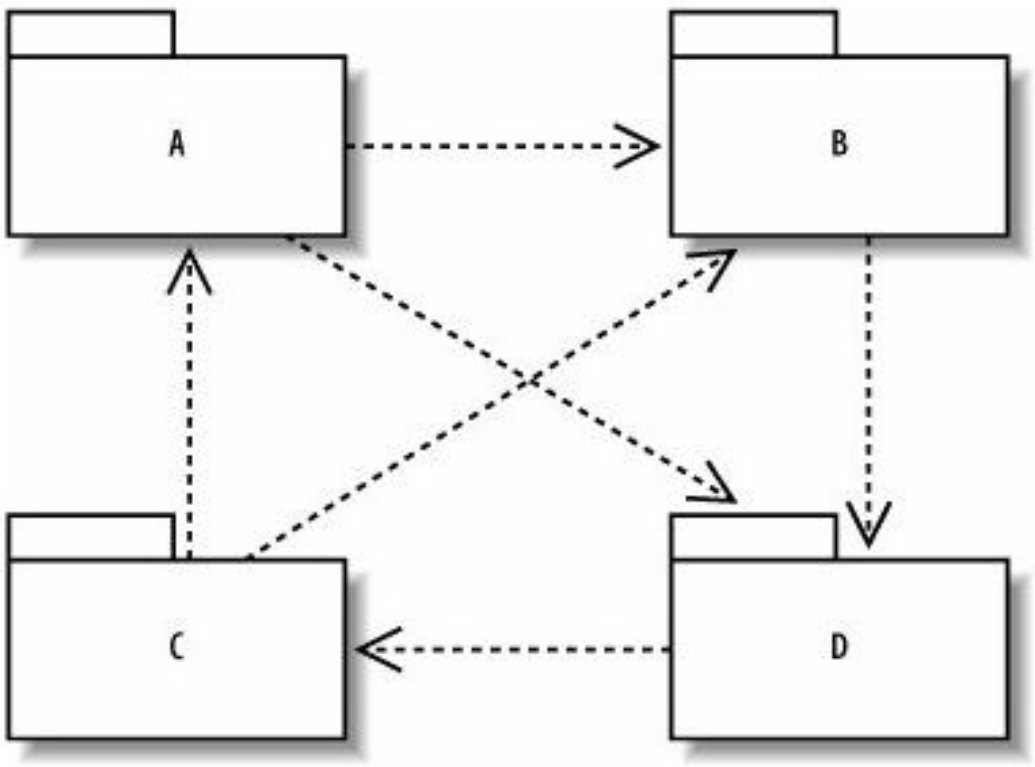
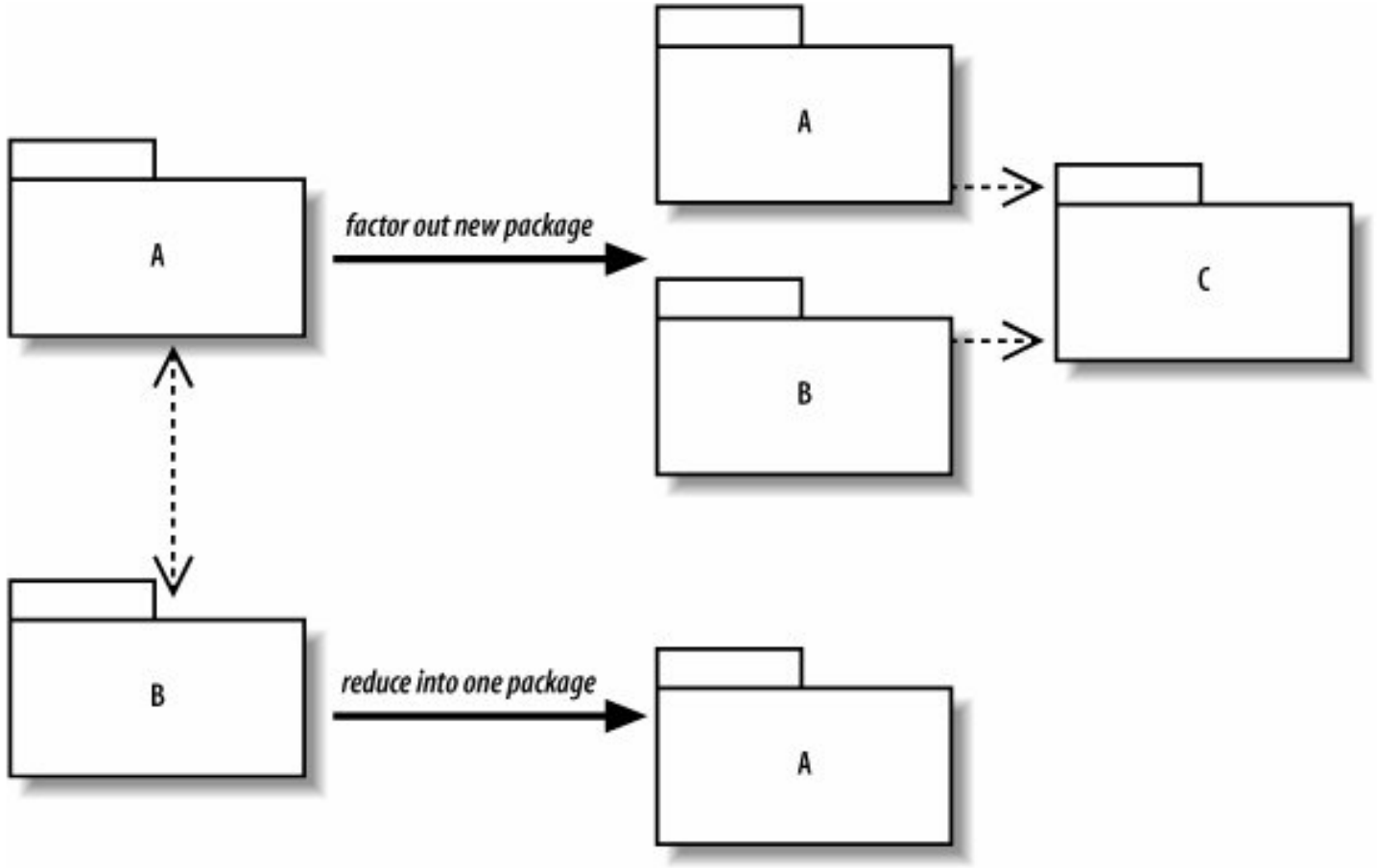


Figure 13-18. Directly or indirectly, a change in any one package could affect every other package

Figure 13-19. Removing cycles in package dependencies



13.7. Using Packages to Organize Use Cases

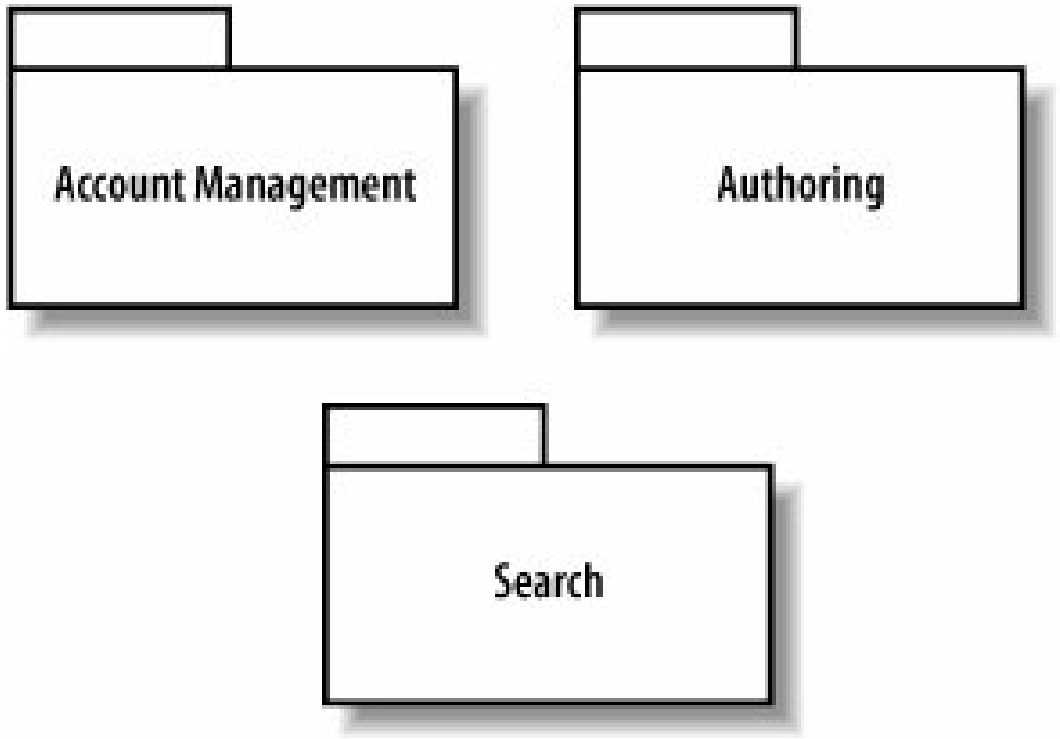
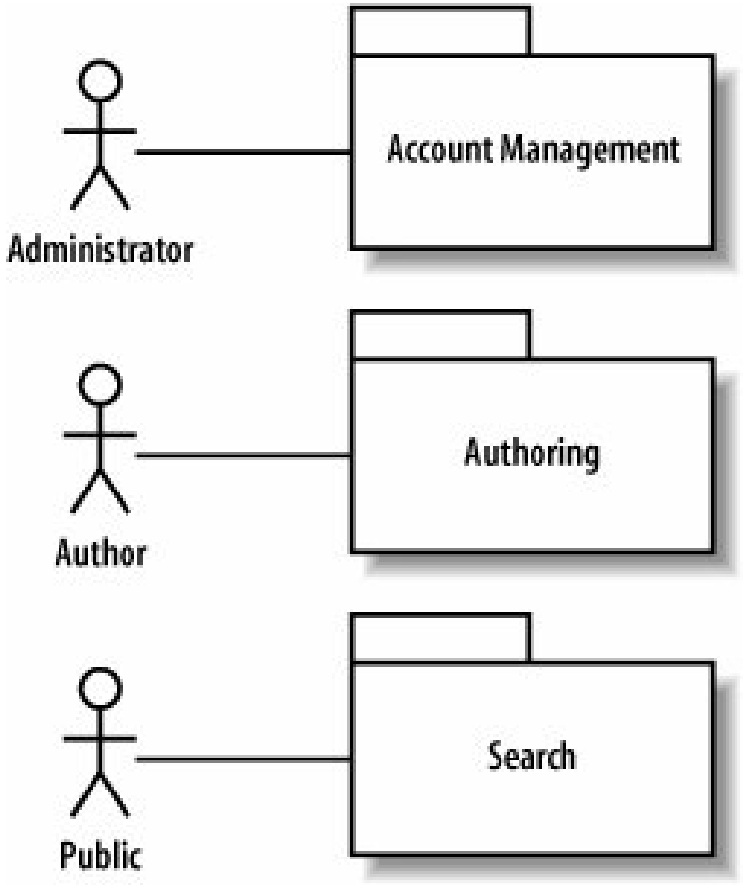


Figure 13-20. Packaging major use case groups within a CMS

Figure 13-21. Packages enable a higher level view of how actors interact with the system



See you …