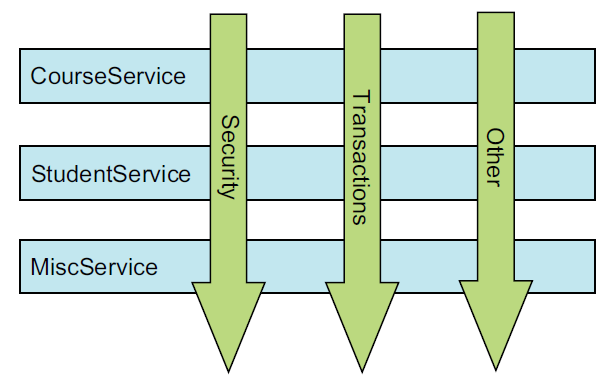
*Aspect-oriented Spring*

***What is aspect-oriented programming?***

* Aspects help to modularize cross-cutting concerns. In short, a cross cutting concern can be described as any functionality that affects multiple points of an application. Security, for example, is a cross-cutting concern, in that many methods in an application can have security rules applies to them



* This figure represents a typical application that’s broken down into modules. Each module’s main concern is to provide services from its particular domain. But each module also requires similar ancillary functionality, such as security and transaction management.
* A common object-oriented technique for reusing common functionality is to apply INHERITANCE OR DELEGATION. But inheritance can lead to a brittle object hierarchy if the same base class throughout an application and delegation can be cumbersome because complicated calls to the delegate object may be required.
* Aspects offer an alternative to inheritance and delegation that can be cleaner in many circumstances. With AOP, you still define the common functionality in one place, but you can declaratively define how and where this functionality is applied without having to modify the class to which you’re applying the new feature.
* Cross cutting concern can now be modularized into special classes called ***aspects****.* This has two benefits. First, the logic for each concern is in one place, as opposed to being scattered all over the code base. Second, your service modules are cleaner because they only contain code for their primary concern (or core functionality), and secondary concerns have been moved to aspects.