**Where we are:**

|  |  |
| --- | --- |
| Intro to MVC | Annotation and validation |
| Controllers and views | Identity - Authentication |
| Models and Entity Framework | Identity - Authorization |
| Forms and HTML helpers | EF Code-First Migrations, Deploying to Azure |
| Customizing Scaffolding | Dependency Injection, Unit Testing |

**Announcements / Code Review**  
Changing the topics for this week  
Code review for lab 7

Unit Testing

* What to test?
  + Business logic (in models, or in separate classes)
  + Controller methods
* Test directly against the database? No!
  + Add a test for a simple controller method: Search for book
  + Add a test for a more complex method: Create book
* Issues with testing against a real database
  + The database may not contain optimal data for doing certain tests. It’s not always practical/advisable to put data into a real data base during the arrange phase
  + The data can change over time and cause tests to fail.
  + The tests could cause unwanted changes to the real data.

Dependency Injection and the Repository Pattern

* Dependency injection – control dependency at run-time.
* For testing, use a fake version of the database
* Use different database contexts depending on whether we’re doing testing or not? No, the database context is too complex to simulate.
* Create a repository class that is an abstraction layer over the database context. It is typically put in a project folder named DAL (Data Access Layer).

References

* [MSDN: Verifying Code by Using Unit Tests](https://msdn.microsoft.com/en-us/library/dd264975(v=vs.120).aspx)
* [ASP.NET MVC Controller Dependency Injection for Beginners, by S. M. Ahasan Habib, 31 Dec 2013](http://www.codeproject.com/Articles/560798/ASP-NET-MVC-controller-dependency-injection-for-be)
* [Creating Unit Testable Applications in ASP.NET MVC - A Beginner's Tutorial, by Rahul Rajat Singh, 16 Apr 2013](http://www.codeproject.com/Articles/579035/Creating-Unit-Testable-Applications-in-ASP-NET-MVC)