Semantic Models

Domain / UI / Entity Data / Security / REST / Learning / Extensibility / Admin / Deployment Model

[Detailed Intro] Imagine a dev team has mastered core technologies and building on that solid foundation, now needs to design a next generation solution – how do they go about it? If you team is in this situation, this advanced course is what you need. Looking beyond the technologies, this course examines how to develop cutting-edge solutions that are of production quality, commercially competitive and feature all the "-abilities" your customers demand (scaleability, testability, manageability, usability, reliability, ...). The central idea is that software development revolves around a series of semantic models (domain, entity data, user interaction, security, admin, deployment, etc.) and these are grounded in a "single truth" of the source code which ensure all models work together.

For a new solution, we need to examine what is required to design and build each of these models so that the integrated end result delivers upon the expectations. Modern developer technologies allow us to be highly creative and more productive in how we go about designing web solutions. However, using them all together is somewhat of a challenge and this course carefully explores how to proceed.

Target Audience

Advanced software architects and senior software engineers who wish to design cutting-edge solutions using the latest design ideas.

Prerequisites

It is essential that attendees have a good all-round experience of the technologies they wish to use.

Contents of One-Day Training Course

Model Driven Development

Q: Assuming we know the very latest technologies and APIs, how do we go about designing next generation solutions?
A: Using a series of semantic models

Domain Model

Domain Driven Design
Handling complexity
Central role of the Domain Model
Evans' patterns for Domain Driven Design – Ubiquitous Language, Entities, ValueObjs, Intention-Revealing Interfaces, Services, Repositories, Layered Arch, ...

Entity Data Model

How we bind objects and relational data
Creating data models & abstractions
Using entities in other parts of app

User Interaction Model

Thinking past the widgets and visual layouts, we concentrate on what the user is really trying to achieve
Task-focused design
Making available functionality as needed
Fluid interaction; paradigm selection

Security Model

Deciding what needs to be protected and how; verifying security of solution
How to explain your solution's security model to others (e.g. security auditors)

REST Model

Representational state transfer can be used to selectively expose the Domain Model and Entity Data Model to remote clients

Deployment Model

Get the bits deployed is important (on average, 50% of support calls relate to this)
Treating deployment as first class feature

Administration Model

Different levels of users
Administrative features

Support Model

What happens when a user has a problem
How can software help with support
Instrumentation for support

Education Model

Often an afterthought, the education model is how users/admins learn to use the app
It is crucial to on-ramping of new users and the amount of (costly) support they need

Test Model

Unit tests, load tests, security tests, ...
Dependency injections and mocking
Managing the testing process

Intelligence Model

Extracting actionable results from various information stores and making it promptly/easily available

Extensibility Model

Exposing an API to enable third parties extend your solution – approaches

Reference Architecture

Exploring a reference architecture built on ideas covered in this course.

Project

Attendees will work in teams to develop slice of a suitable project (they choose)