DevExpress®

.NET Programmers and Architects
Here are your options

Oliver Sturm
olivers@devexpress.com

Oliver Sturm

- Training Director at DevExpress
- Experienced consultant, trainer, author, software architect and developer
- Microsoft C# MVP

- Contact: olivers@devexpress.com

Agenda

- Idea: talk about technology
- Client applications
- Data access
- Web applications
- Services
- Mobile
- Cloud

Client applications

- What's a client application today?
- What was it yesterday?

Client style UI

- WinForms
- WPF
- Silverlight
- WinRT
- … XAML based

Client style UI?

- HTML

Reasoning about client applications

- For: Utilizing local resources, integrating with other client (legacy?) systems, highest possible performance, more advanced toolsets available
- Against: restricted to local resources, maintenance and installation effort, tied in to target environment

Client UI Application Patterns

- MVVM

Reasoning about client UI application patterns

- For: separation of concerns -> advantages in testing, team work, better abstraction
- Against: potentially hard to adapt existing structures to pattern boundaries

Storing Data

- Relational databases
- NoSQL options
  - Key/value and column family stores
  - Document
  - Data analytics (e.g. MapReduce)

- Reasoning
  - For RDBMS
  - For NoSQL

Accessing Stored Data

- Direct access - ADO.NET?
- NoSQL APIs?

ORM

- Entity Framework
- Third parties
- Top Down vs Bottom Up
- DB independence

CQRS

- Command Query Responsibility Segregation
- Query and Command Models
- Consistency often not Strong - Eventual consistency?
- Back to data access basics?
- Conflicts with ORM?

Reasoning About Data Access

- For direct access: SQL level features (projection, …), speed in edge case scenarios, server-side command statements, NoSQL APIs
- For ORM: OO programmers' ease of use, potentially DB independent, time saver (only create one half of your data access solution manually), standard collection types support out of the box, advanced data layer features (caching, …)
- For CQRS: query/command split and its consequences

Web Applications

- What is a web application?

Server-side rendering on ASP.NET

- WebForms
- MVC

- Reasoning
  - For WebForms: client application event driven programming style, 3rd party component platform
  - For MVC: clean HTTP-like structure, code-focused developer environment, enforced application pattern

- For server-side rendering: centralized location for rendering, deployment and testing
- Against server-side rendering: client concerns taken into account, generally useful application pattern today

HTML “direct”

- … with little or no server rendering
- Can be delivered by a server

- Reasoning
  - For: more easily cacheable, saving server resources, ability to deploy CDNs, local deployment
  - Against: maintainability, localization more complex

Services

- Part of most architectural concepts
- SOA?
- “Real-time web?” SignalR? socket.io?

Web Services

- ASMX? WSE?
- WCF
  - Configurability - decision late binding
  - Complexity

- Reasoning
  - For: Interfacing with arbitrary 3rd party environments, WCF: external configuration, industry standards WSE
  - Against: overhead of standard data transfer formats (XML), complexity

Web API

- Based on ASP.NET MVC infrastructure
- JSON/XML content negotiation
- REST

- Reasoning
  - For: modern style (REST) services accessible by JavaScript and other 3rd party platforms
  - Against: more specific set of service use cases covered (compared to WCF, for instance)

Mobile

- Mobile support as a conceptual module
- Strategic platform?

“Native” Mobile

- Windows Phone SDK
  - More XAML!
- iOS SDK
- Android SDK

- Reasoning
  - For: best possible performance, direct hardware access
  - Against: tie-in to the vendor platform of choice, multiple investments, hardly any reuse of code

Mobile .NET

- Xamarin
- Reasoning:
  - For: reuse (C#) code while targeting UI platforms separately
  - Against: vendor-specific solution, appeal to certain developer groups

HTML/Hybrid

- HTML (5), JavaScript, CSS
- PhoneGap/Cordova or similar
- Cross-platform

- Reasoning
  - For: broadest reuse possible,
  - Against: performance, interfacing with hardware requires special module support

Cloud

- Deployment option
- Managed infrastructure

Cloud functionality

- Supplied services, vertical features
- Base platform functionality
  - Dynamic scalability
  - SLA

Legal considerations

- Locations
- Industry/governmental requirements

Cloud options

- Azure
- Amazon Web Services
- Third parties?

Cloud Reasoning

- For cloud: scalability, data safety, availability
- Against cloud: legal reasons, people afraid of others stealing their property, difficulty of refactoring legacy code

- For/against specific platforms:
  - Azure: good choice for Microsoft/VS- developers
  - AWS: most mature platforms, most choice of OS and related environments
  - AppEngine: restricted developer choices

Thank you

Please feel free to contact me about the content anytime.

olivers@devexpress.com