Managing & Deploying SQL Server Database Code

Benjamin Day
TRAINER | COACH | DEVELOPER

@benday www.benday.com

Overview

Databases & DevOps

Entity Framework

- .NET Framework
- .NET Core

SQL Server Data Tools (SSDT)

Lookup & Configuration Data

Automated Database Deployment

Managing database changes is the stickiest, most annoying problem in DevOps.

I want your life to be easy.

I want you to be able to focus.

I want you to be able to build, test, and deploy with ease.

I want you to be able to go faster and get more done.

I want you to be
awesome at DevOps.

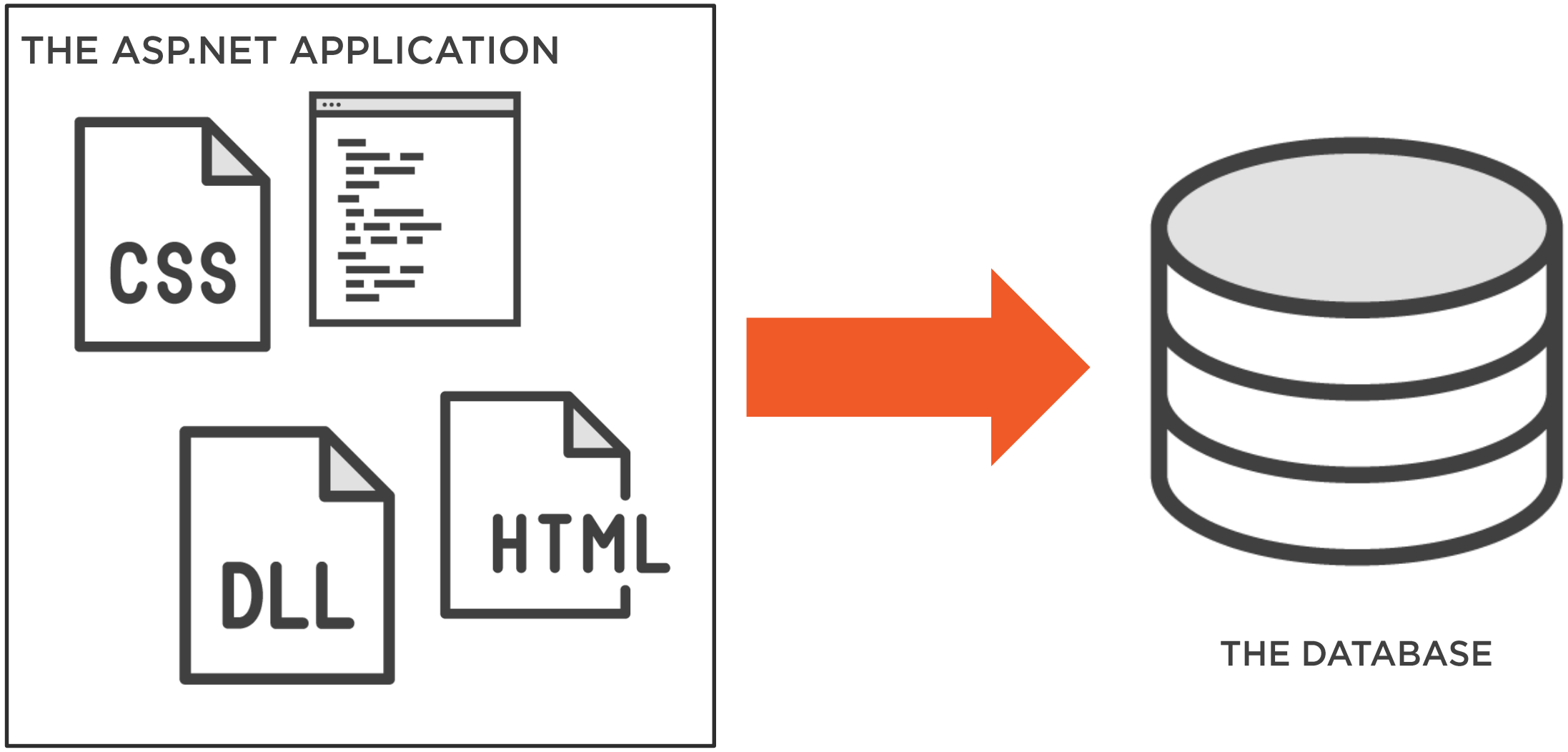
Database change management is the #1 blocker for DevOps “awesomeness.”


EAT YOUR
VEGETABLES!

Here’s the problem.

Approximately 99.99% of applications write to a database.

Typical ASP.NET Application



THE DATABASE

“The Database” isn’t really a thing

It’s a lot of little things…

- Tables
- Primary Keys
- Foreign Keys
- Stored Procedures

…and then also the data

Deploying the Application

ASP.NET

Delete the existing assets

Copy the new assets

Database Schema

Don't delete the existing data

Run an update script

Update script adds, updates, deletes all the schema objects

Run a script that modifies any required lookup data

Where does this script come from?

If you use version control, you probably only have the update scripts.

Update scripts don’t tell you what you ultimately want in your database.

It only says how to get there.

Branching with database update scripts is tricky.

Branching with Database & Application Code

Database code & application code MUST be branched together

- They have to be in the same folder structure
- They have to live in the same branch

If you only have database update scripts...

...then how do you decide what order the updates need to be run?

- How do you merge that?

The application code needs
to match
the database code.

You need a way to automate your database deployments.

Version control → builds → releases

You need a way to automate application and database deployments at the same time.

Next up:
What are your options?

Databases & DevOps: What are your options?

Schema Management & Deployment Options

Entity Framework Migrations

Entity Framework Core Migrations

RedGate's tools

SQL Server Data Tools (SSDT)

Options

RedGate's tools

- http://www.red-gate.com/
- $1,895 - $2,945 (USD) / license

Entity Framework Migrations

- Microsoft
- .NET Framework or .NET Core
- Basic schema management

SQL Server Data Tools (SSDT)

- Microsoft
- Enterprise schema management

Options

Entity Framework Migrations	SQL Server Data Tools (SSDT)
Integrated with your data access code	Project type in Visual Studio
Make database changes then create a “migration”	Describe each object in the database by creating individual *.sql scripts
Migration = set of database changes	Model-based
Move from a known schema state to another known schema state	It generates database upgrade scripts based on the model vs. the target
Great for table-centric schema management	Can manage pretty much anything in SQL Server
Less than awesome for stored procedures, functions, indexes, data, etc.	More than just schema management

SQL Server Data Tools

- SQL Server code under source control
- Compile-time checking of your database code
- Database comparisons
- Data comparisons
- Unit tests
- IntelliSense
- Refactoring tools
- SQL code analysis

Entity Framework Migrations or SQL Server Data Tools (SSDT)?

Both are good options

Do the developers have complete control over the database?

- Any pesky DBAs making changes?

Are you fine with default SQL Server storage options?

Do you have cross-database dependencies?

Entity Framework Migrations or SQL Server Data Tools (SSDT)?

EF Migrations

"Hey, man. I'm not trying to boil the ocean here. Good enough is good enough."

Schema mostly about tables

Limited or no stored procedures, functions, triggers, etc.

Not real picky about how things are created (data types, foreign key names, file storage options, etc.)

One database

SSDT

"I need control! Life is complex and my deployment environment is complex."

You use SQL Server to its fullest

You need performance tuning

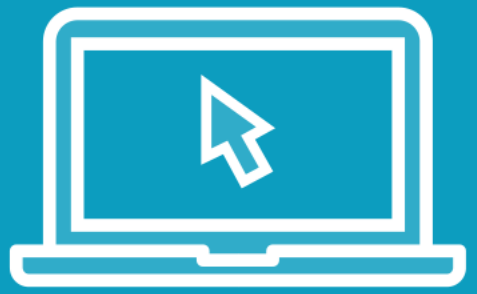
File storage is not the same in every environment

You need to manage permissions

You've got cross-database and/or cross-server dependencies

Next up:
Entity Framework Migrations with EF for .NET Framework

Demo



Entity Framework for the .NET Framework

- "Regular .NET"

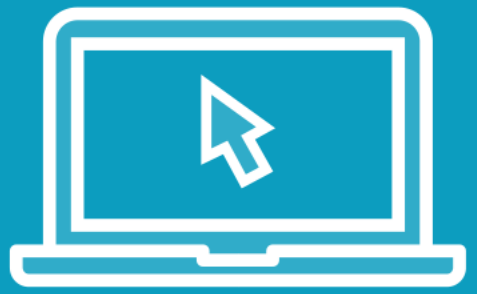
Code-first EF

Create migrations

Deploy migrations

Next up:
Entity Framework Core Migrations

Demo



Entity Framework Core

- .NET Core

Code-first EF

Create migrations

Deploy migrations

Next up:
Database development with SSDT

Demo

Create an SSDT project

Import an existing database

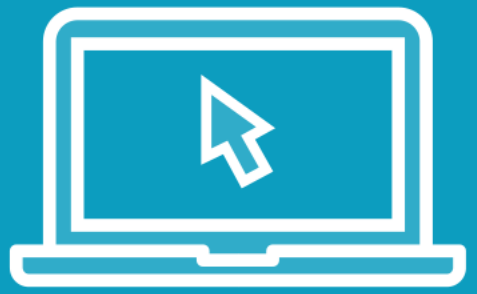
Build

Refactor

Tour of *.dacpac files

Next up:
Schema comparisons with SSDT

Demo



SSDT Schema Comparisons

Compare & update databases with SSDT

- Deploy

Compare a database to an SSDT project and import changes

- “Rogue DBAs”
- Changes made outside of SSDT & source control

Next up:
Managing lookup data with SSDT

Data in SSDT

Lookup Data

Configurable lists of values

Examples

- Lists of states
- Employee titles
- Server names / URLs

Demo

Manage lookup data

Create scripts to include lookup data in the project & keep it up to date

Next up:
Incremental database deployments from the command line

Deploy Database Updates from the Command Line

DevOps is all about automation

Deploy from command line →
Automated script

Automated script →
Automated builds

Automated builds →
Automated deployments

SqlPackage.exe

- Deploys an SSDT *.dacpac

C:\Program Files (x86)\
Microsoft Visual Studio\2017\
Enterprise\Common7\IDE\Extensions\
Microsoft\SQLDB\DAC\130

Demo

Deploy database changes using SqlPackage.exe

Summary

Databases & DevOps

→ Automated Database Deployment

Entity Framework

- .NET Framework
- .NET Core

SQL Server Data Tools (SSDT)

- Lookup & Configuration Data
- Deploy using SqlPackage.exe

Next up:
Feature Flags