FileMaker Plugin Entwicklung

Christian Schmitz

FileMaker Plugin

- FileMaker Plugin Schnittstelle
- Für FileMaker 7 und neuer
- Erlaubt eigene Funktionen in C, C++, Objective-C & Objective-C++ zu definieren

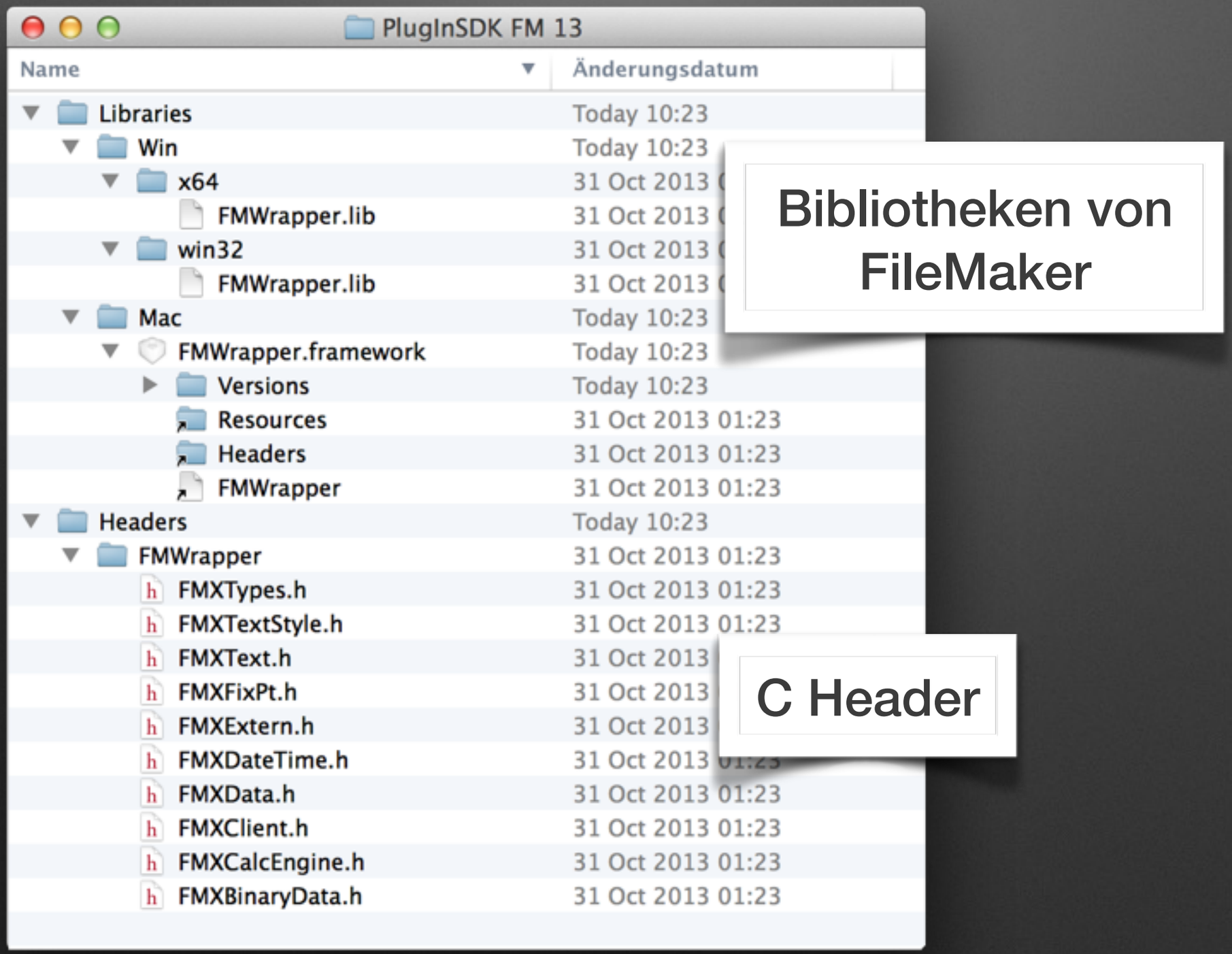
Plugins können nicht

- Skriptschritte definieren/ändern
- Zugriff nehmen auf WebDirect Seiten
- Auf FileMaker Go laufen
- Layouts ändern
- Textausrichtung im Formatierten Text
- Extended Data Source anbieten

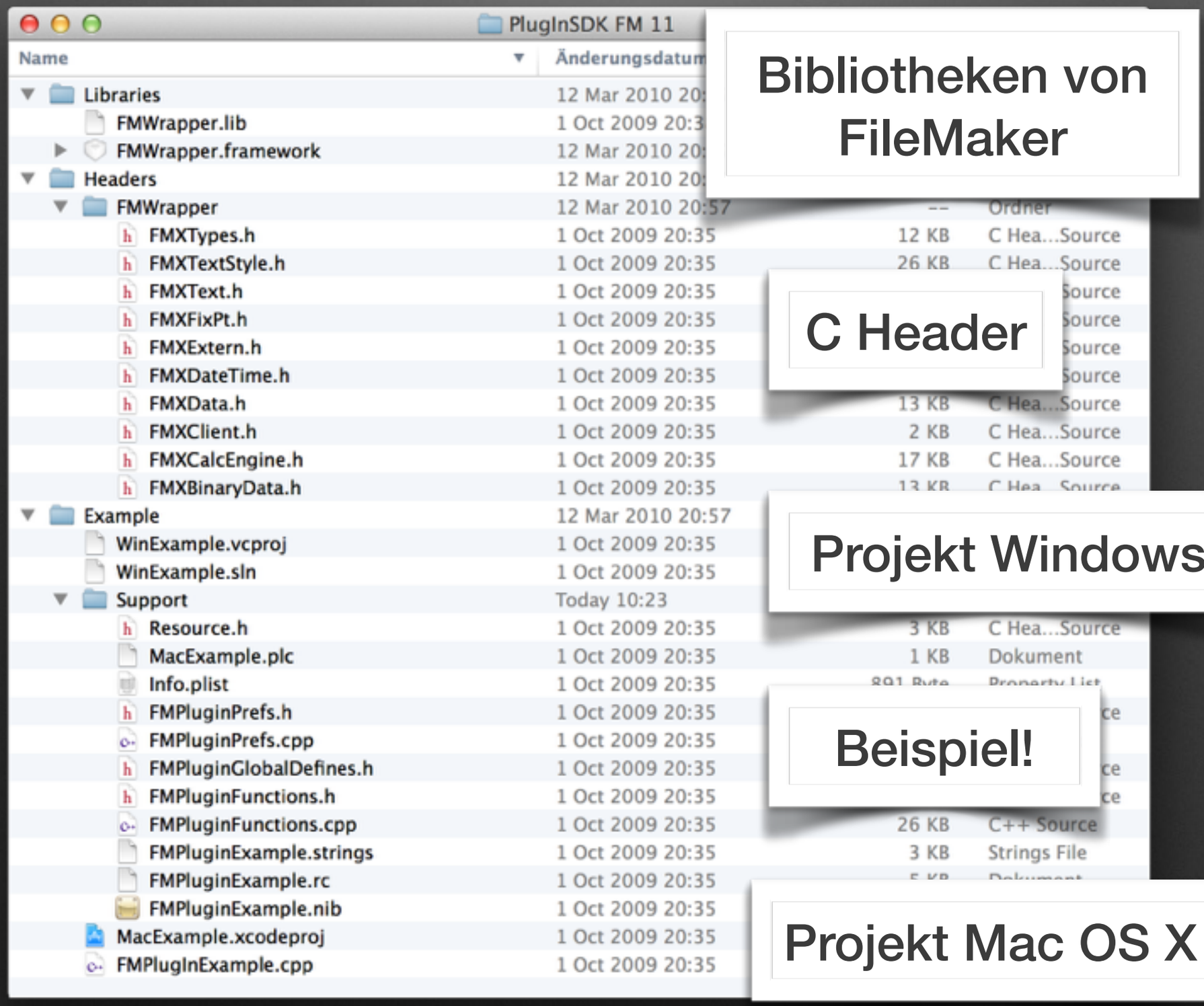
Benötigt

- Plugin SDK von FileMaker
- Download auf filemaker.com
  http://www.filemaker.de/support/docs/fm_plugin_sdk.13.0.1.zip
- Filemaker Plugin Template von 24U
  http://content.24usoftware.com/PlugInTemplate
- Visual Studio für Windows
- Xcode für Mac OS X

FileMaker SDK 13



FileMaker SDK 11



Dokumentation?

- Teilweise im C++ Quelltext

```
            // These three methods expect null terminated strings.
            inline void Assign ( const char *s, Encoding encoding = kEncoding_Native );
inline void AssignWide ( const wchar_t *s );
inline void AssignUnicode ( const uint16 *s );

            inline void AssignWithLength ( const char *s, uint32 strlength, Encoding encoding = kEncoding_Native )
inline void AssignWideWithLength ( const wchar_t *s, uint32 strlength );
inline void AssignUnicodeWithLength ( const uint16 *s, uint32 strlength );

            // Buffer size is in bytes, position and size are in Unicode characters, encoded bytes copied to buffe
inline void GetBytes ( char *charbuffer, uint32 buffersize, uint32 position = 0, uint32 size = kSize_End, Encoding

            // Position and size are in Unicode characters, characters copied to unicodebuffer without any termina
inline void GetUnicode ( uint16 *unicodebuffer, uint32 position, uint32 size ) const;

inline void SetText ( const Text &other, uint32 position = 0, uint32 size = kSize_End );
inline void AppendText ( const Text &other, uint32 position = 0, uint32 size = kSize_End );
inline void InsertText ( const Text &other, uint32 position );
inline void DeleteText ( uint32 positionToDelete, uint32 sizeToDelete = kSize_End );

inline uint32 Find ( const Text &other, uint32 position ) const;
inline uint32 FindPrev ( const Text &other, uint32 position ) const;
inline uint32 FindIgnoringCase ( const Text &other, uint32 position ) const;
inline uint32 FindPrevIgnoringCase ( const Text &other, uint32 position ) const;
```

Dokumentation?

- Teilweise im C++ Quelltext
- Im Internet
  - http://fmplugins.idma.co.nz/index.php/Main_Page
- Open Source Plugins lesen
- Internet Foren
- Viel probieren

Was ist ein Plugin?

- Programm Bibliothek mit Metadaten
- Mac
  - Bundle, 32/64bit
  - Dateiendung .fmplugin
- Windows
  - Zwei DLLs: eine 32bit, eine 64bit
  - Dateiendungen fmx/fmx64

API

- Klassen für Parameter/Ergebnis
- Evaluate
- StartScript
- ExecuteFileSQL & ExecuteSQL

Klassen

- Data
- DataVect
- BinaryData (Container)
- Text
- FixPt (Zahlen)
- DateTime
- CharacterStyle (für formatierten Text)
- Locale

Plugin Configuration

- Plugin name "MBS Plugin"
- Plugin eindeutige 4 Zeichen ID "MBSP"
- Version
- Infotext
- Funktionenliste

Funktionen

- Plugin definiert Funktionen
- C Funktion, e.g. MBS
- ID (laufende Nummer)
- FileMaker Deklaration, e.g. MBS(FunctionName)
- Mindeste und maximale Anzahl Parameter
- Flags: kMayEvaluateOnServer, kDisplayInAllDialogs

Leere Funktion

```
FMX_PROC(fmx::errcode) Test
(short funcId,
 const fmx::ExprEnv& environment,
 const fmx::DataVect& dataVect,
 fmx::Data& results)
{
  return 0;
}
```

Funktionparameter in C

- funcID: ID aus Funktionsdefinition
- environment: Aktuelle Umgebung für Evaluate, ExecuteSQL, StartScript
- dataVect: Parameterliste aus FileMaker

Funktionsergebnis in C

- results: Variable für Ergebnis der Funktion
  - Ergebnis der Funktion hier zuweisen
  - Text, Zeitstempel, Zahlen, Container
- errcode: Fehlercode für Rückgabe
  - Hole (LetzteFehlerNr)

Hello World Funktion

```
{
  fmx::TextAutoPtr resultText;
  resultText->Assign("Hello World");
  return result.SetAsText( *resultText,
                           result.GetLocale() );
}
```

AutoPtr

- C++ Memory Management
- const &fmx::Text &text
  - Referenz von FileMaker für existierendes Objekt
- fmx::TextAutoPtr
  - Neues Objekt zum Ändern
  - Wird automatisch freigegeben, wenn Funktion endet.

AutoPtr

- TextAutoPtr
- FixPtAutoPtr
- DateTimeAutoPtr
- BinaryDataAutoPtr
- DataAutoPtr
- ColorAutoPtr
- CharacterStyleAutoPtr
- QuadCharAutoPtr
- LocaleAutoPtr
- DataVectAutoPtr
- RowVectAutoPtr
- ExprEnvAutoPtr

Addieren Funktion

```
const fmx::FixPt &Zahl1 = dataVect.AtAsNumber(0);
const fmx::FixPt &Zahl2 = dataVect.AtAsNumber(1);

fmx::FixPtAutoPtr Ergebnis;
num->AssignFixPt( Zahl1 );
num->Add( Zahl2, *Ergebnis );

results.SetAsNumber(*Ergebnis);
return 0;
```

Großbuchstaben Funktion

```
const fmx::Text &text = dataVect.AtAsText(0);

fmx::TextAutoPtr resultText;
resultText->SetText(text);
resultText->Uppercase();

return results.SetAsText( *resultText,
                          results.GetLocale() );
```

- Mit den Text Funktionen vom Plugin SDK

Großbuchstaben Funktion

```
char buffer[200] = {0};
const fmx::Text &text = dataVect.AtAsText(0);
text.GetBytes(buffer, sizeof(buffer), 0,
      fmx::Text::kSize_End, fmx::Text::kEncoding_UTF8);

char* b = buffer;
while (*b)
{
    *b = toupper(*b);
    b++;
}

fmx::TextAutoPtr resultText;
resultText->Assign(buffer, fmx::Text::kEncoding_UTF8);

return results.SetAsText( *resultText,
                          results.GetLocale() );
```

Anzahl Parameter

```
if (dataVect.Size() < 2)
{
    return kErrorParameterMissing; // 102
}
```

SDK Kompatibilität

- Neuere SDKs definieren neue SDK Funktionen
- FM13 SDK bei Windows inkompatibel zu FM12.
- Weak Linking hilft.

Trying 24U Template

- Template laden
- Xcode & Visual C Projekte kopieren und aktualisieren
- eventuell kleine Fixes für aktuelles Mac/Win SDK
- Eigene PLUGIN_ID_STRING festlegen!
- Eigene Funktion definieren
- Probieren & Testen

Fragen?

- Demo

Vielen Dank unseren Sponsoren



Danke für das Bewerten dieses Vortrages