# KAPITOLA 5

# Tabulky

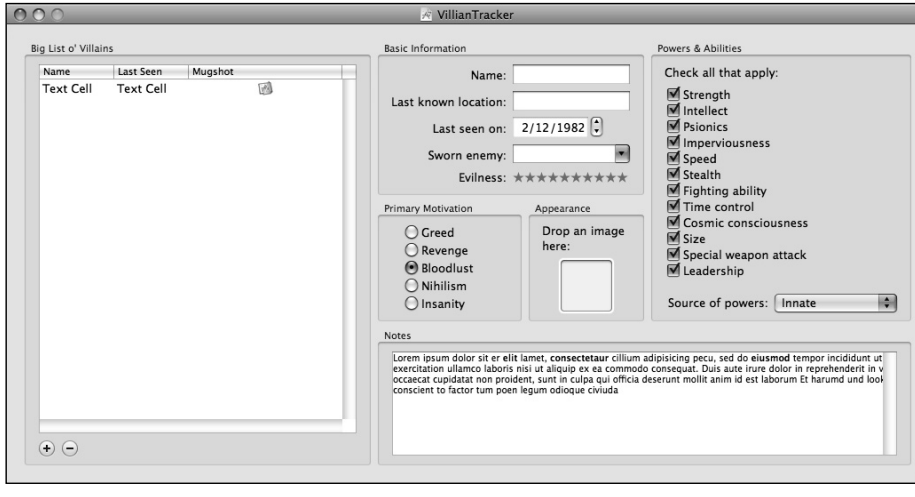
Ve 4. kapitole jste se blíže seznámili s některými z nejběžnějších komponentů grafického uživatelského rozhraní, počínaje tlačítky a jednoduchými poli pro zadávání vstupu až po plnohodnotné textové editory. Zatím jsme však nemluvili o jedné z největších a nejsložitějších tříd prostředí Cocoa, o třídě `NSTableView` – rámci tabulky. V této kapitole se naučíte třídu `NSTableView` používat ke zobrazování dat celé kolekce komponentů, reagovat na změnu výběru v tabulce provedenou klepnutím na některý z jejích řádků a upravovat její hodnoty.

S rámcem tabulky se naučíte pracovat při rozšiřování aplikace VillainTracker, kterou jste začali psát v kapitole 4. Nová verze aplikace, kterou vytvoříte v této kapitole, bude uchovávat pole padouchů, zobrazovat je v tabulce a umožňovat uživateli úpravu hodnot atributů konkrétních padouchů, které bude moci vybrat klepnutím na jejich záznam v tabulce. Začnete rozšířením rozhraní třídy `VillainTrackerAppDelegate` v prostředí Xcode a přidáte do něj pole padouchů, několik nových outletů pro propojení s novým rámcem tabulky a se samotným oknem a několik nových metod akcí pro přidávání a mazání padouchů. Dále připravíte návrh uživatelského rozhraní v nástroji Interface Builder a následně se vrátíte zpět do prostředí Xcode, kde implementujete změny řídicí třídy pro obsluhu tabulky. Konečnou podobu aplikace si můžete prohlédnout na obrázku 5.1.

# Příprava třídy VillainTrackerAppDelegate na více padouchů

V prostředí Xcode otevřete projekt, který jste vytvořili při četbě 4. kapitoly, a najděte soubor *VillainTrackerAppDelegate.h*, ve kterém upravíte rozhraní třídy `VillainTrackerAppDelegate` tak, aby třída mohla obsluhovat více padouchů. Protože budete udržovat seznam padouchů, vytvoříte instanci třídy `NSMutableArray` s názvem `villains`, která bude všechny padouchy obsahovat. Dále přidáte outlet `villainsTableView`, abyste mohli nějak přistupovat k instanci třídy `NSTableView`, jejímž prostřednictvím budete seznam padouchů prezentovat. A když už v tom budete, přidáte další outlet, `window`, jehož prostřednictvím propojíte instanci třídy `NSWindow`, která obsahuje všechny prvky uživatelského rozhraní aplikace. Tento outlet využijete o něco později.

Pro úplnost přidáte pro novou instanční proměnnou `villains` také deklaraci `@property`, aby mohl ostatní zdrojový kód (včetně zdrojového kódu v implementaci třídy `VillainTrackerAppDelegate`) k této hodnotě jednoduše a bezpečně přistupovat a upravovat ji.




**Obrázek 5.1** Hotové okno aplikace

Poslední úpravou zdrojového kódu v souboru *VillainTrackerAppDelegate.h* pak bude přidání deklarací dvou nových metod akcí `newVillain:` a `deleteVillain:`. Následující výpis zdrojového kódu představuje kód v souboru *VillainTrackerAppDelegate.h* po provedení uvedených změn (nové řádky zdrojového kódu jsou zvýrazněné tučným písmem):

```
#import <Cocoa/Cocoa.h>

@interface VillainTrackerAppDelegate : NSObject {
  IBOutlet NSTextField *nameView;
  IBOutlet NSTextField *lastKnownLocationView;
  IBOutlet NSDatePicker *lastSeenDateView;
```

```
  IBOutlet ComboBox *swornEnemyView;
  IBOutlet NSMatrix *primaryMotivationView; // matice přepínačů
  IBOutlet NSMatrix *powersView; // matice zaškrtnutelných políček
  IBOutlet NSPopUpButton *powerSourceView;
  IBOutlet NSLevelIndicator *evilnessView;
  IBOutlet NSImageView *mugshotView;
  IBOutlet NSTextView *notesView;
  IBOutlet NSTableView *villainsTableView;
  IBOutlet NSWindow *window;

NSMutableDictionary *villain;
NSMutableArray *villains;
}

@property (retain) NSMutableDictionary *villain;
@property (retain) NSMutableArray *villains;

- (IBAction)takeName:(id)sender;
- (IBAction)takeLastKnownLocation:(id)sender;
- (IBAction)takeLastSeenDate:(id)sender;
- (IBAction)takeSwornEnemy:(id)sender;
- (IBAction)takePrimaryMotivation:(id)sender;
- (IBAction)takePowerSource:(id)sender;
- (IBAction)takePowers:(id)sender;
- (IBAction)takeEvilness:(id)sender;
- (IBAction)takeMugshot:(id)sender;
- (IBAction)newVillain:(id)sender;
- (IBAction)deleteVillain:(id)sender;

@end
```

Nyní přidejte do souboru *VillainTrackerAppDelegate.m* odpovídající direktivu `@synthesize` a dokončete tak definici `@property` atributu `villains`. Do souboru budete také muset přidat dvě nové implementace metod (prozatím pouze jejich prázdné schránky) pro obě metody akcí. Přidejte tedy do sekce `@implementation` třídy `VillainTrackerAppDelegate` následující zdrojový kód:

```
@synthesize villains;
- (IBAction)newVillain:(id)sender {}
- (IBAction)deleteVillain:(id)sender {}
```

Všechny nové součásti rozhraní třídy `VillainTrackerAppDelegate` a pahýly nových metod jsou na svém místě, takže si můžete kompilací aplikace ověřit, že jste neudělali žádnou chybu, a přesunout se do nástroje Interface Builder, kde upravíte grafické uživatelské rozhraní aplikace, abyste uvolnili potřebné místo pro tabulku.

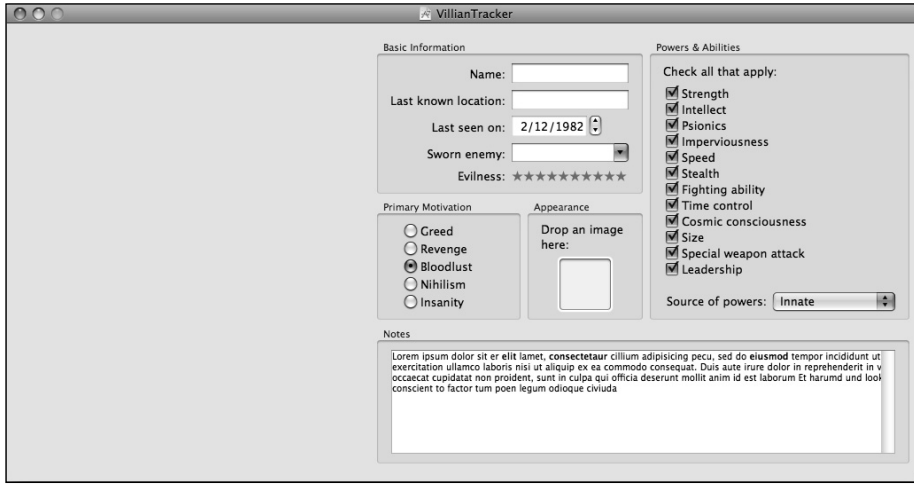
# Uvolnění místa pro rámec tabulky

V navigačním panelu prostředí Xcode poklepejte na soubor *MainMenu.xib* a otevřete jej tak v nástroji Interface Builder. V nástroji Interface Builder nyní zvětšíte okno aplikace, přidáte do něj náhled tabulky a několik tlačítek a upravíte vlastnosti automatické

změny rozměrů instancí třídy `NSBox`, aby se při zvětšování rámce tabulky v obou směrech ostatní rámečky posouvaly v okně.

Začněte změnou rozměrů okna a zvětšete jeho šířku o zhruba 400 pixelů (zhruba o půlku jeho aktuální šířky), ale neroztahujte jej do výšky. Při navrhování uživatelského rozhraní aplikace se budete řídit západní orientací textu zleva doprava a rozmístíte objekty v okně tak, aby výběr položky v levé části okna (v tabulce) specifikoval data, která se zobrazí v jeho pravé části (ve všech ostatních rámcích vzhledu), takže rovnou přetáhněte všechny existující objekty uživatelského rozhraní do pravé části okna. Pro lepší představu si můžete prohlédnout výslednou podobu našeho okna na obrázku 5.2.

Nyní otevřete okno knihovny, zadejte do jeho vyhledávacího pole řetězec „tableview“ a přetažením do okna vytvořte v okně instanci nalezené třídy `NSTableView`. Výchozí rozměry rámce tabulky jsou o něco menší než volný prostor, do kterého jste jej vložili, ale jejich úpravou se zatím nezabývejte, na to dojde až za chvíli; prozatím nechte rámec tabulky ležet uprostřed volného místa v okně.




**Obrázek 5.2** Příprava okna na vložení rámce tabulky

První úpravou rámce tabulky bude konfigurace sloupců. Nová tabulka má ve výchozím nastavení dva sloupce, ale vy potřebujete sloupce tři, abyste mohli v tabulce zobrazovat jména padouchů, data jejich posledního spatření a jejich fotografie. Chcete-li do tabulky přidat nový sloupec, musíte se „provrtat“ několika vrstvami rámců vzhledu. Otevřete inspektor atributů, který vám při tom bude pomáhat, protože vždy zobrazí několik informací o vybraném objektu, zejména název jeho třídy. Po prvním klepnutí na rámec tabulky v okně inspektor atributů zobrazí atributy rámce s posuvníkem (*Scroll View Attributes*). Je tomu tak proto, že rámec tabulky, který jste přetáhli do okna z knihovny, je obsažený v instanci třídy `NSScrollview`. Klepněte znovu a inspektor zobrazí atributy rámce tabulky (*Table View Atributtes*); už se blížíte! Nyní klepněte do

volného místa v tabulce (pod hlavičky sloupců) a inspektor zobrazí atributy sloupců tabulky (*Table Columns Attributes*), což přesně potřebujete. Když se vám tedy konečně podařilo označit jeden ze sloupců tabulky, vyberte z nabídky **Edit** položku **Duplicate** a do tabulky přibude další sloupec.

Sloupce tabulky nyní potřebujete nastavit tak, aby zobrazovaly jména, data posledního spatření a fotografie uložené v objektech padouchů. Uděláte to tak, že každému z nich přiřadíte identifikátor odpovídající názvu atributu, který chcete ve sloupci zobrazit. Později vám ukážeme, jak využít tento identifikátor ve zdrojovém kódu ke snadné přípravě zobrazovaných hodnot. Vyberte první sloupec a v inspektoru atributů vložte do položky jeho identifikátoru řetězec „`name`“. A když už jste v tom, vložte také rovnou do položky *Title* řetězec „Jméno“ – tento řetězec se zobrazí v hlavičce sloupce. Stejný postup zopakujte pro druhý sloupec a nastavte jeho identifikátor na hodnotu „`lastSeenDate`“ a text hlavičky na „Naposledy spatřen/a“; zbývá ještě třetí sloupec, jehož identifikátor nastavte na hodnotu „`mugshot`“ a text hlavičky na hodnotu „Fotografie“.

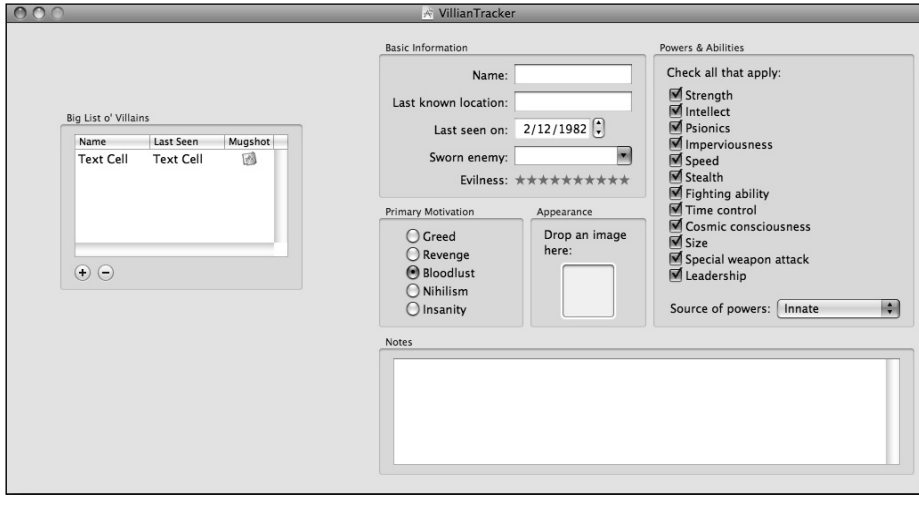
Třetí sloupec pak vyžaduje ještě další úpravy, protože bude zobrazovat obrázky, a ne text. Prostředí Cocoa však naštěstí obsahuje třídu, díky které je tato úprava velmi snadná. Možná si vzpomenete, že když jste ve 4. kapitole používali instanci třídy `NSMatrix` plnou ovládacích prvků, byly tyto ovládací prvky ve skutečnosti instancemi podtřídy třídy `NSCell`, která je jednodušší než třída `NSView` nebo `NSControl`. Třída `NSTableView` také vykresluje svůj obsah s pomocí buněk a ve výchozím nastavení používá třídu buněk `NSTextFieldCell` pro zobrazování a úpravu textových řetězců. Vy tuto třídu nyní změníte na `NSImageCell`, takže bude moci třetí sloupec zobrazovat obrázky. Vraťte se ještě jednou do okna knihovny a zadejte do vyhledávače řetězec „imagecell“ – výsledkem vyhledávání bude třída `NSImageCell`. Přetáhněte ji do rámce tabulky do prostoru sloupce fotografií. Sloupec se zvýrazní, a až dojedete kurzorem na místo, kde můžete položku upustit, zobrazí se u kurzoru myši malé zelené znaménko plus. Uvolněte tlačítko myši a je to! Sloupec je nyní připravený zobrazovat obrázky.

Nyní přidáte pár tlačítek – jedno pro přidání nového padoucha na seznam a jedno pro odstranění označeného padoucha ze seznamu. V okně knihovny zadejte do vyhledávače „button“ a zobrazí se neuvěřitelně rozsáhlý seznam různých druhů tlačítek. Někdo ve společnosti Apple má opravdu rád tlačítka. Navzdory rozdílům, které mezi nimi panují, jsou však téměř všechna tlačítka na seznamu instancemi třídy `NSButton`, která pak v podstatě dělá vždy stejnou věc. Některá tlačítka jsou přednastavená do různých režimů, aby pracovala jako zaškrtnutelná políčka nebo rozbalovací trojúhelníčky, ale jejich funkce je jinak stejná: když na ně někdo klepne, zavolají metodu akce.

Vyberte jedno z prvních několika tlačítek na seznamu (třeba *Push Button* nebo *Rounded Rect Button*), přetáhněte jej do okna aplikace a umístěte jej pod rámec tabulky. Toto tlačítko bude sloužit k přidávání padouchů. Avšak místo toho, abyste mu přiřadili popisek „Přidat padoucha“, použijte jeden ze zabudovaných obrázků prostředí Cocoa, aby tlačít-

ko vypadalo jako tlačítka pro přidávání položek v jiných aplikacích. Dokud je tlačítko vybrané, otevřete inspektor atributů a najděte v něm rozbalovací seznam *Image*. Klepněte na malý trojúhelníček, který je napravo od něj, a začněte psát řetězec „nsaddtemplate". Až se zadávaný text nahradí názvem „NSAddTemplate", stiskněte klávesu **RETURN** a na vašem novém tlačítku se zobrazí znaménko plus. Roztáhněte tlačítko horizontálně tak, aby se do něj obrázek vešel, a poté jej duplikujte (stiskem kláves ⌘+**D**). Nové tlačítko označte, vraťte se zpět do rozbalovacího seznamu *Image* v inspektoru atributů a začněte psát řetězec „nsremovetemplate" – stejným způsobem jako obrázek tlačítka plus tak získáte obrázek pro tlačítko minus.

Nyní tlačítko pro přidávání padouchů uchopte myší, přetáhněte jej pod spodní okraj rámce tabulky a s pomocí modrých pomocných čar zarovnejte s jeho levým okrajem a do doporučené vzdálenosti od okraje spodního. Vpravo vedle něj pak umístěte tlačítko pro odebírání padouchů. Poté označte obě tlačítka i rámec tabulky, vyberte položku **Box** z podnabídky **Embed Objects** nabídky **Layout** a nastavte popisek nového rámečku na hodnotu „Velký seznam padouchů". Všechny nové rámce jsou nyní součástí instance třídy `NSBox`, stejně jako všechny ostatní ovládací prvky (pouze ještě musíte upravit její rozměry a zarovnat ji s ostatními rámečky). Na obrázku 5.3 si můžete prohlédnout, jak by mělo vaše okno v tuto chvíli vypadat.




**Obrázek 5.3** Rámec tabulky a tlačítka jsou na svém místě, ale ještě je třeba upravit velikost jejich rámečku

# Vyladění automatické změny rozměrů

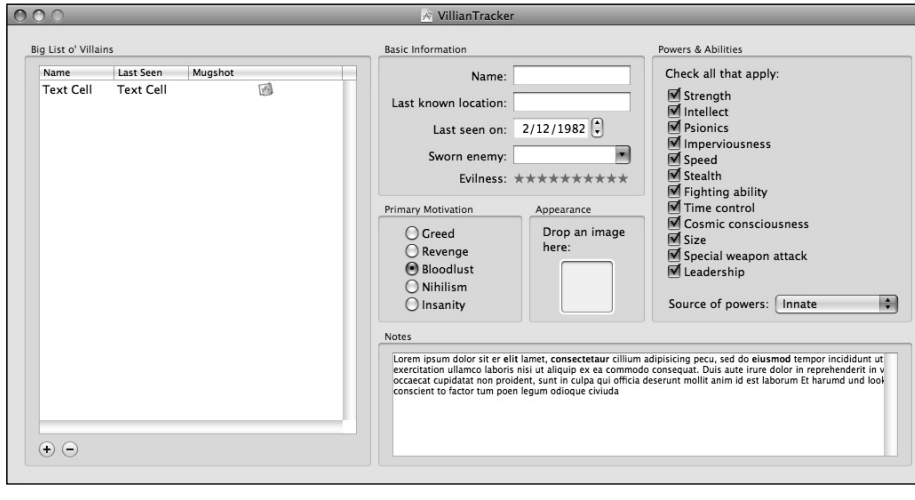
Nyní přišel čas upravit rozměry rámce tabulky a jeho rámečku a vyplnit volné místo v okně. Uchopte myší nový rámeček a umístěte jej tak, aby byl jeho spodní okraj zarovnaný se spodním okrajem rámečku *Poznámky* a jeho levý okraj v odpovídající vzdále-

nosti od levého okraje okna. Poté uchopte jeho pravou horní rukojeť pro změnu rozměrů a táhněte nahoru a doprava, dokud nebude horní okraj rámečku ve správné vzdálenosti od horního okraje okna a jeho pravý okraj ve správné vzdálenosti od ovládacích prků, které jsou v pravé části okna. Jako obvykle vám při tom pomohou modré pomocné čáry.

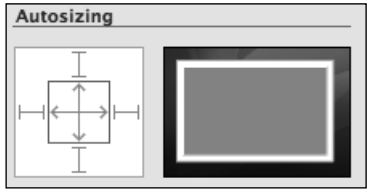
Poté vyberte rámec tabulky v rámečku a prostřednictvím pravé horní rukojeti upravte jeho rozměry, aby vyplnil volné místo v rámečku. Výslednou podobu okna si můžete prohlédnout na obrázku 5.4.

Ve 4. kapitole jste nastavili rámečky tak, aby se rámeček *Poznámky* zvětšoval ve všech směrech spolu s oknem a všechny ostatní rámečky zůstaly přilepené k jeho levému hornímu rohu. Nyní je však v levé části okna rámec tabulky, a proto je třeba nastavit vlastnosti automatické změny rozměrů tak, aby se ve všech směrech spolu s oknem zvětšoval rámec tabulky, rámeček *Poznámky* se zvětšoval pouze vertikálně (a uchovával si stejnou šířku) a ostatní ovládací prvky se „lepily" k pravému hornímu rohu okna.

Začněte výběrem rámce tabulky. Otevřete inspektor rozměrů (⌘+**3**) a v sekci *Autosizing* postupně klepněte na všechny červené přerušované čáry (vně i uvnitř černého čtverce), aby se z nich staly čáry plné. Plné čáry uvnitř černého čtverce nastaví rámec tabulky tak, aby se zvětšoval ve všech směrech, a plné čáry vně čtverce zajistí, že si zachová vzdálenost svých vnějších okrajů od okrajů rodičovského rámce (rámce, který jej obsahuje). Jinými slovy, rámec tabulky se bude zvětšovat ve všech směrech spolu se svým rodičovským rámcem. Poté vyberte klepnutím na jeho popisek samotný rámeček *Velký seznam padouchů* a nastavte jeho vlastnosti automatické změny rozměrů stejným způsobem, takže se bude zvětšovat a zmenšovat ve všech směrech spolu s oknem. Jak tato konfigurace vypadá v inspektoru rozměrů, si můžete prohlédnout na obrázku 5.5.

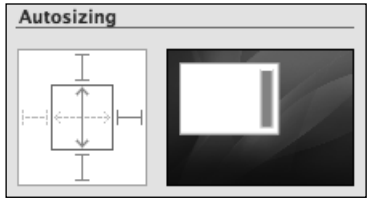



**Obrázek 5.4** Cílem je tato podoba okna



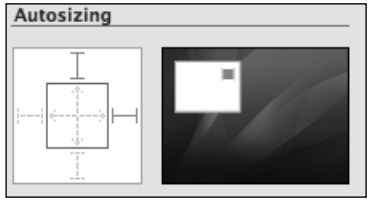

**Obrázek 5.5** Toto nastavení automatické změny rozměrů rámce tabulky a obklopujícího rámečku umožní jejich zvětšování a zmenšování ve všech směrech spolu se změnou rozměrů okna

Nyní vyberte rámeček *Poznámky* a změňte jeho vlastnosti nastavené v sekci *Autosize* tak, aby se jeho velikost mohla měnit pouze vertikálně a jeho levý okraj byl flexibilní (a umožňoval změnu rozměrů rámce tabulky). Dosáhnete toho tak, že přepnete vnitřní horizontální čáru a levou čáru okraje na přerušovanou, zatímco ostatní čáry ponecháte plné – viz obrázek 5.6.




**Obrázek 5.6** Tuto konfiguraci automatické změny rozměrů použijte pro rámeček Poznámky

Nyní přišel čas nastavit vlastnosti automatické změny rozměrů všech ostatních rámečků v pravé části okna (nad poznámkami). Klepněte postupně na každý rámeček a upravte jeho nastavení automatické změny rozměrů tak, aby byly červené čáry horního a pravého okraje plné a všechny ostatní čáry přerušované – viz obrázek 5.7.




**Obrázek 5.7** Toto nastavení automatické změny rozměrů aplikujte na všechny zbývající rámečky a přilepte je tak k pravému hornímu rohu okna

Až nastavíte všechny rámečky, vyberte z nabídky **File** položku **Simulate Interface**. Zkuste změnit rozměry okna a všechny rámce, které okno obsahuje, by měly změnit své rozměry požadovaným způsobem.

V následujících sekcích a kapitolách již nebudeme problematiku automatické změny rozměrů rozebírat a necháme nastavení vlastností automatické změny rozměrů v návrzích uživatelských rozhraní na vás.

# Nová propojení

Předtím, než se dáte do psaní zdrojového kódu aplikace, zbývá ještě provést jeden poslední krok v nástroji Interface Builder: zapojit nové outlety a akce třídy `VillainTrackerAppDelegate`. Klepněte proto na její ikonu v hlavním okně nib souboru a otevřete inspektor propojení (⌘+**5**). Na seznamu outletů by se měly objevit oba dva nové outlety, `villainsTableView` a `window`, které jste do třídy před chvílí přidali. Propojte myší ikonu třídy `VillainTrackerAppDelegate` s oknem (táhněte myší za současného stisku klávesy **CONTROL** z ikony třídy nad ikonu okna v hlavním okně nib souboru nebo nad horní lištu samotného okna aplikace), uvolněte tlačítko myši a vyberte outlet `window`. Poté stejným způsobem propojte myší ikonu třídy s rámcem tabulky a po uvolnění tlačítka myši vyberte outlet `villainsTableView`.

Nyní přišel čas propojit pár outletů rámce tabulky s třídou `VillainTrackerAppDelegate`. Tyto dva outlety vám umožňují specifikovat, který objekt bude moci reagovat na metody delegátu (tyto metody se volají například při změnách výběru v tabulce) a datového zdroje (tyto metody se používají k zaplnění tabulky daty) třídy `NSTableView`. V tomto případě pak obsluhuje oba dva tyto úkoly třída `VillainTrackerAppDelegate`. Propojte proto myší rámec tabulky se třídou `VillainTrackerAppDelegate` (táhněte myší z tabulky nad ikonu třídy za současného stisku klávesy **CONTROL**) a vyberte outlet `dataSource`. Poté propojte rámec tabulky se třídou ještě jednou a tentokrát vyberte outlet `delegate`. Pokud se při propojování rámce tabulky se třídou nezobrazí seznam outletů, propojujete pravděpodobně se třídou `VillainTrackerAppDelegate` nechtěně instanci třídy `NSScrollView`. V takovém případě na rámec tabulky klepněte ještě jednou, abyste jej vybrali, a zkuste to znovu.

Posledním krokem pak je propojení nových tlačítek pro přidávání a odebírání padouchů s příslušnými metodami akcí ve třídě `VillainTrackerAppDelegate`. Nejprve propojte myší za současného stisku klávesy **CONTROL** s třídou `VillainTrackerAppDelegate` tlačítko s ikonou plus (táhněte z tlačítka nad ikonu třídy) a vyberte metodu `newVillain:`. Poté propojte s třídou `VillainTrackerAppDelegate` tlačítko s ikonou minus (opět táhněte z tlačítka nad ikonu třídy) a vyberte metodu `deleteVillain:`.

Po ustavení všech těchto propojení je návrh grafického uživatelského rozhraní prozatím hotový a můžete se vrátit ke psaní zdrojového kódu.

# Zdrojový kód obsluhující rámec tabulky

Díky základům položeným v kapitole 4. je přidání podpory pole padouchů překvapivě snadné. V zásadě stačí vytvořit pole, sdělit rámci tabulky, kdy chcete zobrazovat jeho obsah (kdykoliv se změní), implementovat několik metod datového zdroje, které načtou do tabulky její obsah a metodu delegátu, která se zavolá, když se výběr v tabulce změní (abyste mohli aktualizovat všechny ostatní rámce). Poté přidáte pár metod pro přidávání a odebírání padouchů z pole a je to!

Začněte inicializací pole, které bude uchovávat vaše zločince, a pokynem říkajícím rámci tabulky, že má načíst svůj obsah. Dosáhnete toho prostřednictvím několika nových řádků zdrojového kódu, které přidáte do metody `applicationDidFinishLaunching:`, viz tučně vtištěné řádky v následujícím výpisu. Nejprve vytvoříte pole uchovávající všechny padouchy (které na začátku obsahuje pouze jednoho padoucha vytvořeného ve zdrojovém kódu) a přiřadíte jej instanční proměnné `villains`. Poté řeknete rámci tabulky, aby načetl svůj obsah a vybral první řádek.

```
- (void)applicationDidFinishLaunching:(NSNotification *)aNotification {
self.villain = [NSMutableDictionary dictionaryWithObjectsAndKeys:
     NSLocalizedString(@"Lex Luthor", @"Lex Luthor"), kName,
     NSLocalizedString(@"Smallville", @"Smallville"), kLastKnownLocation,
     [NSDate date], kLastSeenDate,
     NSLocalizedString(@"Superman", @"Superman"), kSwornEnemy,
     NSLocalizedString(@"Revenge", @"Pomsta"), kPrimaryMotivation,
     [NSArray arrayWithObjects:
        NSLocalizedString(@"Intellect", @"Inteligence"),
        NSLocalizedString(@"Leadership", @"Vudcovstvi"), nil], kPowers,
     NSLocalizedString(@"Superhero Action", @"Predany"), kPowerSource,
     [NSNumber numberWithInt:9], kEvilness,
     [NSImage imageNamed:@"NSUser"], kMugshot,
     @"", kNotes,
     nil];
  self.villains = [NSMutableArray arrayWithObject:self.villain];
  [villainsTableView reloadData];
  [villainsTableView selectRow:0 byExtendingSelection:NO];
  [self updateDetailViews];
}
```

Na konec každé z metod `takeName:`, `takeLastSeenDate:` a `takeMugshot:` budete muset také přidat volání `[villainsTableView reloadData]`, aby se při úpravě ovládacích prvků reprezentujících tyto atributy příslušným způsobem upravil také obsah tabulky. Možná se vám zdá, že je opětovné načtení celé tabulky kvůli změně jediné hodnoty přehnané, ale nebojte se – třída `NSTableView` využívá koncepci odloženého načítání a za normálních okolností vyžaduje načtení obsahu řádku pouze v případě, že se má řádek zobrazit. V tabulce může být třeba milion řádků, ale pokud nikdy neposunete obsah rámce tak, aby zobrazoval více řádků než prvních třicet, rámec tabulky pravděpodobně více než oněch třicet řádků nikdy nenačte. A podobně, řeknete-li rámci tabulky, aby znovu načetl svá data, načte okamžitě pouze řádky, které zrovna zobrazuje, a ostatní se načtou až ve chvíli, kdy se posunou do zobrazované oblasti.

V tuto chvíli by mělo být možné aplikaci zkompilovat a spustit a až na velký prázdný rámec tabulky a několik nepoužitelných tlačítek by měla vypadat zhruba stejně jako na konci 4. kapitoly. Přišel čas tabulku naplnit padouchy!

# Rámec tabulky potřebuje vaši pomoc

Rámci tabulky jste sice řekli, aby načetl svůj obsah, ale dokud neimplementujete v objektu, který jste propojili s outletem `dataSource`, několik metod z protokolu `NSTableView-`

DataSource, nemá to jak udělat. NSTableViewDataSource je neformální protokol podobný většině protokolů delegátů, což znamená, že nemusíte deklarovat, že se vaše třída tímto protokolem řídí. Implementace metod protokolu NSTableViewDataSource je proto ve výsledku z pohledu jazyka Objective-C nepovinná. Navzdory tomu však protokol obsahuje několik metod, které implementovat musíte, pokud chcete, aby váš rámec tabulky něco zobrazil. Jedná se o metody numberOfRowsInTableView: a tableView:objectValueForTableColumn:row:, jejichž prostřednictvím rámec tabulky připraví svůj obsah na zobrazení. Následující výpis zdrojového kódu demonstruje způsob, jakým byste měli obě metody v souboru VillainTrackerAppDelegate implementovat:

```
- (NSInteger)numberOfRowsInTableView:(NSTableView *)aTableView {
  return [villains count];
}
- (id)tableView:(NSTableView *)aTableView
  objectValueForTableColumn:(NSTableColumn *)aTableColumn
   row:(NSInteger)rowIndex {
   return [[villains objectAtIndex:rowIndex]
           objectForKey:[aTableColumn identifier]];
}
```

Implementace první z obou metod by měla mluvit sama za sebe: jednoduše vrátíte velikost pole, aby rámec tabulky věděl, kolik řádků musí zobrazit. Druhou metodu pak rámec tabulky volá pokaždé, když se má v tabulce zobrazit nějaká její buňka. Obdržíte ukazatel na sloupec, ve kterém buňka je, a index jejího řádku.

Index řádku je stejný jako index příslušného objektu v poli obsahu, takže volání [villains objectAtIndex:rowIndex] vrací z pole villains příslušný objekt. Možná si vzpomínáte, že objekty modelu, které používáte v této aplikaci, jsou ve skutečnosti instance třídy NSMutableDictionary, jejichž hodnoty jsou dostupné prostřednictvím klíčů, a že jste při nastavování identifikačních atributů jednotlivých sloupců rámce tabulky použili stejné názvy klíčů, jaké používají objekty modelu, takže tento klíč ze sloupce tabulky vrátíte a použijete k získání příslušné hodnoty z objektu modelu.

Má-li se například zobrazit buňka v horním řádku a ve sloupci *Jméno*, předá se metodě index řádku (rowIndex) 0 a parametr aTableColumn ukazující na sloupec (instanci třídy NSTableColumn) s popiskem „Jméno“. Na základě indexu řádku pak metoda vybere z pole villains příslušného padoucha a poté použije hodnotu vrácenou metodou identifier sloupce tabulky – @"name" – k vrácení příslušné hodnoty z objektu vybraného padoucha.

Jestli máte potíže tomuto mechanismus přijít na kloub, podívejte se na následující způsob implementace stejné metody, který je o něco delší (má více řádků, ale provádí se přesně stejně dlouho) a který by vám mohl pomoci její funkci pochopit:

```
- (id)tableView:(NSTableView *)aTableView
 objectValueForTableColumn:(NSTableColumn *)aTableColumn
  row:(NSInteger)rowIndex {
```

```
  NSMutableDictionary *villain = [villains objectAtIndex:rowIndex];
  id keyName = [aTableColumn identifier];
  return [villain objectForKey:keyName];
}
```

Nyní by mělo být možné aplikaci zkompilovat a spustit a data výchozího padoucha by se měla zobrazit v jediném řádku tabulky i v ostatních ovládacích prvcích v okně. A pokud upravíte prostřednictvím ovládacích prvků jméno padoucha, datum jeho posledního spatření nebo jeho fotografii, měly by se nově zadané hodnoty aktualizovat i v tabulce. V samotném rámci tabulky zatím stále žádné úpravy provádět nemůžete, ale to se hned změní. Čeká vás totiž implementace další metody datového zdroje, `tableView:setObjectValue:forTableColumn:row:`, která umožní uživateli upravovat hodnoty přímo v rámci tabulky a současně bude aktualizovat novými hodnotami i příslušné ovládací prvky. Tato metoda je v zásadě inverzí předchozí metody a její implementace je skoro stejně jednoduchá:

```
- (void)tableView:(NSTableView *)aTableView
  setObjectValue:(id)anObject
  forTableColumn:(NSTableColumn *)aTableColumn
  row:(NSInteger)rowIndex {
  [[villains objectAtIndex:rowIndex] setObject:anObject
    forKey:[aTableColumn identifier]];
  [self updateDetailViews];
}
```

Vidíte, že metoda vyhledá příslušného padoucha stejným způsobem, avšak tentokrát místo vrácení hodnoty použijete identifikátor sloupce k nastavení atributu objektu padoucha. Metodu ukončíte voláním metody `updateDetailViews`, aby se jakékoliv úpravy dat provedené v rámci tabulky projevily i v ostatních ovládacích prvcích aplikace.

V tuto chvíli by se měla aplikace bez problémů zkompilovat a spustit a po úpravě hodnoty ve sloupci *Jméno* by se měla provedená změna projevit po stisku klávesy **RETURN** nebo **TAB** i v ovládacím prvku *Jméno*.

Nadešla chvíle pro implementaci metody delegátu, která se zavolá pokaždé, když se změní výběr v rámci tabulky. Díky ní budete vědět, který řádek v tabulce uživatel vybral, budete moci příslušným způsobem upravit instanční proměnnou padoucha, aby ukazovala na odpovídající řádek v poli padouchů, a znovu zobrazit všechny ovládací prvky, aby jejich hodnoty odpovídaly provedenému výběru. Přidejte tedy do sekce `@implementation` v souboru *VillainTrackerAppDelegate.m* následující metodu:

```
- (void)tableViewSelectionDidChange:(NSNotification *)aNotification {
  if ([villainsTableView selectedRow] > -1) {
    self.villain = [self.villains
              objectAtIndex:[villainsTableView selectedRow]];
    [self updateDetailViews];
    NSLog(@"aktualni hodnoty atributu padoucha: %@", villain);
  }
}
```

Jádro metody je zabalené do klauzule `if`. To je nezbytné, protože rámec tabulky také může ohlásit, že v něm není aktuálně vybraný žádný řádek. Udělá to tak, že jeho metoda `selectedRow` vrátí hodnotu `-1`. Jinak by vám pochopení funkce zdrojového kódu této metody nemělo činit větší potíže. Zkompilujte aplikaci, abyste se ujistili, že jste se nedopustili žádných chyb, ale zatím ji nespouštějte – ještě zbývá přidat kód pro vytvoření a smazání padoucha, takže zatím nelze změnu výběru v tabulce nijak otestovat (tabulka má zatím pouze jeden řádek).

# Přidávání a mazání padouchů

Nyní konečně dojde na implementaci metod `newVillain:` a `deleteVillain:`, jejichž „pahýly" jste vytvořili výše. Metoda `newVillain:` přidá do pole padouchů nový „prázdný" objekt padoucha, řekne tabulce, aby znovu načetla svůj obsah, a rámci tabulky, který řádek tabulky má vybrat (poslední, protože jste ho právě přidali):

```
- (IBAction)newVillain:(id)sender {
  [window endEditingFor:nil];
  [villains addObject:[NSMutableDictionary
  dictionaryWithObjectsAndKeys:
            @"", kName,
            @"", kLastKnownLocation,
            [NSDate date], kLastSeenDate,
            @"", kSwornEnemy,
            @"", kPrimaryMotivation,
            [NSArray array], kPowers,
            @"", kPowerSource,
            [NSNumber numberWithInt:0], kEvilness,
            [NSImage imageNamed:@"NSUser"], kMugshot,
            @"" , kNotes,
            nil]];
  [villainsTableView reloadData];
  [villainsTableView selectRow:[villains count]-1
  byExtendingSelection:NO];
}
```

Zde je nové pouze volání metody `[window endEditingFor:nil]`, které jednoduše řekne oknu, že je čas ukončit jakékoliv úpravy hodnot ovládacích prvků, jako je například vkládání textu do textového pole. Tuto metodu musíte zavolat, aby bylo možné upravovanou hodnotu uložit do jejího objektu padoucha, protože později v metodě `newVillain:` změníte výběr v tabulce a tím pádem také odstraníte hodnoty všech ovládacích prvků.

Nyní již bude tlačítko pro přidání padoucha fungovat, takže můžete aplikaci zkompilovat, spustit a tlačítko vyzkoušet. Navíc již také lze vybírat myší v rámci tabulky různé padouchy a hodnoty všech ostatních ovládacích prvků se budou přizpůsobovat provedenému výběru.

Další na řadě je metoda `deleteVillain:`. Přidali jsme do ní komentáře popisující její různé sekce, o kterých budeme podrobně mluvit, až ji implementujete.

```
- (IBAction)deleteVillain:(id)sender {
 //
 // 1. sekce:
 //
 [window endEditingFor:nil];
 int selectedRow = [villainsTableView selectedRow];
 //
 // 2. sekce:
 //
 [villains removeObjectIdenticalTo:villain];
 [villainsTableView reloadData];
 //
 // 3. sekce:
 // if (selectedRow >= [villains count]) {
  selectedRow = [villains count]-1;
 }
 // 4. sekce:
 if (selectedRow > -1) {
  // zrušte výběr všech řádků, abyste zajistili, že rámec tabulky vidí
  // "změnu" provedeného výběru, dokonce i když může být vybraný
  // stále stejný index řádku.
  // 5. sekce:
  [villainsTableView deselectAll:nil];
  [villainsTableView selectRow:selectedRow byExtendingSelection:NO];
  [self updateDetailViews];
 }
}
```

Tato metoda je o něco složitější než většina ostatního zdrojového kódu, se kterým jste se v knize až doposud setkali, takže ji rozebereme o něco podrobněji.

V 1. sekci řeknete oknu, aby ukončilo jakékoliv probíhající úpravy, a poté zjistíte index aktuálně vybraného řádku – tj. řádku, který se má smazat. Sice již víte, který padouch je vybraný, protože je uložený v instanční proměnné, ale index řádku je důležitý pro správný výběr řádku v tabulce po smazání řádku aktuálního (ve 4. sekci).

V 2. sekci smažete vybraný objekt (na který ukazuje instanční proměnná `villain`) z pole `villains` a poté řeknete rámci tabulky, aby znovu načetl svůj obsah. Všimněte si metody `removeObjectIdenticalTo:`, která odstraňuje padoucha z pole. Tato metoda nechá nalézt hledaný objekt samotnou instanci třídy `NSMutableArray`, která porovná jeho aktuální adresu v paměti s adresami objektů, které obsahuje, takže skutečně najde a odstraní přesně ten objekt, který metodě předáte. Kdybyste použili častěji používanou metodu `removeObject:`, porovnávala by objekty prostřednictvím metody `isEqual:`, která porovnává jejich hodnoty, a riskovali byste smazání padouchů s identickými hodnotami atributů.

V 3. sekci provádíte malou úpravu: pokud byl předchozí vybraný index řádku v poli poslední, je po odstranění objektu z pole mimo platný rozsah. V takovém případě jej nastavíte tak, aby ukazoval na aktuálně poslední objekt.

Ve 4. sekci metoda zkontroluje, zda je hodnota proměnné `selectedRow` 0 nebo vyšší. Tato kontrola je důležitá, protože existuje reálná šance, že metoda nastavila v 3. sekci hodnotu proměnné na -1! Představte si, že máte v poli pouze jediný objekt a klepnete na tlačítko pro odstranění padoucha: výsledkem bude nastavení hodnoty proměnné `selectedRow` na -1 (na začátku byla její hodnota 0, což je index jediného řádku v poli, takže třetí sekce nastaví její hodnotu na -1). 4. sekce proto ošetřuje tuto možnost prostým vynecháním zbytku zdrojového kódu metody.

V 5. sekci metoda zruší výběr všech položek v rámci tabulky, poté vybere řádek, který se má vybrat po odstranění některého z padouchů, a nakonec hodnotami nově vybraného padoucha aktualizuje všechny ostatní rámce v okně. Možná to není na první pohled úplně zřejmé, ale zrušení výběru všech řádků v rámci tabulky je nutné, protože jinak by se nemusela vždy zavolat metoda `tableViewSelectionDidChange:` (po smazání některého z řádků bude často index vybraného řádku stejný a tabulka nemá jak zjistit, že jste smazali objekt z pole a že index nyní odkazuje na jiný objekt).

Aplikaci by nyní mělo být možné bez problémů zkompilovat a spustit a měla by již umožňovat i mazání padouchů a při smazání některého z nich aktualizovat hodnoty ostatních ovládacích prvků hodnotami atributů nově vybraného padoucha.

# Závěr

Nová verze aplikace VillainTracker je jednoduchou demonstrací údržby seznamu položek, jejich zobrazování v rámci tabulky a upravování detailů jednotlivých položek v oddělené sadě ovládacích prvků. Při jejím programování jste se dozvěděli něco o způsobu, jakým rámec tabulky používá svůj datový zdroj při přístupu k položkám, které má zobrazovat a které v něm lze upravovat, a viděli, jak informuje svůj delegát, když se výběr v tabulce změní a umožní vám ručně aktualizovat obsah rámců vzhledu, které závisí na výběru v rámci tabulky.

Máte-li nějaké zkušenosti s jinými prostředími pro vývoj grafických uživatelských rozhraní desktop aplikací, možná vám byla tato demonstrace poněkud cizí, ale doufáme, že se nám podařilo předvést některé výhody přístupu, který prostředí Cocoa podporuje, reprezentovaného například důsledným oddělením zdrojového kódu od návrhu uživatelského rozhraní aplikace. V tuto chvíli se vám však musíme k něčemu přiznat: způsob, jakým jsme vás nutili postupovat v kapitolách 4. a 5., není nutně tím nejlepším způsobem programování podobné funkcionality v prostředí Cocoa. Přesto, že byla doposud rozebraná problematika poměrně jednoduchá, se ve skutečnosti jednalo o *náročnější* způsob řešení daných problémů. V průběhu několika posledních let zakořenil v komunitě vývojářů Cocoa aplikací nový přístup k programování grafických uživatelských rozhraní aplikací, který se prosazuje stále častěji. Tato technologie má název Cocoa Bindings a je také tématem 6. kapitoly.