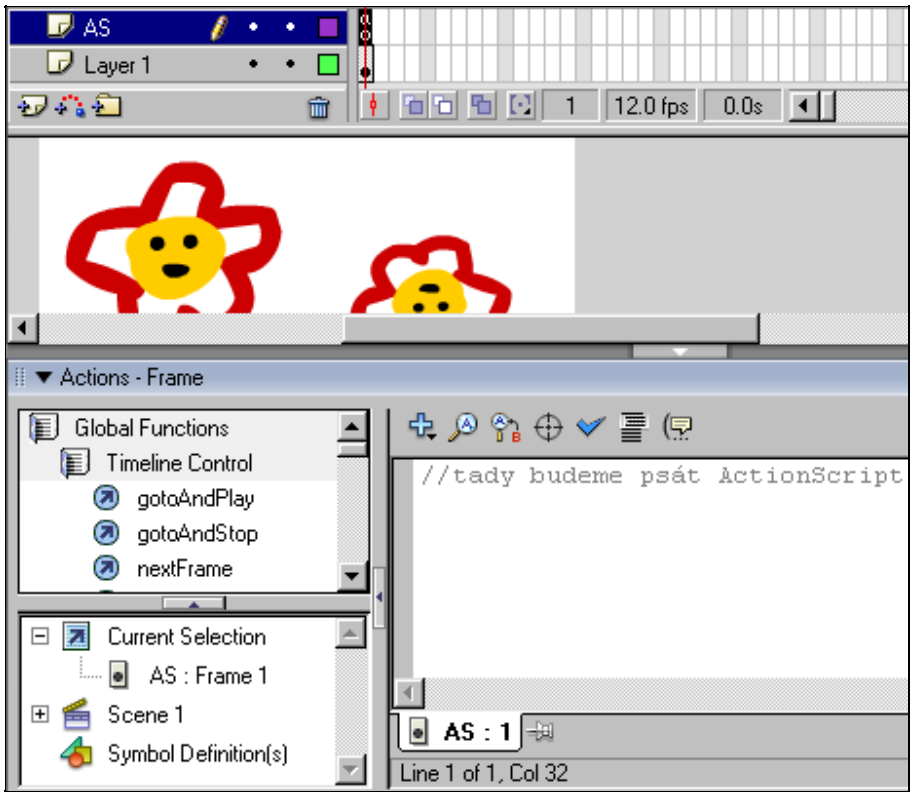
3. Adresování a ovládání movie klipů

Intermezzo - Jak psát ActionScript

Při samotném psaní ActionScriptu (dále jen AS)- příkazů k ovládání (nejen) grafických objektů Flashe - narazíme na četná úskalí. Nejmenší problém je najít panel Actions (F9), do nějž budeme kód zapisovat:



V jeho levé horní části je strukturovaný seznam příkazů AS, který je ovšem příliš rozsáhlý a nepřehledný na to, aby byl použitelný; v levé spodní části je okénko navigace mezi částmi AS rozmístěnými na různých místech klipu, ale ani na to nebudeme příliš spoléhat a budeme se snažit psát AS samostatně a na takové místo, kde bude vždy a očích. Proto si **zavedeme pro AS vlastní vrstvu v časové ose**, nazveme ji příznačně AS a před tím, než začneme psát do okna Actions, vybereme vždy nějaký snímek ve vrstvě AS.

Vykonávání AS je vázáno na časovou osu klipu, tj. okamžitě po spuštění klipu se vykoná ActionScript ve snímku 1 a kusy kódu v dalších snímcích se vykonávají tehdy, když dojde k přehrání těchto snímků. AS zapsaný ve snímku časové osy se projeví malým symbolem alfa; nyní vás upozorním na první úskalí AS a to, že se dá zapsat i jinam než do snímku časové osy (např. lze ho přímo přiřadit libovolnému symbolu), ale tím kód ztrácí na přehlednosti a proto bychom se toho měli dopouštět pouze v opodstatněných případech.

Druhé úskalí je, že AS používá symboly, které nejsou na české klávesnici snadno dostupné, což lze vyřešit zapamatováním speciálních kombinací kláves nebo používáním např. anglické klávesnice.

Úskalím třetím, a pro tuto chvíli snad posledním, je to, že ActionScript příliš systematicky nekontroluje chyby, takže si musíte dávat velký pozor, co píšete (včetně velkých a malých písmen!) a kam to píšete - to, že vám něco nefunguje, je s největší pravděpodobností způsobeno nějakým banálním překlepem nebo tím, že jste nevložili AS do hlavní časové osy.

3.1 Příkazy *play()* a *stop()*

Zpět od teorie k praxi. Možná už jste se setkali s příkazem *stop()*, který jste vložili do posledního snímku vašeho klipu, abyste mu zabránili v tom, aby hrál v nekonečné smyčce znovu od začátku. My tento příkaz použijeme k zastavení rotace květu a vzhledem k tomu, že jsme si dvě instance květu na ploše pojmenovali, budeme je moci zastavit adresně, nikoliv obě najednou.

V AS platí jednoduché pravidlo, nejdříve **oslovíme jménem objekt**, se kterým chceme komunikovat, a **přes tečku připojíme příkaz**, který chceme, aby objekt vykonal, tj. např

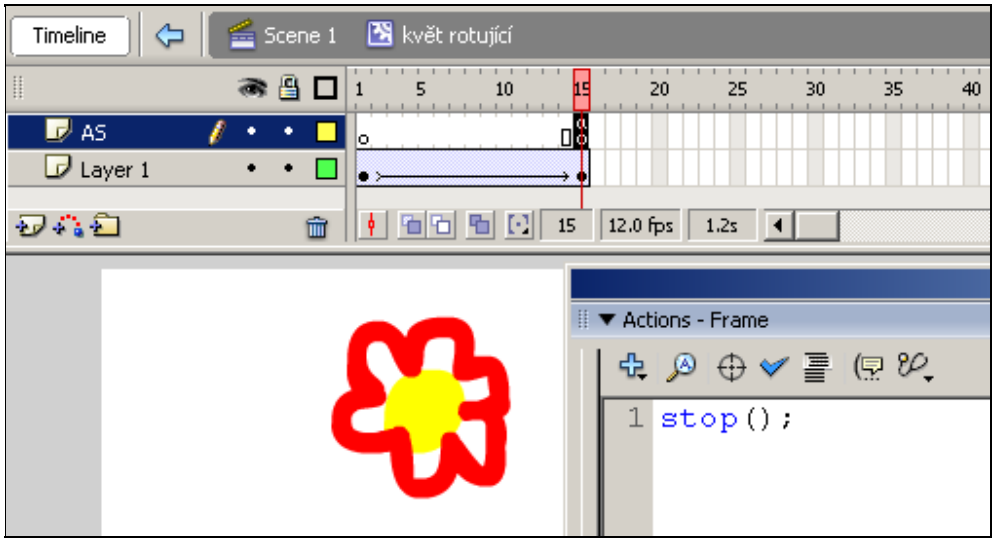
kvet1_mc.stop();

Přeloženo do lidské řeči, tímto sdělujeme objektu jménem KVET1_MC (což je animace rotujícího květu, který máme na ploše), aby zastavil své přehrávání. Vložíme-li tento příkaz do prvního snímku vrstvy AS a spustíme klip, měl by jeden z květů stát, zatímco druhý se točí. Většina příkazů je v AS následovány kulatými závorkami (...), ve kterých se mohou vyskytovat nějaké parametry (poznáme dále), ale i když žádný parametr nepředáváme, musíme prázdné závorky uvést. Každý slušný příkaz je ukončen středníkem ; aby se dalo AS jasně najevo, kde příkaz končí, kdyby se např. nevešel do jedné řádky.

Intermezzo - Skript v hlavní časové ose vs. skript v časové ose symbolu

Zapíšeme-li skript do hlavní časové osy, ovládáme to, co se děje na ploše, na úrovni symbolů na plochu vložených. Co když ale chceme ovládat to, co se děje uvnitř symbolu - movie klipu? **Každý movie klip má vlatní časovou osu a nám nic nebrání v tom do jeho časové osy vkládat ActionScript.** Tak docílíme toho, že se AS vložený do určitého snímku vykoná vždy, když v movie klipu dojde k přehrání tohoto snímku. Typicky budeme využívat první a poslední snímek movie klipu - v prvním snímku bude skript pro nějakou inincializaci (nastavení počátečního stavu) a v posledním snímku bude skript, který rozhodne co dál, když skončí přehrávání movie klipu - standartně klipy hrají stále dokola, tak my ho můžeme např. zastavit nebo skočit v přehrávání na úplně jiný klip...

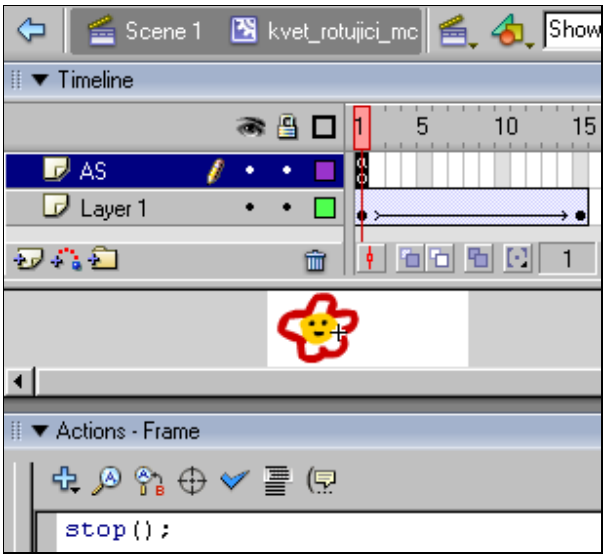
Skript v časové ose symbolu si nejsnáze vyzkoušíme na našem příkladu rotujícího květu - vložíme-li dovnitř časové osy tohoto movie-klipu na poslední snímek příkaz *stop()*, animace po prvním proběhnutí skončí a zůstane stát. Proč příkaz *stop()* nepředchází jméno květu bude vysvětleno v dalším odstavci. Pozor, pokud vkládáme skript do samostatné vrstvy (doporučeno), musíme na jejím posledním snímku nejdříve vytvořit klíčový snímek (F6), jinak by se skript uložil do snímku prvního:



Ošemetný příklad:

Nyní se můžeme vrátit k našim jednoduchým příkazům pro přehrávání klipů. Opakem příkazu *stop()* je příkaz *play()*, který zastavený movie klip spustí (a ten bude hrát, dokud nenarazí na další příkaz *stop();*). Abychom si

play(), který zastavený movie klip spustí (a ten bude hrát, dokud nenarazí na další příkaz *stop();*). Abychom si příkaz *play()* mohli vyzkoušet, potřebujeme movie klip, který na začátku stojí. Jak na to - stačí si uvědomit, že každý movie klip má svou vlastní časovou osu (kterou vidíme, když poklepáním dostaneme klip do symbol editing módu), napíšeme-li v prvním snímku časové osy movie klipu příkaz *stop()*, dostaneme klip, který vždy stojí.

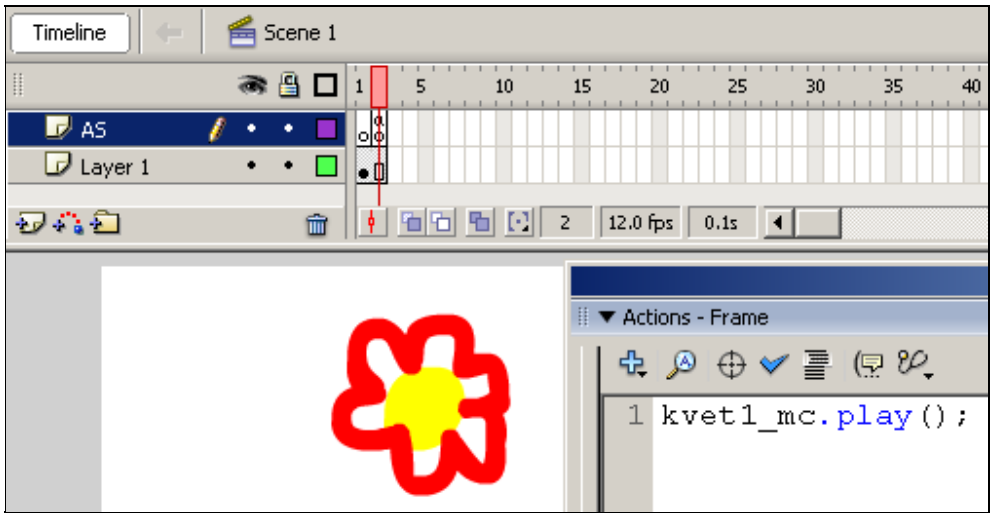


Nyní byste mohli správně podotknout, že není jasné, komu je příkaz *stop()* určen, neboť mu nepředchází jméno žádného objektu. **Zaveďme si tedy pravidlo, že není-li výslovně uveden adresát příkazu, týká se příkaz té časové osy, ve které je uveden** . To je v našem případě časová osa animovaného květu, která správně zareaguje tím, že zastaví své přehrávání. Šlo by uvést (kdybysme byli puntičkáři), příkaz ve tvaru *this.stop()*, kde výraz *this* (který vždy odkazuje na objekt, ve kterém se právě nacházíme) zastupuje chybějícího adresáta (více o *this* v dalším odstavci).

Spustíme-li tento klip, měl by květ stát. A nyní zrada - napíšeme-li do prvního snímku vrstvy AS hlavní časové osy

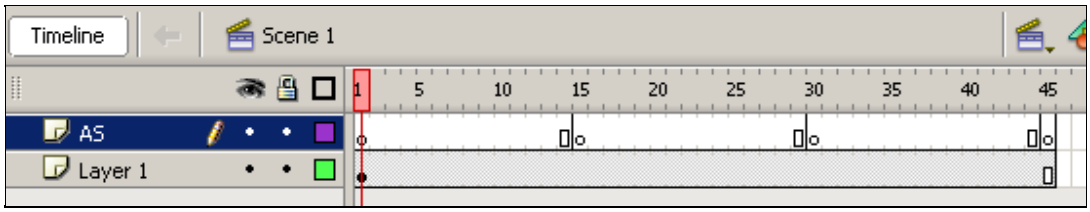
kvet1_mc.play();

očekávali bychom, že květ se bude točit, ale nestane se tak. Je to díky tomu, že příkaz *play()* v hlavní časové ose vykoná Flash dřív než příkaz *stop()* uvnitř květu. Tento problém vyřešíme tím, že přidáme do hlavní časové osy snímek (F5) a příkaz *play()* přesuneme do druhého snímku hlavní časové osy, čímž už se příkazy vykonají ve správném pořadí a květ se roztočí:



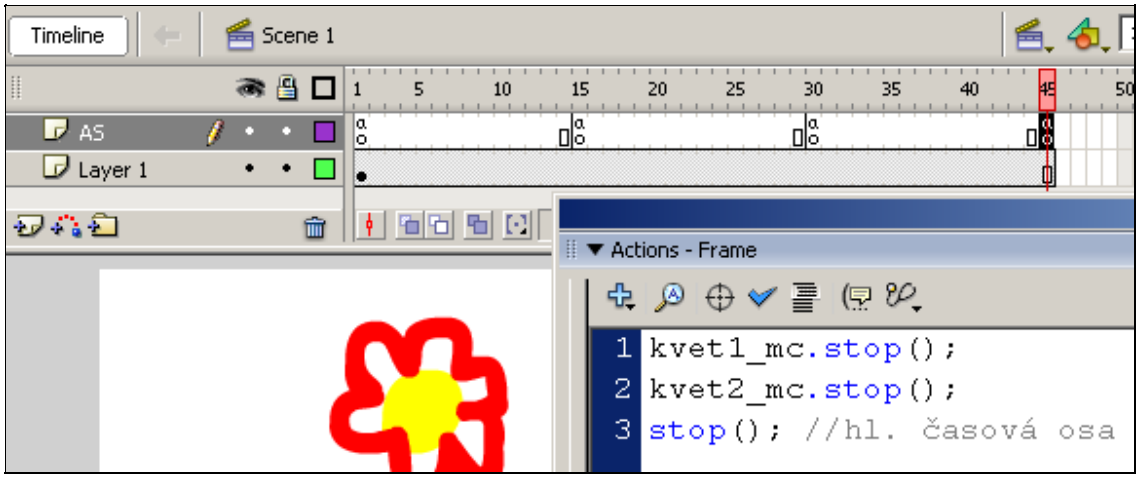
Praktičtější příklad:

Pojďme si zkusit ovládat přehrávání rotujících květů podél nějaké delší časové osy - připravme si např. 45-snímkovou hlavní časovou osu, v níž prvních 15 snímků se bude točit pouze květ č. 1, druhých 15 snímků se bude točit pro změnu pouze květ č. 2 a posledních 15 snímků se budou točit oba květy současně. Pak se vše zastaví. Připravíme si tedy časovou osu, kde vrstva s květinami bude jeden dlouhý 45-snímkový úsek, zatímco vrstva ActionScriptu bude obsahovat klíčové snímky na snímcích 1, 15, 30 a 45:



1. Pokud časová osa květiny neobsahuje žádné příkazy "stop", roztočily by se po spuštění obě květiny. My však chceme, aby se točila pouze první z nich, tzn. druhou budeme muset hned v prvním snímku hlavní časové osy zastavit příkazem
   kvet2_mc.stop();
2. Ve snímku 15 naopak zastavíme květinu první a pustíme květinu druhou příkazy
   kvet1_mc.stop();
   kvet2_mc.play();
3. Ve snímku 30 se k druhé květina přidá i první
   kvet1_mc.play();
4. Mají-li se na konci obě květiny zastavit, logicky doplníme do snímku 45
   kvet1_mc.stop();
   kvet2_mc.stop();

Pokud bychom nyní klip pustili (CTRL+Enter), nezastavil by se na posledním snímku úplně, ale pustil by se znovu od prvního snímku, takže květiny by se po chvíli opět roztočily. Bude tedy potřeba na posledním snímku zastavit hlavní časovou osu samotným příkazem *stop()* bez určení "adresáta". Skript na posledním snímku bude tedy vypadat takto (na pořadí všech tří příkazů samozřejmě nezáleží):



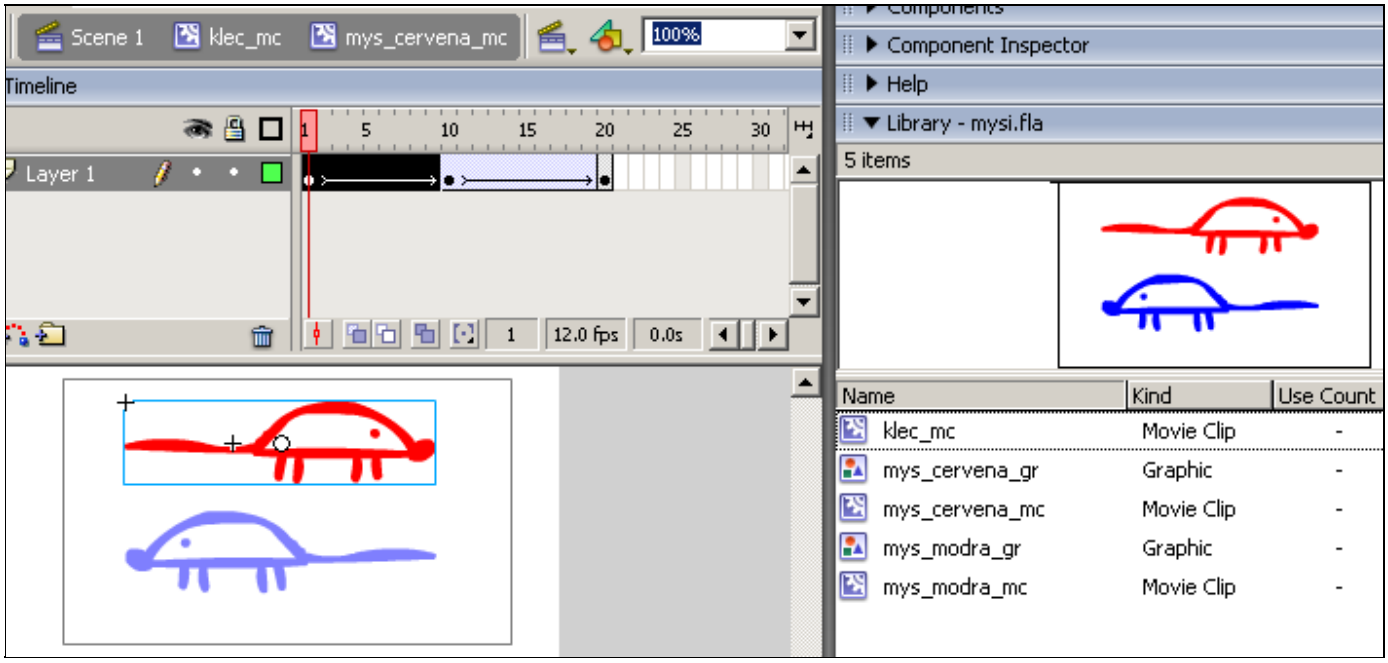
3.2 Adresování vnořených movie-klipů

Doposud jsme adresovali pouze objekty ležící přímo na ploše (pomocí jejich jména insatnce), ale Flash umožňuje strukturovat klipy složitěji a pak si nevystačíme s takto jednoduchým adresováním. To je třeba případ "vnořených movie klipů", kdy jeden klip je vložen do druhého - vnořený klip pak už neleží přímo na ploše a plocha už se ho jeho jménem nedovolá.

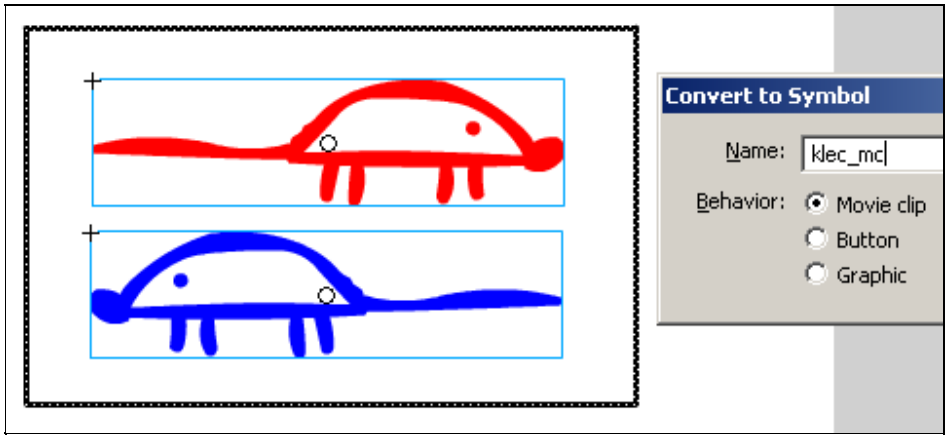
3.2.1 Vnořený movie klip - příklad

Vytvoříme si klip "Myši v kleci", kde dvě animace myší (nezávislé symboly - movie klipy *mys_cervena_mc* a *mys_modra_mc*) jsou sloučeny do jednoho symbolu - movie klipu *klec_mc*. Zdrojový soubor lze stáhnout zde mysi.fla.

Struktura klipu je patrná z následujícího obrázku - vidíme jak použité knihovní symboly, tak vnitřek movie klipu *klec_mc*, v němž máme zrovna otevřen vnořený movie klip *mys_cervena_mc*. Hierarchii objektů ukazuje lišta nad časovou osou: Scene 1 -> klec_mc -> mys_cervena_mc.



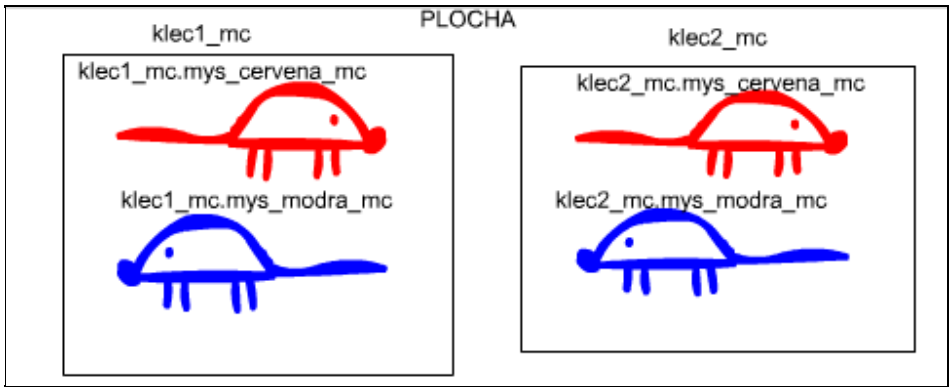
Kdybyste si klip připravovali sami, vytvořit animace myšek by pro vás neměl být žádný problém, takže se jenom krátce zastavím u toho, jak je "zavřít" do klece - tj. jak vytvořit movie-klip *klec_mc*. Na prázdnou plochu vytáhněme instance obou myší z knihovny, orámujme je obdélníkem klece a vyberme všechny objekty na ploše. Nyní přes F8 budeme převádět na symbol - nezapomeňte zvolit jako typ *movie klip*, jinak se v budoucnu zbavíme možnosti "klec" pojmenovat a adresovat!



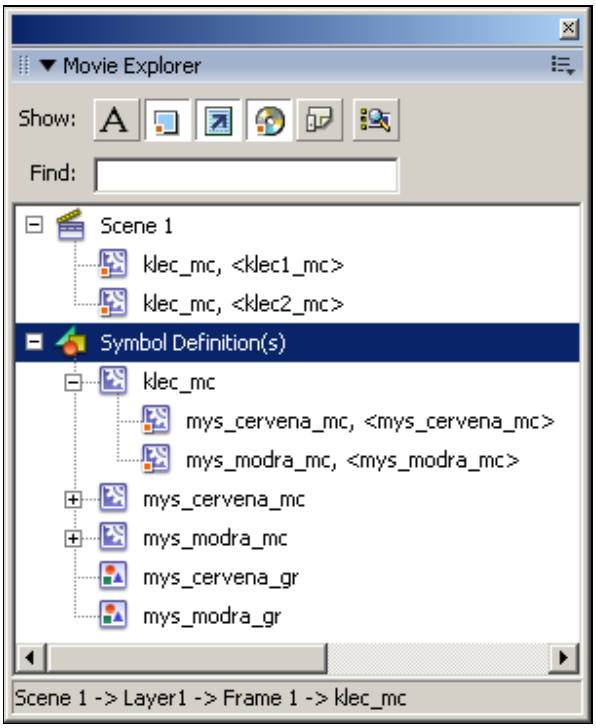
3.2.2 Hierarchická struktura dokumentu

Pojďme si nyní vytáhnout 2 instance movie klipu *klec_mc* na plochu, takže budeme mít celkem 4 instance myší v dokumetu. Abychom s instancemi mohli dále pracovat, je třeba je pojmenovat - otevřme si jednu z klecí na ploše a pojmenujme v ní instance myší stejně jako v knihovně, tj. *mys_cervena_mc* a *mys_modra_mc*. Pokud nyní otevřeme druhou klec, zjistíme, že myši v ní už pojmenované jsou - stejně jako v první kleci. Nemělo by nás to překvapit - klece jsou přeci instance téhož knihovního symbolu, takže změny uvnitř (v našem případě pojmenování) v jedné instance se automaticky projeví ve všech dalších instancích. Správně byste mohli namítnout, že myši v našem dokumentu nyní nemají unikátní (jedinečná) jména - obě modré a obě červené se jmenují stejně a bude tedy problém je adresovat. Není to tak docela pravda, protože při adresování používáme ve skutečnosti tzv. **cestu k instanci**, nikoliv jen jméno instance.

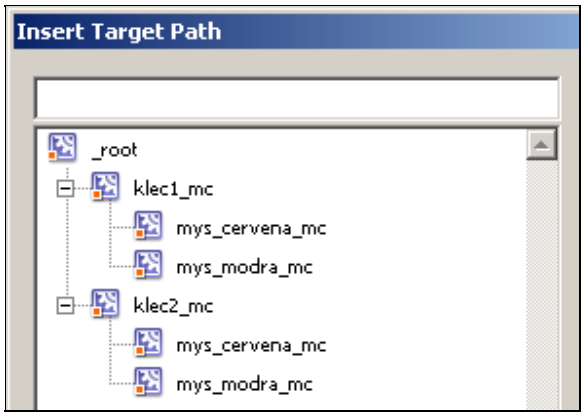
Cesta k instanci - anglicky "target path" [tárget páθ] se skládá ze všech jmen objektů, kterými musíme projít cestou k objektu, který chceme adresovat. Je-li nějaký objekt vložen přímo na plochu, nestojí mu při adresování z plochy v cestě nic, a proto se adresuje pouze jménem. To jsme s úspěchem odzkoušeli v sekci 3.1 s příkazy *stop()* a *play()*. Pojmenujme nyní obě instance klecí na ploše, a to *klec1_mc* a *klec2_mc*; myši uvnitř už pojmenované jsou. Chceme-li z plochy adresovat některou z myší, v cestě nám vždy stojí jedna z klecí; takže jméno klece musí být zahrnuto v cestě k instanci, správné adresování myši je pak např. *klec1_mc.mys_cervena_mc* - adresy všech myší jsou viditelné na následujícím obrázku.



Nyní je nám už asi intuitivně jasné, co se rozumí "hierarchickou strukturou dokumentu"; zmíníme se o dvou nástrojích Flashe, které nám tuto strukturu formálně zobrazí. Prvním (ale ne běžně používaným) je okno **Movie Explorer** (ALT+F3), které nám ukazuje, že naše scéna obsahuje 2 instance symbolu *klec_mc* a pak obsahuje seznam použitých symbolů s naznačením jejich struktury.



Pravdivější obrázek o struktuře dokumentu nám podá "**zaměřovač**" - nástroj Insert Target Path v panelu Actions (jeho využití viz dále) - ten nám ukáže strukturu všech pojmenovatelných objektů v dokumentu (tj. v našem případě movie klipů). Vidíme obě dvě klece a v každé z nich dvě myši. *_root* je odkaz na nejvyšší úroveň struktury objektů v dokumentu, tj. na Plochu.



Abychom si vyzkoušeli, že umíme adresovat vnořené objekty a posílat jim příkazy, pojďme "znehybnit" např. modrou myš v druhé kleci. Příkaz zadaný nejlépe do prvního snímku hlavní časové osy bude vypadat následovně:

klec2_mc.mys_modra_mc.stop();

Klip by se měl nyní chovat jako animace vložená na úvod oddílu 3.2.1.