More Flash MVC: A Multiple-Choice Quiz

- Suppose we wanted to implement a multiple-choice quiz in Flash and ActionScript
- For such a quiz, it would be useful to have a Question object, which would contain the question text, the available choices, and the correct answer
- For convenience, let's create a complete Question in a single line — thus, its *constructor* should take the question text, the set of choices, and the answer
- Finally, there's no need to change a Question, so we'll forego setter methods

- Based on this specification, *Question.as* would be:

```
class Question {
    private var questionMessage: String;
    private var choices: Array;
    private var answer: String;

    /**
     * All-in-one constructor.
     */
    function Question(msg: String, choices: Array, ans: String) {
        this.questionMessage = msg;
        this.choices = choices;
        this.answer = ans;
    }

    function getMessage() : String {
        return this.questionMessage;
    }

    function getChoices() : Array {
        return this.choices;
    }

    function getAnswer() : String {
        return this.answer;
    }
}
```

Initializing the Model

- Now that we have a Question object defined, we can create our quiz; a straightforward way to do this is to build an Array of Question objects:

```
var quiz = [
    new Question("Who are you?", ["me", "you", "him", "her"], "me"),
    new Question("Who's on first?", ["what", "why", "who", "how"], "who"),
    new Question("What the hey?", ["la", "dee", "da"], "la"),
    new Question("Whoop dee do?", ["re", "mi", "fa", "sol", "la"], "mi")
];
```

- Note how we can have varying numbers of choices — array size is not restricted here
- As usual, this typically belongs in the Actions layer of the scene that holds the quiz

Other Model Options

- Instead of typing out the questions in ActionScript, you can read them from a file — this allows you to modify your set of questions without having to touch your .FLA and .SWF files
- Having a *quiz* array allows you to access the questions by number (0 to the array's size – 1; the sample array above has four questions, so its range is 0 to 3) — instead of reading the entire array into memory, you can create individual question files that are numbered, ("q0.txt") and read those in as needed

Setting Up the View and Controller

- There are a lot of possible views for a multiple-choice quiz; the general guidelines are:
  - Give instance names to your components so that you can touch them from ActionScript
  - Plan out which parts of your scene are meant to display which parts of the question (text, choices)
  - Decide on how the user is to answer the question, and what will happen for right and wrong answers
- Then, lay everything out on the stage

- The following ActionScript (one of many possibilities) uses a Label for the question text (*question_label*), a List to hold the choices (*choices_list*), and a Button symbol for confirming the user's answer (*answer_btn*); with the right answer we advance to the next question in the pre-defined question array (*quiz*):

```
function setCurrentQuestion(number) {
    question_label.text = quiz[number].getMessage();
    choices_list.dataProvider = quiz[number].getChoices();
}

var currentQuestion = 0;
setCurrentQuestion(currentQuestion);

answer_btn.onRelease = function() {
    if (choices_list.selectedItem == quiz[currentQuestion].getAnswer()) {
        response_label.text = "RIGHT!";
        currentQuestion = currentQuestion + 1;
        setCurrentQuestion(currentQuestion);
    } else {
        response_label.text = "WRONG!  Try again";
    }
};
```

Details and Possibilities

- Remember that the *quiz* array has a finite size, so you need to check if *currentQuestion* is in the valid range
- Instead of going through questions in sequence, you can learn how to use the *Math.random()* method to pick a question at random — of course, you still have to ensure that your random number is in range
- A wrinkle in choosing random questions is to make sure that you don't re-ask the same question
- You may want to keep score — maintain a *score* variable and display its value

- Other ideas to consider:
  - Remember that you still have access to Flash's other features, so you can do more than just show a message for right or wrong answers — for example, you can play a sound or movie clip
  - Additional scripting may allow you to do things like supply letters (a, b, c, d) for the answer choices, provide a timer, or have different difficulty levels
  - As an alternative design, you can ask the Question object *itself* whether an answer is wrong or right — define a method called *isCorrectAnswer(answer)* that returns true or false, and use that instead of performing the comparison at the level of the button's *onRelease* action function