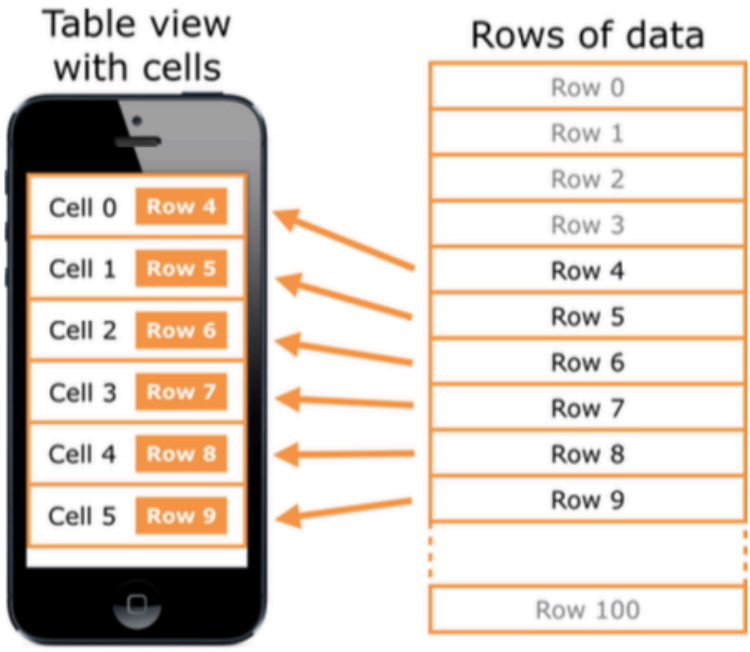
Table View

Cell

- **Tables display their data in cells. A cell is related to a row but it's not exactly the same. A cell is a view that shows a row of data that happens to be visible at that møment. If your table can show 10 rows at a time on the screen, then it only has 10 cells, even though there may be hundreds of rows with actual data.**

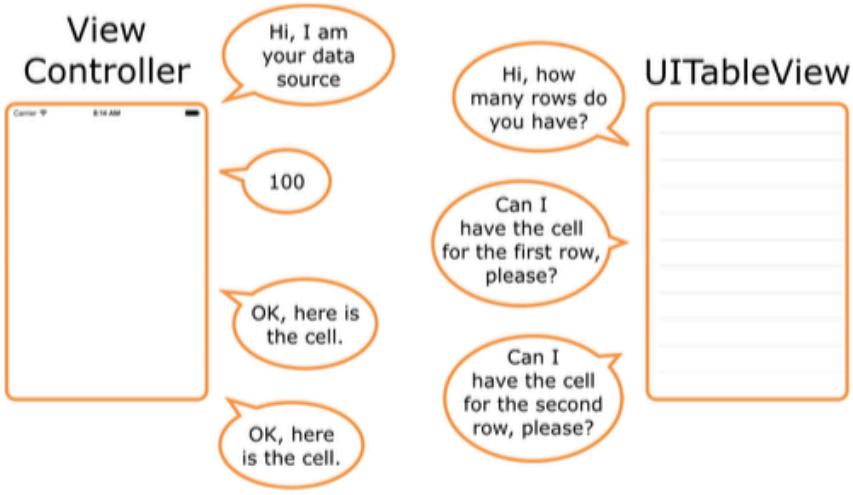


Cells display the contents of rows

Data Source

- The data source is the link between your data and the table view.
- Usually the view controller plays the role of data source and therefore implements these methods.
- The table view needs to know how many rows of data it has and how it should display each of those rows. But you can't simply dump that data into the table view's lap and be done with it. You don't say: "Dear table view, here are my 100 rows, now go show them on the screen." Instead, you say to the table view: "This view controller is now your data source. You can ask it questions about the data anytime you feel like it."
- Once it is hooked up to a data source – i.e. your view controller – the table view sends a "numberOfRowsInSection" message to find out how many rows there are.
- And when the table view needs to draw a particular row on the screen it sends the "cellForRowAtIndexPath" message to ask the data source for a cell.

- You see this pattern all the time in iOS: one object does something on behalf of another object. In this case, the ChecklistViewController works to provide the data to the table view, but only when the table view asks for it.



The dating ritual of a data source and a table view

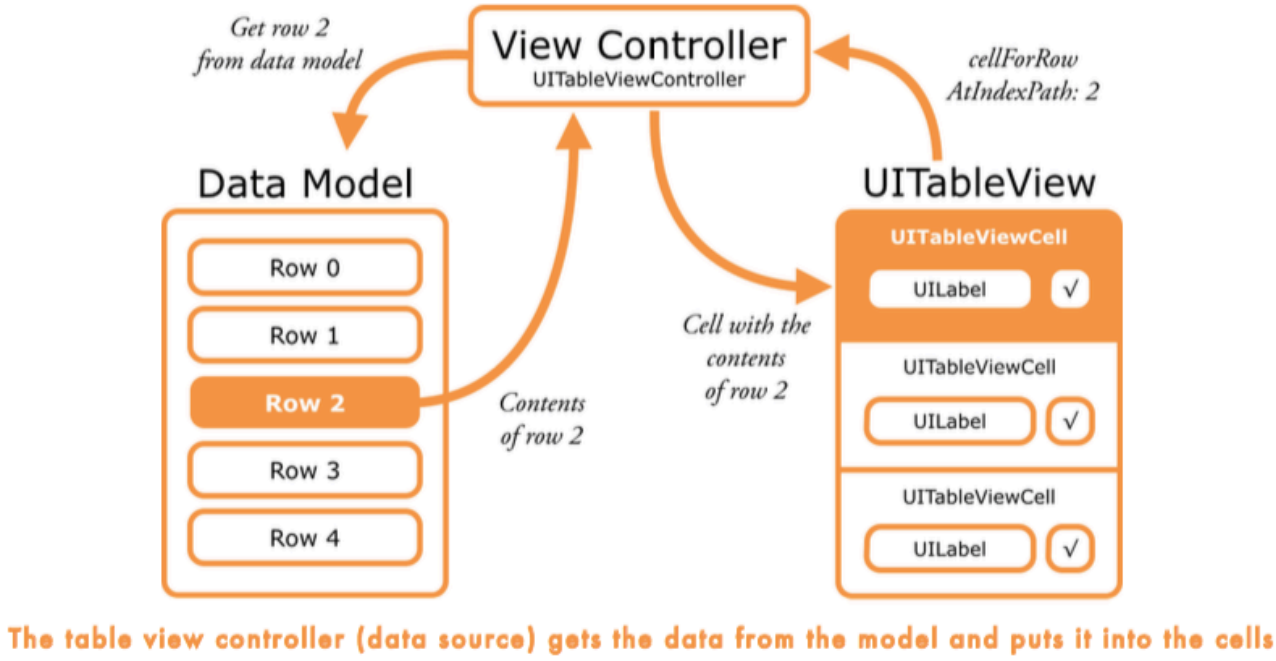
Index Path

- NSIndexPath is simply an object that points to a specific row in the table. It is a combination of a row number and a section number, that's all. When the table view asks the data source for a cell, you can look at the row number inside the indexPath.row property to find out for which row this cell is intended.

Delegate

- The concept of delegation is very common in iOS. An object will often rely on another object to help it out with certain tasks. This separation of concerns keeps the system simple, as each object does only what it is good at and lets other objects take care of the rest. The table view offers a great example of this.
- The table view doesn't really care who its data source is or what kind of data your app deals with, just that it can send the cellForRowAtIndexPath message and that it will receive a cell in return. This keeps the table view component simple and moves the responsibility for handling the data to where it belongs: in your code.
- Likewise, the table view knows how to recognize when the user taps a row, but what it should do in response completely depends on the app. In this app you'll make it toggle the checkmark; another app will likely do something totally different.
- Using the delegation system, the table view can simply send a message that a tap occurred and let the delegate sort it out.

MVC



The table view controller (data source) gets the data from the model and puts it into the cells