CHAPTER 8

Storing Data on the Device

Topics

- App Inventor Storage Components
- The Application Sandbox
- `File` component
- Retrieving a File
- `TinyDB`
- Tag-Value Pairs
- Storing a Tag-Value Pair
- Retrieving a Value
- Tag-Value Pairs When the Value is a List
- `TinyDB` Across Multiple Screens

At Inventor Storage Components

- Most of the applications that we create will have data that needs to be saved or *persisted*.
- Much data in the real world is stored in files or databases.
- App Inventor provides several ways to store and interact with data. The `File` component allows you to read and write files on your device.

At Inventor Storage Components

- The `TinyDBComponent` allows you to store a list of tag-value pairs.
- The `FusionTablesControl` allows you to interact with *Google's Fusion Tables*.
- `TinyWebDB` component allows you to store tag-value pair data.

The Application Sandbox

- Each Android application runs in its own isolated space, or *sandbox*.
- You can think of this sandbox as a protected folder on your device for each application.
- The `TinyDB` only stores information in the sandbox.
- By default, the `File` component stores files in the sandbox too. However, the `File` component can write to your devices SD card as well.

File Component

- App Inventor allows you to read, create, and modify files on your device.
- App Inventor's File component is a non-visible component. Once it is added to a project, its blocks are able to save, read, append to and delete files.

File Component

`File` Component Methods and Event

- The `File` component's methods include `SaveFile`, `AppendToFile`, `Delete` and `ReadFrom`.
- The `ReadFrom` method call invokes the single event in the `GotText` component.

File Component

Figure 8-1 File Component (Source: MIT App Inventor 2)

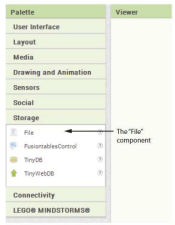


Figure 8-2 `SaveFile` Method (Source: MIT App Inventor 2)

File Component

`File` Component Methods and Event

- The `SaveFile` method call block saves a file.
- This is also the method used to create a new file.
- It's also important to note that if the file with the same name already exists, *this block will overwrite the file* with the new information.
- If you name a file *without* a preceding (/) The file will be saved in the application sandbox.
- If you put forward slash, it will be saved relative to the device's SD card.

File Component

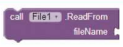
Figure 8-3 `Delete` Method (*Source:* MIT App Inventor 2)

- The `Delete` method deletes a file.
- You have to provide the filename. If the file name starts with a `/`, App Inventor will delete the file from the SD card otherwise it will delete the file from the application sandbox.

File Component

The `AppendToFile` method will add a text to the end of an existing file.

Figure 8-5 `ReadFrom` Method (*Source:* MIT App Inventor 2)



The `ReadFrom` method will open and read the contents of an existing file. It will invoke the `File.GotText` event handler, which will give you access to the contents of the file.

File Component

Figure 8-6 shows the `GotText` event handler. This block is used to process the file contents after the `ReadFrom` method is called.

Figure 8-6 `GotText` Event Handler (*Source:* MIT App Inventor 2)

File Component

See Figure 8-7 and note that you can change the `text` parameter name in the `GotText` block if you choose to.

Figure 8-7 Finding the `get text` Block (*Source:* MIT App Inventor 2)

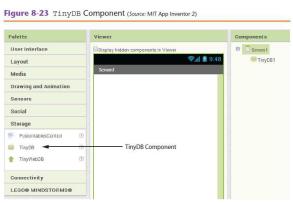
TinyDB

- App Inventor allows you to store data on your device by using a database called `TinyDB`.
- You store data in Tag-Value pairs.
- `TinyDB` will allow your app to store data and then retrieve it later.
- Each `TinyDB` can only be seen by the application it applies to.
- You cannot share the data store between two different applications.
- You should only have one `TinyDB` per app.

TinyDB

- Add `TinyDB` to your application by dragging the `TinyDB` component from the Storage Palette.
- It is a non-visible component.

Figure 8-23 `TinyDB` Component (*Source:* MIT App Inventor 2)



Tag-Value Pairs

- A Tag-Value pair consists of a tag and a value.
- The tag is used to identify the data item and the value is the data that you want to associate with the tag.
- The tag is a text item and the value can be any data type (number, text, Boolean, or list).

Tag-Value Pairs

You can picture a TinyDB as a table of tags and values.

Figure 8-24 TinyDB with Tag-Value Pairs (*Source:* MIT App Inventor 2)

Tag	Value
Mark Little	336-555-4343
Carrie Crum	919-555-1212
Patrick Jefferson	910-346-1818

Tag-Value Pairs

Each tag must be unique.

If you want to have two values, both a home and a mobile number, you would have to indicate that somehow in the tag.

Figure 8-25 Each Tag is Unique (*Source:* MIT App Inventor 2)

Tag	Value
Mark Little - Home	336-555-4343
Mark Little - Mobile	336-555-7668
Carrie Crum	919-555-1212
Patrick Jefferson	910-346-1818

Another way of adding multiple numbers to a tag would be to have a list as the value.

Storing a Tag-Value Pair

- The `TinyDB.StoreValue` block stores data.
- Once you've added a `TinyDB` component to your project, you find this block in the *My Blocks* -> *TinyDB1* drawer.

Figure 8-26 `StoreValue` Block (*Source:* MIT App Inventor 2)

Retrieving a Value

The `TinyDB.GetValue` *block*, which takes in the `tag` as parameter and will return the value that is stored in the `TinyDB` for that tag.

Figure 8-29 Retrieving a Value (*Source:* MIT App Inventor 2)

Retrieving a Value

- This figure shows how to use the `TinyDB1.GetValue` block to retrieve the value for Mark Little.
- The "valueIfTagNotThere" slot gives you the ability to default a value if the tag happens to not be in the TinyDB.

Tag-Value Pairs when the Value is a List

- We can create a list and store it as the value of the tag. This will allow us to store more information about our contact in our database.
- To demonstrate this we will create another "Contact" application gathering the contact's name, home phone, mobile phone, and email.
- It will be important to create each contact's list in a consistent manner. The first element in each list should always be the home phone, the second should be the mobile phone, etc.

`TinyDB` Across Multiple Screens

- Storage and media components are shared on the application level and shared between screens.
- Each application has a single storage space that is isolated from the other applications.
- However, when you add media or storage components to an application they are visible to all screens.

`TinyDB` Across Multiple Screens

- The `ListPicker`'s elements will be populated from the `TinyDB` on the first screen.
- The picture and sound files will be retrieved from the same `TinyDB` on the second screen.

Figure 8-44 Get All Tags (Source: MIT App Inventor 2)

TinyDB Across Multiple Screens

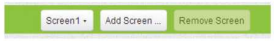
Figure 8-45 Clear All (Source: MIT App Inventor 2)

The `ClearAll` block for the `TinyDB` that will clear all elements from the storage space.

TinyDB Across Multiple Screens

Adding a Second Screen

Figure 8-46 Add Screen (Source: MIT App Inventor 2)



- To add a second screen click the *Add Screen* button at the top of the designer or blocks editor.
- Once you click the *Add Screen* button, you will be prompted to name the new screen. Name it something meaningful.
- You can, and probably should, rename additional screens however, you cannot rename `Screen1`.

TinyDB Across Multiple Screens

Adding a Second Screen

- Each screen has its own unique designer and blocks editor spaces.
- You can switch between the two.

Figure 8-48 Switching between Screens during Development (Source: MIT App Inventor 2)

TinyDB Across Multiple Screens

Adding a Second Screen

- There are a few noteworthy blocks that allow you to programmatically navigate between screens at runtime.
- The first block allows you to load a different screen.
- The second block allows you to send the start value to the screen.
- The close screen block will allow you to close the current screen and return to the previous screen.

Figure 8-49 Navigating between Screens at Runtime (Source: MIT App Inventor 2)

`TinyDB` Across Multiple Screens

Adding a Second Screen

- Applications with multiple screens work best when deployed to a device. Tutorial 8-7 will work best if you download your project's .apk file.
- To learn more about deploying the .apk to your device, see http://appinventor.mit.edu/explore/ai2/share.html