Titan Robotics Club

**Scouting App Code Generator**

User’s Guide

**Contents**

Overview 2

Launching the Code Generator 3

Loading and Creating Session Data 4

Fields 5

CSV Bindings 6

Pages 7

Elements 8

App Info 9

Generating and Exporting Java Code 10

Integration w/ Android Studio 11

**Overview**

The TRC Scouting App is a software application written for the Android™ family of mobile operating systems, to allow competitive robotics team members to collect individual team performance data during match play to allow for optimized alliance selection based on team performance data analysis. TRC Scouting App supports Android versions 4.4 and above. Data collected by the TRC Scouting App is exportable in comma-separated value (CSV) format, which is compatible with a variety of data-analysis tools such as Microsoft Excel™ and Tableau® software.

The modular structure of TRC Scouting App’s codebase allows for the app to be reused for different competitions easily. The Android Studio integrated development environment allows for visual, drag-and-drop customization of graphical form layouts. Additional coding in the Java programming language is needed to parse and serialize data collected from such graphical form layouts. The Scouting App Code Generator is designed to assist novice programmers in writing the Java code that accompanies the graphical layouts which is necessary to create a complete, functional Android application.

**Launching the Code Generator**

The Scouting App Code Generator supports the Microsoft Windows, OS X, and GNU/Linux operating systems, owing to its Java-based compile once, run anywhere architecture.

**Installing Prerequisites**

The Code Generator requires version 1.8 or higher of the Java Runtime Environment to run. To check if Java is installed and your PATH variable is set properly, follow the instructions below corresponding to your operating system.

**Microsoft Windows**

1. Open Command Prompt.
2. Type the command java -version. Press Enter.
3. If the Java version displayed is at least 1.8, Java is installed. Stop.
4. If the Java version displayed is below 1.8 or the command is not recognized, install the Java runtime environment here: <https://java.com/en/download/>
5. Add the Java installation directory to the PATH variable. Instructions are here: <https://java.com/en/download/help/path.html>

**OS X or Linux**

1. Open Terminal.
2. Type the command java -version. Press Enter.
3. If the Java version displayed is at least 1.8, Java is installed. Stop.
4. If the Java version displayed is below 1.8 or the command is not recognized, install the Java Runtime Environment for OS X here: <https://java.com/en/download/help/mac_install.html>

Install the Java Runtime Environment for Linux here:

<https://java.com/en/download/help/linux_install.html>

**Launching the Scouting Application**

To launch the Scouting App Code Generator on Windows, navigate to the Code Generator directory and double-click on the executable Batch file, “launch.bat”, and a command window should open with the Code Generator’s text-based interface ready. To launch the Scouting App Code Generator on OS X or Linux, open a Terminal instance and navigate to the Code Generator directory, then run the command:

./launch.sh

to launch the Code Generator.

**Loading and Creating Session Data**

The Scouting App Code Generator stores templates which complete Java code can be generated from, in session files that can be saved and loaded from disk for easy revision of these code layouts. These session files contain JSON-formatted data representing abstract structures that the Code Generator serializes to generate resultant Java code: match data fields and their corresponding CSV column mappings, pages containing lists of graphical elements which are mapped to the preceding data fields, and application information constants.

To load an existing session file in the same directory as the Code Generator, given the filename, run the following command in the Code Generator:

load <filename>

For example, if the session file is named “session.dat”, run the following command:

load session.dat

To create a new session file and load it in the same directory as the Code Generator, given the filename, run the following command in the Code Generator:

create <filename>

For example, if the session file is named “session.dat”, run the following command:

create session.dat

**Fields**

The TRC Scouting App stores the data collected for each match using a Java class called “MatchInfo”, which contains fields consisting of a field name and field type. The Code Generator can generate the Java code for the MatchInfo class dynamically, given a list of field names and their associated field types. The Code Generator supports four field types: int, double, bool, and str. An int refers to a 32-bit signed integer. A double is a double-precision 64-bit floating point. A bool is a Boolean value (true or false). A str is a string, referring to an alphanumeric combination. Below are the commands for manipulating MatchInfo fields with the Code Generator.

The following command lists the fields in the current session:

lsfields

The following command adds a field with a given name to the current session:

field add <name> <type>

The following command renames a field from its old name to the new name given:

field rename <oldname> <newname>

The following command changes the field type of a field given the name:

field settype <name> <type>

The following command deletes a field given its name:

field del <name>

**Field Flags**

Field flags are attributes added to a field that the Code Generator uses to generate specialized code for specialized fields. For example, if a field is marked with the flag must\_be\_filled, any textbox assigned to that field will require the user to enter text into the box before the data collected for the match containing that field is made savable.

Field flags can only be assigned to fields of supported types. Here’s a list of supported field flags and their supported types:

alliance\_type (str), match\_num (int), must\_be\_filled (str)

match\_type (str), match\_num (int)

The following command adds a flag to a field:

field flag add <name> <flag>

The following command removes a flag from a field:

field flag del <name> <flag>

The following command lists the flags assigned to a field:

Field flag list <name>

**CSV Bindings**

As mentioned earlier, the TRC Scouting App exports collected match information in CSV (comma-separated value) format for easy compatibility with data-analysis software such as Tableau and Microsoft Excel. The order of the fields as they are exported in CSV format is configurable through the Code Generator. Below are the commands for mapping and viewing mappings of MatchInfo fields to CSV columns.

To list the current mappings of CSV columns to fields, run the command:

csv list

To map a field to a CSV column number, run the command:

csv map <column no#> <field name>

To unmap a CSV column number, run the command:

csv unmap <column no#>

**Pages**

In the Code Generator, a page is the abstract representation of the components that make up a page in the Scouting App’s tabbed data-entry layout. Pages contain the displayed name of the page in the tabbed layout, the Android R(esource) ID of the Fragment view corresponding to the page, the class name of the Fragment class instance to generate, the page number of the page in the tabbed layout, and the list of Elements (more in next page) contained in the page.

To list the current pages in the session, run the command:

lspages

To add a page, run the command:

page add <tabname> <classname> <fragmentname> <pagenum>

Where tabname is the displayed name of the page in the tabbed layout, classname is the class name of the Fragment class instance to generate from the page, fragmentname is the Android resource ID of the fragment layout to base the page on, and pagenum is the page number of the page in the tabbed layout.

To remove a page, run the command:

page del <classname>

**Elements**

In the Code Generator, an element is the abstract representation of the child Views of a Fragment class from which match data fields are parsed. Elements are mapped to the field which the data inside the element is parsed to, and contain the Android resource ID of the child View of the Fragment the element represents. Pages contain a list of Elements within the page. As different types of elements have different types of parsable data, each element type can only be mapped to certain field types. Here’s a list of element types and supported field types the element types can be mapped to:

edittext (string, double, int), checkbox (bool), numberpicker (int), spinner (string), switch (bool)

To edit the list of elements within any page, first set the current working page with the command:

wp set <classname>

After the current working page is set, the elements in the current working page can be listed with the command:

elements list

To add elements to the current working page, use the following command:

elements add <Android R-id> <type> <fieldname>

To delete an element from the current working page, use the following command:

elements del <Android R-id>

To unset the current working page, run the following command:

wp unset

**App Info**

The appinfo command sets attributes of the code to be generated, which are the year number of the robotics game the app is designed for, and the CSV header string that the app exports recorded match data with. Both must be populated to allow code generation.

To list the current appinfo settings for the session, run the command:

appinfo list

To set the CSV header of the appinfo settings, run the command:

appinfo set csv\_header “your csv header here”

To set the year number of the appinfo settings, run the command:

appinfo set year\_number <year\_num>

**Generating and Exporting Java Code**

After all fields and CSV mappings are finalized, page/element configurations are finished, and app info settings are configured, the Code Generator can generate Java code.

To generate the Java code from the session data, run the command:

generatecode

Generated code should be found in the “output” directory, in the same directory as the jar file of the Code Generator. To export this code, copy the contents of the “output” directory into the “app/src/main/java/trc3543/trcscoutingapp” directory of the TRC Scouting App Android project, replacing any files as needed with the new files from “output”.

**Integration w/ Android Studio**

1. Using Android Studio, create fragments in the “res/layout” directory of the Scouting App Android project. Add necessary child Views (EditText, CheckBox, etc.) to fragments. Add a confirm button with onClick method: SetMatchInfo -> confirmTypes to the fragment to be used for the last page.
2. Launch the Code Generator. Create a new session file.
3. Add all fields in Code Generator for the match data to be collected.
4. Add CSV bindings to fields. Set AppInfo CSV header.
5. Add pages corresponding to fragments added in step 1, including the fragment’s Android Resource ID, page number, and Fragment class name.
6. Add elements corresponding to child Views added in step 1 to each page corresponding to a fragment, including each View’s Android Resource ID.
7. Set year number in AppInfo.
8. Generate and export code.
9. Exit code generator with command “exit”.