# Getting started with ANTLR4

# Set up the environment

1. Open up terminal and copy the following commands
wget http://ati.ttu.ee/~hkinks/antlr/antlr.sh
sh antlr.sh

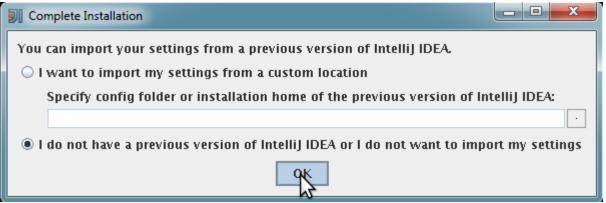
It will download and execute a script which downloads ANTLR4 library, plugin for Intellij IDEA and sets up necessary aliases.

The library and the plugin can be found in 'workspace' directory after running the script.

2. Next open Intellij IDEA where the development will take place

```
cad
idea &
```

#### 3. Press OK in the following dialog window



4. First let's set up the ANTLR plugin to make code generation more convenient. In the Intellij IDEA

🔊 IntelliJ IDEA	- <b>-</b> X
Welcome to IntelliJ IDEA	
Recent Projects Quick Start	
Create New Project	
No Project Open Yet	
Open Project	
Check out from Version Control	
Configure	\$
Docs and How-Tos	⇒
Intellij IDEA 12.1.4 Build 129.713. Check for updates now.	

## 5. Choose Plugins.

🔊 IntelliJ IDEA		- <b>·</b> ×
<b>J</b> We	lcome to IntelliJ IDEA	
Recent Projects	¢ Configure	
No Project Open Yet	Settings   Plugins   Import Settings   Export Settings   Create Desktop Entry   Project Defaults	¢
Intellij IDEA 12.1.4 Build 129.71	3. Check for updates now.	

## 6. Click Install plugin from disk...

I Plugins		
🕼 🛓 🏦 🗯 Show: All plugins 🔻		<u>ଚ</u>
🗹 💉 Android Designer	Bundled	Drevides visual adjiting support for Andraid la put files
🗹 💉 Android Support	Bundled	Provides visual editing support for Android layout files.
🗹 💉 Ant Support	Bundled	
🗹 💉 Byte Code Viewer	Bundled	
🗹 💉 Commander	Bundled	
🗹 💉 Copyright	Bundled	
🗹 💉 CVS Integration	Bundled	
🗹 💉 Eclipse Integration	Bundled	
🗹 💉 GenerateToString	Bundled	
🗹 💉 Git Integration	Bundled	
🗹 💉 GitHub	Bundled	
🗹 💉 Gradie	Bundled	
🗹 💉 Groovy	Bundled	
🗹 💉 hg4idea	Bundled	
🗹 💉 I18n for Java	Bundled	
🗹 💉 Inspection Gadgets	Bundled	
🗹 💉 IntelliLang	Bundled	
🗹 💉 Intention Power Pack	Bundled	
🗹 💉 JavaFX	Bundled	
🗹 💉 JUnit	Bundled	
🖌 📝 Maven Integration	Bundled	
🗹 💉 Plugin DevKit	Bundled	
🗹 💉 Properties Support	Bundled	
🗹 💉 Subversion Integration	Bundled	
🗹 💉 Task Management	Bundled	
🗹 💉 TestNG-J	Bundled	
🗹 💉 UI Designer	Bundled	
🗹 💉 UI Designer (Core)	Bundled	
🗹 💉 XPathView + XSLT Support	Bundled	
🔽 🔌 XSLT-Debuaaer	Bundled	
Install letBrains plugin Browse repositories	Install plugin	from <u>d</u> isk
		N2
		OK Cancel Apply Help

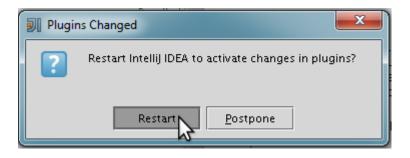
7. Choose the plugin. When running the bash script in the first section, it should have downloaded it into workspace directory in your home folder: /home/INTRA/<uri-id>/workspace/plugin.zip

Choose Plugin File	
JAR and ZIP archives are accepted	
🔗 🍠 🕞 🖙 🗙 💋 📖 🛛 Hide pat	h
/home/INTRA/hannes.kinks/workspace/plugin.zip	
🕀 🗖 bin	
🛨 🗖 boot	
🛨 🛅 cad	
⊕ E dev	
⊕ 🔁 eda	
⊕ etc	
□ □ home	
🔁 🛅 gritten.pahn	
- hannes.kinks	
🖃 🗖 workspace	
plugin.zip	
Drag and drop a file into the space above to quickly locate it in the tree.	
OK Cancel Help	

8. Click OK and the plugin should appear in the plugins list.

I Plugins		
💋 🛓 🏦 🎼 Show. All plugins 🔻		Q. Q
<ul> <li>Android Designer</li> <li>Android Support</li> <li>Ant Support</li> <li>Ant Support</li> <li>ANTLR v4 grammar plugin</li> &lt;</ul>	Bundled Bundled Custom Bundled Bundled Bundled Bundled Bundled Bundled Bundled Bundled Bundled Bundled Bundled Bundled	This plugin is for ANTLR v4 grammars. Features: syntax highlighting, syntax error checking, semantic error checking, navigation window, live preview, parse tree view, profiler, goto-declaration, find usages, rename tokens, rename rules. Generates code in /gen/package/YourGrammarRecognizer.java unless you override in the configuration dialog. Shortcut to generate parsers is ctrl-shift-G / meta-shift-G but it's in Tools menu, popups. Code completion for tokens, rule names. finds tokenVocab option for code gen if there is a tokenVocab option, don't warn about implicit tokens. shortcut conflicted with grammar-kit plugin. Has live grammar interpreter for grammar preview. Right click on rule and say "Test ANTLR Rule". Changes to grammar seen in parse tree upon save of grammar. Works with Intellij 13.x and requires 12.1.x. You can configure the ANTLR tool options per grammar file; right-click in a grammar or on a grammar element within the
<ul> <li>✓ Ø 118n for Java</li> <li>✓ Ø Inspection Gadgets</li> <li>✓ Ø IntelliLang</li> <li>✓ Ø Intention Power Pack</li> <li>✓ Ø JavaFX</li> <li>✓ Ø JUnit</li> <li>✓ Ø Maven Integration</li> </ul>	Bundled Bundled Bundled Bundled Bundled Bundled Bundled	structured view. When you change and save a grammar, it automatically builds with ANTLR in the background according to the preferences you have set. ANTLR tool errors appear in a console you can opened by clicking on a button in the bottom tab. You can use the meta-key while moving the mouse and it will show you token information in the preview editor box via tooltips.
<ul> <li>Plugin DevKit</li> <li>Properties Support</li> <li>Subversion Integration</li> <li>Task Management</li> <li>TestNG-J</li> <li>U Designer</li> <li>U U Designer (Core)</li> <li>XPathView + XSLT Support</li> </ul>	Android Designer       Bundled         Android Support       Bundled         Android Support       Bundled         Ant Support       Bundled         Byte Code Viewer       Bundled         Commander       Bundled         Copyright       Bundled         Coty Stream       Bundled         Coty Stream       Bundled         Cory Stream       Bundled         Stream       Bundled         Generate String       Bundled         Grin Hub       Bundled         Struate Stream       Bundled         S	

9. Restart when it requires it.



## **10.** Start off with a new project.

🔊 IntelliJ IDEA		
<b>J</b> We	lcome to IntelliJ IDEA	
Recent Projects	Quick Start	
	Create New Project	
No Project Open Yet	Import Project	
	Open Project	
	Check out from Version Control	
	and Configure	\$
	Docs and How-Tos	⇔
Intellij IDEA 12.1.4 Build 129.713	. Check for updates now.	

11. Name your project and set it to be located at the workspace in your home folder. Next specify project SDK.

🔊 New Project		X
Java — Java — Java — Java — Java — Java Module	Project <u>S</u> DK: Description — Java module desktop an applications	workspace         Iome/INTRA/hannes.kinks/workspace         Image:
	<u>P</u> revious	Next Cancel Help

12. Locate to /opt/JDK17 and select it.

Street Home Directory for JDK	×
A JI ⊑   ⊑ ×   ⊈   ■	Hide path
/opt/JDK17	<b>Ľ</b>
🕀 🛅 media	
🛨 🗖 mnt	
🛨 🛅 net	
E opt	
🕀 🗖 asdk	
🕀 🗖 eclipse	
⊕ idea-IC-129.713     □	
□ - <b>□</b> JDK17	
🗄 🗖 bin	
🕀 🔁 db	
😐 🗖 include	
🕀 🗖 jre	
🗈 – 🛅 lib	
🗄 🛅 man	
æ kde3	
🕀 🛅 microchip	
🕀 🛅 mplab	
Drag and drop a file into the space above to quickly locate it in the t	ree.
OK Cancel	Help

## 13. When everything is set, press Next

🔊 New Project		×
Java         Java Module         Maven Module         JavaFX Application         Command Line App         Groovy         Mathematical Command Line App         Application Module         Project         Dava Project         Samples Gallery         Samples Gallery         Java Hello World	Project name:       tutorial         Project location:       me/INTRA/hannes.kinks/wo         Project SDK:       Image: 1.7 (java version "1.7.0.1)         Description	orkspace/tutorial 25") ▼ Ne <u>w</u> JVM-based ng applications
	Previous Nex Cance	el Help

## 14. Finish project creation.

I New Project				×
Please select the desired techn	ologies.			
This will download all needed li	braries and cr	eate Facets in project (	configuration.	
🔲 🌀 Groovy	Use library	[No library selected]	Create	
	Error library	is not specified		
	Litor. norary	is not specifica		
	<u>P</u> rev	ious <u>F</u> inish	Cancel	Help

# CSV parser tutorial

Now that the environment is set up, we can continue with making a simple CSV parser as a first sample project. The goal would be to have our program accept valid CSV file and parse its contents. What should the program do with the parsed data is up to the application. For an example it could be converted to JSON format and outputted to a file.

Finished project is available in GIT repository <a href="http://git.hkinks.com/hkinks/csv-tutorial">http://git.hkinks.com/hkinks/csv-tutorial</a>

#### Input

The input should be a regular comma (",") separated file. Contents can be made up or a book list example can be downloaded from <u>http://ati.ttu.ee/~hkinks/antlr/input.csv</u>.

#### Grammar

To get started we should describe the grammar.

## 1. Create a new file with .g4 extension.

列 csv - [E:\De	ownloads\csv] -	IntelliJ II	DEA 14.0.	2						
<u>F</u> ile <u>E</u> dit <u>V</u>	<u>(</u> iew <u>N</u> avigate	<u>C</u> ode	Analy <u>z</u> e	<u>R</u> efactor	<u>B</u> uild	R <u>u</u> n	<u>T</u> ools	VC <u>S</u>	<u>W</u> indow	
📑 csv 👌 🗖	src									
Project	• ÷	÷ 🛱	+ [←							
🔻 📑 csv (E:)	\Downloads\cs\	/)								
🕨 🗖 🔳	ea									
🗖 src							Java Cla			
J csv						B	File	155		
	₩ Cu <u>t</u>				Ctrl+X	_	Packag	J		
Externa	[]] <u>С</u> ору				Ctrl+C	_	packag		iava	
	C <u>o</u> py Path				Shift+C		HTML		,	
	Copy Refer	rence		Ctrl+Alt+		_	XSLT St		~ <b>t</b>	
	<u>P</u> aste				Ctrl+V	<b></b>				
	Find <u>U</u> sage				Alt+F7	-	Edit File		olates	
	Find in <u>P</u> at				Shift+F		GUI For	m		
	Repl <u>a</u> ce in	Path		Ctrl+	Shift+R		Dialog			
	Analy <u>z</u> e		10014		IDEA 4	•	Form Si	napsho	ot	
	ownloads\csv] -			-			<b>T</b> 1	1100	14C 1	
×	iew <u>N</u> avigate	<u>.</u>	Analy <u>z</u> e	<u>R</u> efactor	Build	K <u>u</u> n	<u>1</u> 00ls	VC <u>S</u>	Window	<u>H</u> elp
	src 🖉 🖉 CSV.g4	$\rangle$								
Project	• 😳	÷   ₩	- ∥← _ (	CSV.g4 >	c					
	Downloads\csv	r)								
🕨 🖿 .ide	a									
V 🗖 src			_							
	CSV.g4									
J csv										
inp	ut.csv I Libraries									
	LIVIAILES								Ŧ	
									Ι	

2. Write grammar. Note that the grammar file name must match with the name specified in the first line. For example: 'CSV.g4' and 'grammar CSV;'

3. Next step is to let ANTLR generate code. Based on the grammar it will create tokens and parser/lexer classes. Right click on the grammar file and choose Generate ANTLR recognizer...

*Side note*: By default ANTLR generates code for Listener design pattern. However if you wish to use Visitor pattern instead, you can configure ANTLR to do so by choosing Configure ANTLR... and choosing generate parse tree visitor.

🔊 csv - [E:\Downlo	ads\	(csv] -	[csv]	\src\CS	V.g4	- Intelli	IDEA	14.0.2				
<u>File Edit V</u> iew	<u>N</u> av	igate	<u>C</u> ode	Analy <u>z</u>	e <u>R</u>	efactor	<u>B</u> uild	R <u>u</u> n	<u>T</u> ools	VC <u>S</u>	<u>W</u> indow	<u>H</u> elp
$\square$ csv $\square$ src $\square$	00	SV.g4	$\rangle$									
🗗 Project 🔹 🔻		Θ	*   #	F≖ ∥←	0	CSV.g4	×					
CSV (E:\Down	nloa	ds\cs\	/)			gramma	r CSV	;				
🕨 🗖 .idea						filer	handa	. lin/				
				_		file: header			;+;			
CSV.g	1	New							•			
input.csv	Ж	Cut						Ctrl+X	ntry	)* '\	r'? '\n'	;
External Libra	_		,					Ctrl+C	ext			
		С <u>о</u> ру	Path				Ctrl+S	hift+C	trin	g		
		Сору	Referen	nce		Ctrl	+Alt+S	hift+C	mpty			
	đ	<u>P</u> aste	2					Ctrl+V				
	D	Jump	to Sou	rce				F4				
		Find	<u>U</u> sages					Alt+F7	~'	"')	* '"';	
		Analy	y <u>z</u> e						•			
		<u>R</u> efac	tor						•			
		Add	to F <u>a</u> vor	rites					•			
		<u>R</u> efor	rmat Co	de			Ctrl	+Alt+L				
		Optir	mi <u>z</u> e Im	ports			Ctrl+	Alt+0				
	×	<u>D</u> elet						Delete				
		Mark	as Plair	n Text					_			
			e <u>M</u> odul									
		Com	pil <u>e</u> 'CS'	V.g4'			Ctrl+S	hift+F9	_			
	-		l <u>H</u> istory						•			
	Ø	-		'CSV.g4'								
			in Expl	orer								
		File <u>P</u>					Ctrl+A	lt+F12				
	9		te Gist									
				TLR Rec	ogni	Zex	Ctrl+S	Shift+G				
		Conf	igure Al	NTLR	1							

4. As a result we should get a new folder with the name gen which contains the ANTLR generated files. Intellij IDEA newer versions should recognize the folder by itself that it contains source files.

However, if the folder does not get marked with a blue icon, you have to do it yourself by right clicking on it, choosing Mark Directory As and choosing either Sources Root or Generated Sources Root.

	File <u>P</u> ath	Ctrl+Alt+F12	
	Compare Directory with	Ctrl+D	
	Mark Directory As	•	🔁 Sources Root
Ø	Create Gist		Test Sources Root
-	Generate ANTLR Recognizer	Ctrl+Shift+G	📑 Resources Root
	Configure ANTLR		📴 Test Resources Root
			Excluded
			🖙 Generated Sources Root
			65

- 5. Make sure that the classes in the generated directory do not have any unresolved dependencies. If they do, Intellij probably does not have ANTLR library properly configured.
- 6. Next we should create a Main class

🔻 🗖 src 👔		
۵c	New	) C Java Class
🗾 csv.i	<mark>Ж Си<u>t</u></mark>	Ctrl+X E File
🗐 inpu	🖺 <u>С</u> ору	Ctrl+C Package
💼 External		Ctrl+Shift+C package-info.java
🗐 Create	New Class	
<u>N</u> ame:	Main]	†↓
<u>K</u> ind:	Class	
		OK Cancel

7. We should start off by describing where our input file is, opening it and creating an InputStream. If it fails we can throw an exception saying that the file was not found.

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
public class Main {
    public static final String INPUT_CSV = "input.csv";
    public static void main(String[] args) {
        try {
            FileInputStream fileInputStream = new FileInputStream(INPUT_CSV);
    }
}
```

```
} catch (FileNotFoundException e) {
    System.err.println("Input file not found.");
    return;
    }
}
```

8. The input can now be given to ANTLR lexer, CSVLexer that was generated during the 3rd step where we let ANTLR generate code for us.

```
CSVLexer lexer;
try {
    lexer = new CSVLexer(new ANTLRInputStream(fileInputStream));
} catch (IOException e) {
    e.printStackTrace();
    return;
}
```

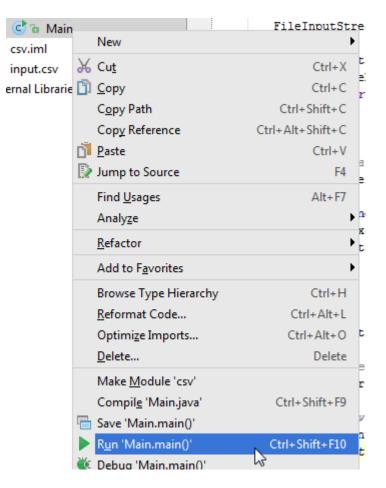
9. Lexer's result will be a stream of tokens which can be passed to parser.

```
CommonTokenStream tokens = new CommonTokenStream(lexer);
CSVParser parser = new CSVParser(tokens);
```

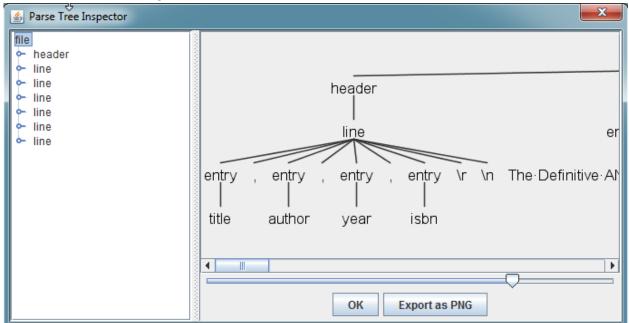
10. As an intermediate step we can let ANTLR draw us a Parse Tree Diagram, which is good for debugging and making sure the grammar was properly parsed.

```
ParserRuleContext ruleContext = parser.file();
Trees.inspect(ruleContext, parser);
```

11. Run the code to see the parse tree.



12. You should see something like this:



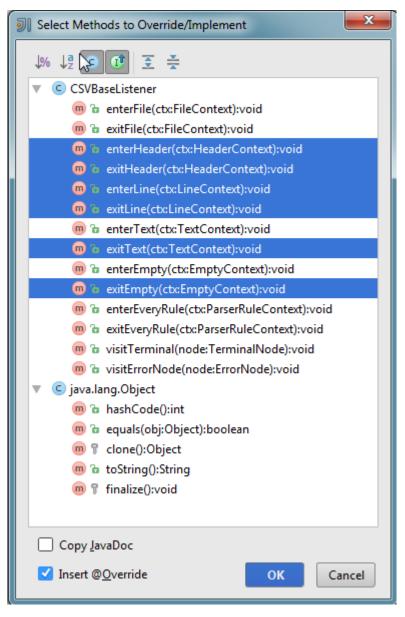
**13.** Now if we wish to actually do something with the parsed input, we should implement a listener class that would extend BaseListener class. Create a new class named *CSVLoader* and make it

public class CSVLoader extends CSVBaseListener					
}	Copy Reference	Ctrl+Alt+Shift+C			
3	🕺 <u>P</u> aste	Ctrl+V			
	Past <u>e</u> from History	Ctrl+Shift+V			
	Paste S <u>i</u> mple	Ctrl+Alt+Shift+V			
	Column Selection <u>M</u> ode	Alt+Shift+Insert			
	Find <u>U</u> sages	Alt+F7			
	Analy <u>z</u> e	•			
	<u>R</u> efactor	•			
	Folding	•			
	Go To	•			
	Generate	Alt+Insert			
	Compil <u>e</u> 'CSVLoader.java'	Ctrl+Shift+F9			

14. By extending the CSVBaseListener we can override methods of interest. Right click in your class, choose generate and from there Override Methods...



15. The methods what would interest are for example where the parser enters and exits different contexts specified in the grammar.



16. For memorizing the data we can create some Lists that would contain the rows of the csv file. As in the example file we also have a header row, we can add it to a different list and finally associate each entry with its header (we are using map for this).

```
List<Map<String, String>> rows = new ArrayList<Map<String, String>>();
List<String> header;
List<String> currentRow;
```

17. Every time we enter a line we create a new row object and when we exit the line, we can add the elements met along the way into it and finally add the row into the list of rows.

```
@Override
public void enterLine(CSVParser.LineContext ctx) {
    currentRow = new ArrayList<String>();
}
@Override
public void exitLine(CSVParser.LineContext ctx) {
    if (ctx.getParent().getRuleIndex() == CSVParser.RULE_header)
        return;
    Map<String, String> m = new LinkedHashMap<String, String>();
    int i = 0;
    for(String v:currentRow){
        m.put(header.get(i),v);
        i++;
    }
    rows.add(m);
}
```

18. And in case the line is a header, we add it to the list of headers instead of lines.

```
@Override
public void exitHeader(CSVParser.HeaderContext ctx) {
    header = new ArrayList<String>();
    header.addAll(currentRow);
}
```

19. Lines are however not terminal nodes and we should also specify what happens when we go through these. In our case terminal nodes can be either empty or some text.

```
@Override
public void exitEmpty(CSVParser.EmptyContext ctx) {
    currentRow.add("");
}
@Override
public void exitText(CSVParser.TextContext ctx) {
    currentRow.add(ctx.TEXT().getText());
}
```

20. Now coming back to our main class we should walk the tree, each time our specified nodes are entered/exited our Loader class will gather the information. The tree drawing code we can comment out for now

```
ParseTree tree = parser.file();
ParseTreeWalker walker = new ParseTreeWalker();
CSVLoader loader = new CSVLoader();
walker.walk(loader,tree);
```

21. Finally we can simply print out the contents of the list for now. However, from this step on, we could use this data to convert it to some other format for example.

```
for(Map<String,String> row:loader.rows)
    System.out.println(row.values());
```

#### 22. The output should look something like this in our case:

[The Definitive ANTLR 4 Reference, Terence Parr, 2013, 1934356999] [The Hobbit, J.R.R. Tolkien, 2002, 0618260307] [A Game of Thrones (A Song of Ice and Fire), George R.R. Martin, 2005, 0553588486] [The Hunger Games, Suzanne Collinns, 2008, 0439023483] [To Kill a Mockingbird, Harper Lee, 2006, 0061120081] [Animal Farm, George Orwell, 2003, 0452284244]