

Code Generation: The One Page Guide

Your introduction to the critical programming technique of the next decade



Take a look at the decision tree on the flip side of this paper. Why are there more code generators for Java than all of the other technologies combined? Java is very popular but that is not the sole reason. Java is easy to generate, and the class libraries for Java tend to be code-intensive to use. Engineers understand that they can work smarter rather than harder in the Java environment by using code generation.

As with the techniques of Object Oriented Programming and Design Patterns before it, the technique of code generation is becoming a critical skill in successful software development.

What is code generation?

Code generation means using a program to help you write your programs. Just like the programs we write for our clients helps them to be more efficient, we can write programs to help us write higher quality code more quickly.

Code generators read a specification of abstract requirements as input, often as an XML file. Using templates, the generator then builds one or more output files based on the requirements.

A *passive* generator (e.g. a Wizard) builds code which you then modify. An *active* generator builds code and then maintains it as the project evolves, just as another member of your development team maintains ownership over a piece of code. The majority of the generators listed on the flip side of the paper are active.

Code generation advantages

Code generation isn't just about writing code quickly. There are four key advantages to using code generators:

- **Quality** – Code generators use templates to build the code specified by the input parameters. The better the templates, the better the output code. As you work on the templates to increase their quality, the quality of the overall code base is improved.
- **Consistency** – The class, methods and variable naming of the output code is completely consistent. This makes the interfaces easy to use, and easier to layer more generated code on top of.
- **Productivity** –The productivity gains of code generation lie in the agility of the code generators to rebuild the code base to suit the changing requirements of the project. With one generation pass you can add or remove large segments of code.
- **Abstraction** – Generators that store the input requirements (such as the database schema or the user interface design) in a completely language neutral form offers advantages based on that abstraction. For example, with a change of the templates these generators can be building for EJB one day and JDO next. This is tangible portability. Another benefit is moving the business rules and structure of the application into a form that can be reviewed by individuals outside of the engineering staff.

Finding the right code generator

On the flip side of this page we have provided a decision tree that will lead you to the right generator for your project. After that you can go to www.codegeneration.net to find links to the generator sites and interviews with their authors.

If none of the generators listed directly address your needs you can use one of the 'custom code generators' which support scripting in a variety of languages. Failing that you can always write one yourself.

Code generation concerns

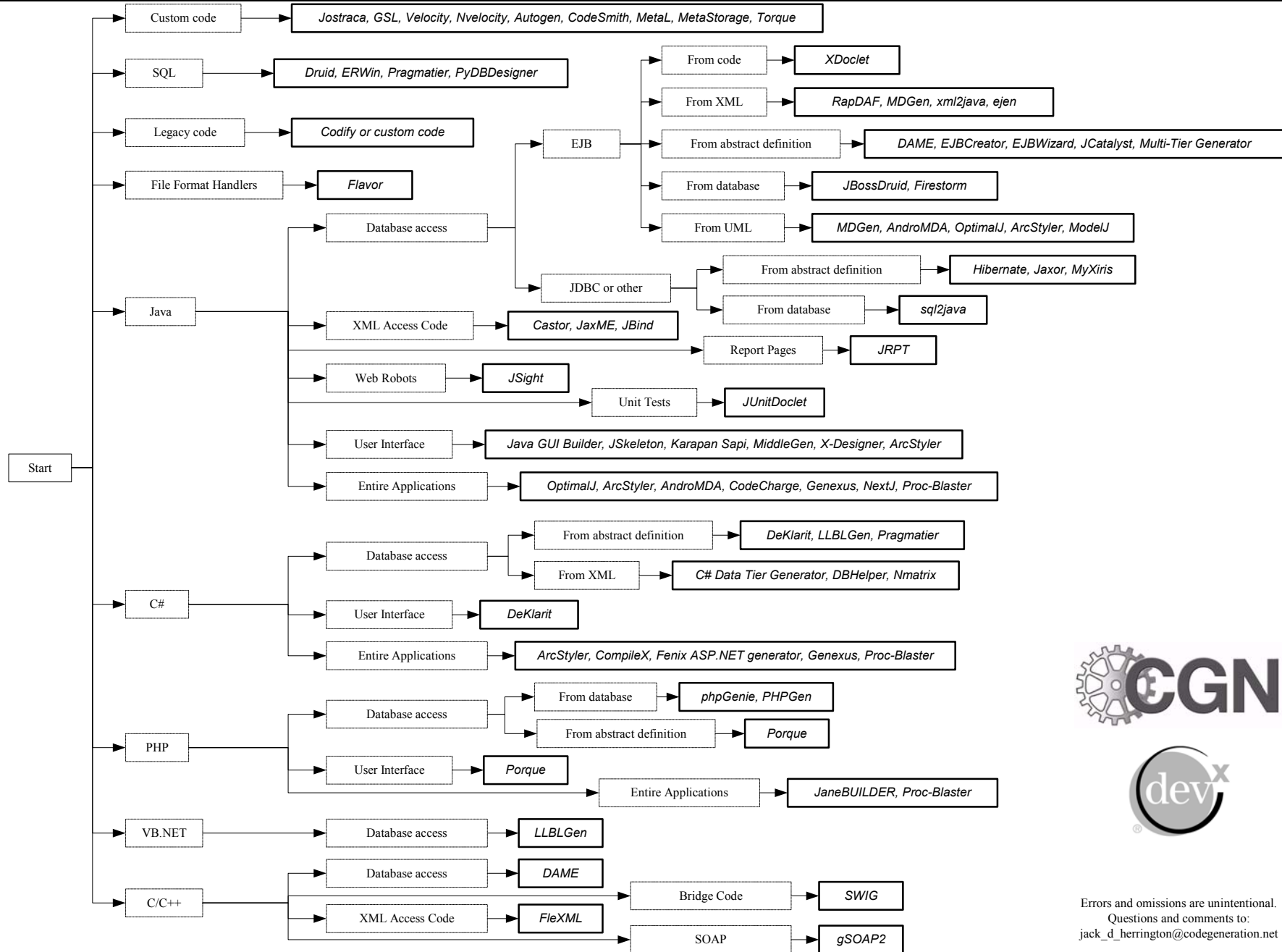
Code generation, like any technique, has both pros and cons. Some of the more prominent concerns are: code generators have fallen into disrepair, engineers avoid code generators, and that code generators are too complex. We address these concerns and others on the CGN web site (www.codegeneration.net).

Code generation resources

“*Code Generation In Action*” is a book about code generation theory, technique and implementation that is being released by Manning in July and is available for pre-order today.

www.codegeneration.net is an online resource for news, articles and interviews about code generation.

Code Generation: Decision Tree



Errors and omissions are unintentional.
 Questions and comments to:
jack_d_herrington@codegeneration.net