liaw - c++ text template engine - day 1 notes

Tuesday, May 13, 2014
5:29 AM

Design Goals (morning session)
- Integrate STL well
- Type Safety
- Template parsing error handling
- Provide customization of markup language
- Make configuration simple
- Any data structure convertible to text should work

Features
- "Compiled templates" (for speed)

Design Decisions
- Use C++11

Key Design Discussion
- Call the internal program data representation "the model"
    - Model is applied to the template to produce output
    - 2 primary approaches
        - engine model forces user to conform
            - Typically it's a hash of strings to strings
            - And a collection of hashes for 'sections'
            - Disadvantage: copy all your data into model before rendering
        - engine accepts user defined model - mapping internal
            - No engines take this approach so far
            - User data copy not required
- Boost library use/integration
    - Possibly us boost::qi for doing formatting
    - Could use boost::spirit for parsing or maybe regex is enough
    - What if model is already boost serializable with name value pairs
        - Could the engine emulate an output Archive type for the objects?
- Data formatting is an issue
    - ctemplate allows program to format -- cannot do anything in template
    - Other systems allow for format control
    - Most systems do html escaping or other format processing of some type
- Test cases
    - print an invoice -- sufficiently complex, easy to understand
    - Hello world...
- Keep it simple to get something done
    - Avoid natural tendency to try and please all
    - Trend in other engines is to simplify template language
        - if it 'looks like programming' its too complex

Some possible approaches
- Put better interface over ctemplate (or plustache)
- New from ground up