Abstract

The paper considers a problem of reuse the legacy code in the Web-oriented software, namely ISAPI extension modules. Since ISAPI classes MFC Application Architecture, reuse of legacy classes that are compliant with the MFC document-view paradigm may be a challenge. The extension library presented in the paper empowers Web programmers to speed up creation of the ISAPI modules by reuse of the document classes from the legacy MFC applications.

Keywords: World Wide Web, ISAPI, legacy applications.

1. Introduction

Since early 90s the Bioinformatics Department of IMG RAS has been developing the electronic encyclopedia "Human Biology. As a part of this project there had been developed a line of the viewer applications for international databases on biology and biochemistry. These applications were developed using the MFC (Microsoft Foundation Classes) framework.

When the decision was made to transfer the encyclopedia to the World Wide Web, the developers faced a problem of converting data from the international databases to the formats recognizable by the Web browser. The data was decided to be converted on-the-fly by the Web server extension modules.

For development of the extension modules the MFC library already familiar to the developers was chosen. The MFC library contains classes for creation of the extension modules that follow ISAPI standard. The use of MFC has opened opportunities of a reuse of the document classes from the legacy viewer applications. But, unfortunately, the ISAPI extension classes do not provide the possibility of interaction with the principal part of the MFC library responsible for document management. So the problem has arisen to interweave the design patterns implemented in the MFC Application Architecture framework and in the MFC ISAPI framework.

2. Architectural Solution

While studying the design solutions implemented in the MFC Application Architecture framework one can discover the important assumption that MFC designers made: generally speaking, every document object should be created by the document template object via the document runtime class object. This scenario follows Object Factory pattern from the renown design patterns collection [5]. Thus, the core class of our converter library should be a document template class that creates the document and connects it to the converter class instead of the view class. Architecture of the library classes is presented in figure 1 in appendix A; it is as much close to MFC Application Architecture as possible. The library provides the two-level framework where the classes of application layer are reused on the server module layer. The main difference is in the style of object interaction and object lifetimes. Figure 2 helps to catch the process of the object interaction at the runtime. The whole library design follows the idea of multilevel framework for Internet applications that was presented in [6].

The library was packaged as the MFC extension DLL and used in a number of data conversion modules. Sample outputs are presented in appendix B.

3. Conclusion

The extension library described provides an interesting example of the new framework creation on the basis of two different ones. The process of studying the MFC architecture illustrates the main problem that the extension library architect faces. Understanding MFC architecture requires browsing the tons of code and rediscovering the design patterns used by library creators. Probably the learning curve may be
shortened by more intensive use of patterns and/or UML diagrams while documenting class library design.

4. References

1. Blaszczak M., Writing Interactive Web Apps is a Piece of Cake with the New ISAPI Classes in MFC 4.1, Microsoft Systems Journal, Vol. 11, No. 5.


5. Appendix A. UML Diagrams

![Class Diagram]

**Figure 1. Class Diagram**

![Interaction Diagram]

**Figure 2. Interaction Diagram**
6. Appendix B. Converters at Work

Figure 3. MEDLINE to HTML converter

Figure 4. CHEMBASE SDF to GIF converter