

**Design and Analysis of the Gene Designer Parameter Persistence
Implementation:**

Project Summary and Lessons Learned

by

Scott B. Bradley

for

CSC541, Introduction to Proteomics, DePaul University

Primary Stakeholders

David Angulo, DePaul University, Illinois Bio-Grid Project
Rob Vogelbacher, DePaul University, Illinois Bio-Grid Project
Scott Bradley, DePaul University

Executive Summary

Gene Designer is a component within the Illinois Bio-Grid Protein Desktop application. Gene Designer is a resource available to research scientists for producing DNA sequences which code for given amino acid sequences. This resource is useful for research related to protein synthesis.

The objectives of this project were primarily to:

1. Provide a persistence mechanism within the Gene Designer application which would allow the user to save the input parameters of a particular run.
2. Provide a mechanism by which saved parameters could be shared among users in order to facilitate the sharing of research information related to the Protein Desktop.

These objectives were realized by means of an XML based storage mechanism which allows the user to save the current input parameters as default, or to export them to an XML file and reload them at a later time.

This document summarizes some key points and lessons learned involved in implementing these objectives in the Gene Designer Desktop.

XML Advantages

XML was chosen as the persistence mechanism primarily because:

1. It provides a simple means of parameter persistence in plain text files
2. It facilitates data exchange by providing a consistently structured data format.

The decision to use XML as a parameter persistence mechanism thus not only allows for simplified saving of parameters, it also simplifies the exchange of parameter data between Gene Designer instances, as well as the export and retrieval of specific parameter sets for a given application instance. Because XML has been widely adopted as a common means of data exchange, this decision also increases the probability that Gene Designer might be made compatible with other applications using similar data sets through the use of simple transforms. Furthermore, since Java has built-in mechanisms for XML-based property persistence, the job of making Gene Designer inputs to be persistable was simplified by this decision.

The Java Properties Class

Java provides two different mechanisms for implementing key/value persistence for properties or preferences. The first is the `java.util.Properties` class, and the second is the functionality provided by

the `java.util.prefs` package. The latter is a newer implementation generally considered to be more flexible and more robust, however, as explained below, the decision was made to use the older, simpler `Properties` class.

Deciding against `java.util.prefs`

Java's `prefs` package has been implemented as a replacement for the simple `java.util.Properties` class with the primary objectives of:

1. Providing a means for hierarchical organization of stored key/value pairs.
2. Abstracting the storage mechanism such that the programmer using the package need not be concerned with the storage details.

These two primary objectives of the `prefs` package were also the source of the biggest problems in using the package, particularly the second. Regarding the first objective, a means of hierarchical organization of the Gene Designer parameters would add value to the storage feature, however the implementation provided by the `prefs` package is too limited to be useful. Organization of key/value preferences using this mechanism is strictly by package. Since all of the parameters to be saved are in the same package, this feature of the `prefs` package was not useful for our current implementation.

Our biggest problems resulted from the second objective of the `prefs` package mentioned above – and it is here that the objective for “improving” key/value storage capabilities in Java actually proved to be self-defeating. The trouble with completely abstracting the storage mechanism away from the programmer is that it creates new potential for unforeseen problems. The documentation on the `prefs` package actually discourages the programmer from learning the details of the storage mechanism, thus we were fortunate to catch the problem early in the project. Special thanks to Mike Boyle for his unexpected contribution on this.

The problem was simply that on the Windows operating system, the JVM uses the Windows Registry as its storage mechanism. While an undesirable means of parameter storage (and further, while one which seems to defeat the “write once run anywhere” mantra of the Java community) this mechanism may have been tolerable except that: 1. Not all users have the necessary permissions to modify the registry, and 2. There is no means for overriding this behavior and requiring that a file be used for storage instead. This is a particular problem in a campus environment where publicly shared computers must be set up not to allow registry access (as was the case here).

There was a third problem that contributed to the decision not to use the `prefs` package. That problem was that while the package allows for export of preferences, it only allows *saved* preferences to be exported. This necessitated a complicated workaround whereby default preferences had to be exported to a temporary file in order to save and export the current set of preferences and then reload the defaults. This difficulty contradicted one of the primary objectives of the project – to provide a simple mechanism for Gene Designer parameter exchange.

It should be clear at this point, there are times when simpler is better, and this was one of them. The decision was made to use the `java.util.Properties` class which provides a simple mechanism for key value storage in `.xml` files.

Using java.util.Properties

Using the Properties class, a key/value property is saved and retrieved respectively by the methods:

setProperty(key, value)

getProperty(key)

Table 1 lists the implemented property keys and their respective Gene Designer parameters.

<i>Parameter key</i>	<i>Description</i>
aminoAcidSequence	Input amino acid sequence
selectedGeneType	Gene type: currently supports one of E Coli All Genes, E Coli CClass II, AcPNV All Genes
GFactor	Guanine factor
Afactor	Adenine factor
Tfactor	Thymine factor
Cfactor	Cytosine factor
includedEnzymes	Included restriction enzymes in the format name:target-sequence delimited by comma (,)
excludedEnzymes	Excluded restriction enzymes in the format name:target-sequence delimited by comma (,)
5primeFlankingSequence	Flanking sequence on 5' end
3primeFlankingSequence	Flanking sequence on 3' end
populationSize	Size of population for the genetic algorithm run
generations	Number of generations for the genetic algorithm
mutationRate	Mutation rate for the genetic algorithm
nucleotideRatios	Requested weight for nucleotide ratios
codonPreference	Requested weight for codon preference
restrictionSiteExclusion	Requested weight for exclusion sites
restrictionSiteInclusion	Requested weight for inclusion sites
preferredOligoSize	Preferred size of resulting oligonucleotides
maxMeltingTemperature	Maximum melting temperature of resulting oligonucleotides
minMeltingTemperature	Minimum melting temperature of resulting oligonucleotides
saltConcentration	Salt concentration (in mM) of resulting oligonucleotides
oligoConcentration	Oligo concentration (in μ M)

Table 1. Parameter Keys

Once saved to an instance of the Properties class, a set of properties can then be written to a standard XML format with the method storeToXML which takes an OutputStream and a String comment which is included in the file. The comment was used as a means for identifying the Gene Designer application and the date/time of the parameter export.

Using Properties rather than Preferences allowed for a dramatic decrease in the amount of code required for the problem at hand. In particular, several points of exception handling were eliminated due to the fact that the complex workaround of temporary storage of default parameters was no longer needed.

The Properties approach has the added advantage of providing a consistent mechanism for both storage and export. Thus, the technically savvy user need not even export his parameters to have them available for exchange, review or manipulation since they always reside on the file-system in the same directory as the IBG Desktop as the GeneDesignerDefaults.xml file. This file is in the same format as a would-be parameters export file. A sample parameters file listing from real Gene Designer data is shown in the Listing 1 below.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
  <comment>Properties file created by the IBG Gene Designer on May 30, 2006 7:22:16 PM
  CDT</comment>
  <entry key="generations">1000</entry>
  <entry key="excludedEnzymes">BamHI:GGATCC,EcoRI:GAATTC,XbaI:TCTAGA,XhoI:CTCGAG</entry>
  <entry key="TFactor">1.0</entry>
  <entry
key="5primeFlankingSequence">CATATGAAACATCATCATCATCACAGCAGCGACTACAAAGACGACGATGACAAAGGTGA
AAACCTGTACTTCCAGGGATCC
  </entry>
  <entry key="restrictionSiteInclusion">0.2</entry>
  <entry key="saltConcentration">50.0</entry>
  <entry key="AFactor">1.0</entry>
  <entry key="minMeltingTemperature">54.0</entry>
  <entry key="GFactor">1.0</entry>
  <entry key="mutationRate">0.01</entry>
  <entry key="maxMeltingTemperature">56.0</entry>
  <entry key="selectedGeneType">1</entry>
  <entry
key="aminoAcidSequence">GGGAEAGITGTWYNQLGSTFIVTAGADGALTGTYESAVGNAESRYVLTGRYDSAPATDGS GTALGWT
VAWKNNYRNAHSATTWSGQYVGGAERINTQWLLTSGTTEANAWKSTLVGHDTFTKVKPSAAS
  </entry>
  <entry
key="3primeFlankingSequence">TAATGACTCGAGCACCACCACCACCACCAGATCCGGCTGCTAACAAAGCCCCGAAAGGA
AGCTGAGTTGGCTGC</entry>
  <entry key="CFactor">1.0</entry>
  <entry key="preferredOligoSize">60</entry>
  <entry key="populationSize">500</entry>
  <entry key="includedEnzymes"></entry>
  <entry key="nucleotideRatios">0.0</entry>
  <entry key="restrictionSiteExclusion">0.35</entry>
  <entry key="oligoConcentration">0.5</entry>
  <entry key="codonPreference">0.35</entry>
</properties>
```

Listing 1. Gene Designer Parameters .xml File

The primary drawback of the Properties class for the purposes of the current project is that it allows only for simple key-value storage. No complexities of ordering or organizing of the key/value pairs has been implemented in this class. This aspect of the Properties class has necessitated the implementation of restriction lists as single elements with delimiters rather than as individual XML elements. This can be seen in Listing 1 by inspecting the entry with the key `excludedEnzymes` which contains the value list: `BamHI:GGATCC,EcoRI:GAATTC,XbaI:TCTAGA,XhoI:CTCGAG`. In this list format, individual enzymes are delimited by commas (,) while the entry for each enzyme contains a name and a sequence delimited by a colon (:). Although this format is not ideal within the traditionally hierarchical context of XML, the consistency of format still allows for simple data exchange. The code in Gene Designer has been implemented to expect this delimited format. For applications outside of Gene Designer which may need to utilize this data, the XSLT transform can be written to parse the elements of a list of restriction enzymes. For such tasks, the XSLT commands `substring-before` and `substring-after` are useful. An example of element tokenization around specified delimiters can be found in the well known *XSLT Cookbook* (Sal Mangano, O'Reilly Press, 2003) on pages 19-21.

Interface Overview

This section summarizes the new features implemented in the Gene Designer interface for parameter persistence. Persistence of the Gene Designer parameters is activated through the IBG Desktop File menu. As illustrated by Figures 1 & 2, this feature has been made context sensitive in order to restrict its functionality to the Gene Designer sub-application.

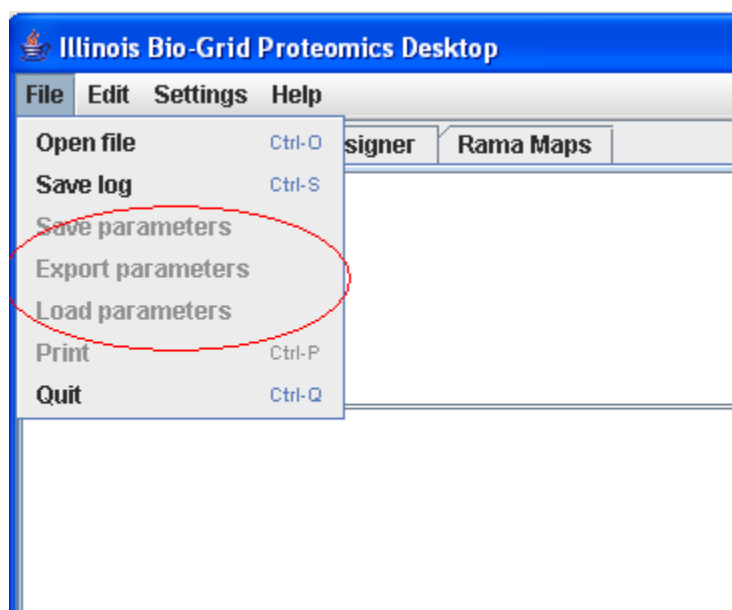


Figure 1. Menu Items Inactive Outside the Gene Designer Context

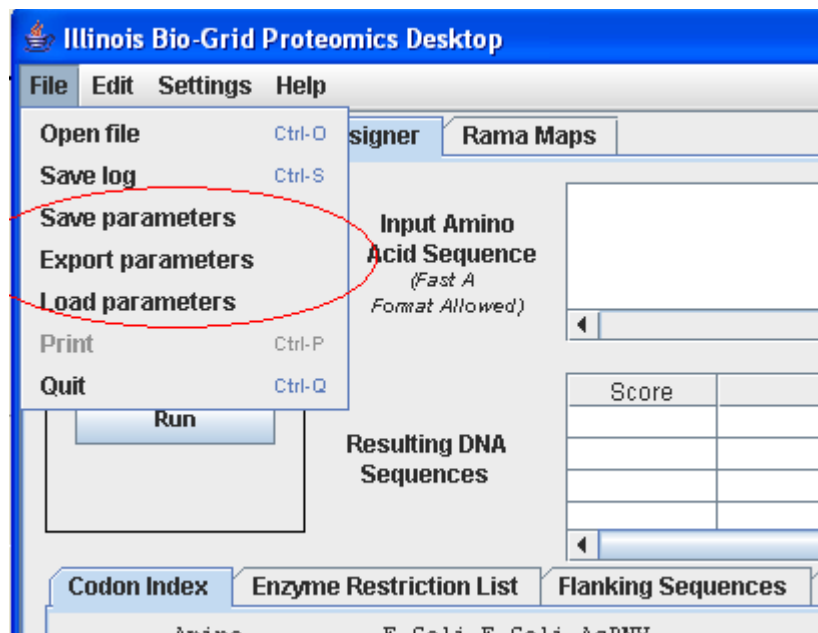


Figure 2. Menu Items Activated in Gene Designer Context

From within the Gene Designer context of the IBG Desktop application, the user can choose to Save parameters, Export parameters, or Load parameters. As shown in Figure 3, Loading gracefully handles cases of an incorrectly formatted or corrupt parameters file. Each of the File menu items results in the following actions respectively:

1. Save parameters: Saves the current Gene Designer input parameters to the default parameters file GeneDesignerDefaults.xml
2. Export parameters: Allows the user to select a destination to write the current Gene Designer parameters with a provided file name. As shown in Figure 4, .xml file filtering is included, with the option of viewing all file types during file destination and name setting. Furthermore, Gene Designer will automatically save the file with the .xml file extension even if the extension is not provided by the user.
3. Load parameters: Allows the user to select a parameters file to load into the Gene Designer interface. XML file type filter is again implemented.

The resulting file from the save and export actions contains key/value pairs as discussed above in the section "Using java.util.Properties". See Table 1 in that section for a list of the implemented parameter keys.



Figure 3. Parameters Not Loaded

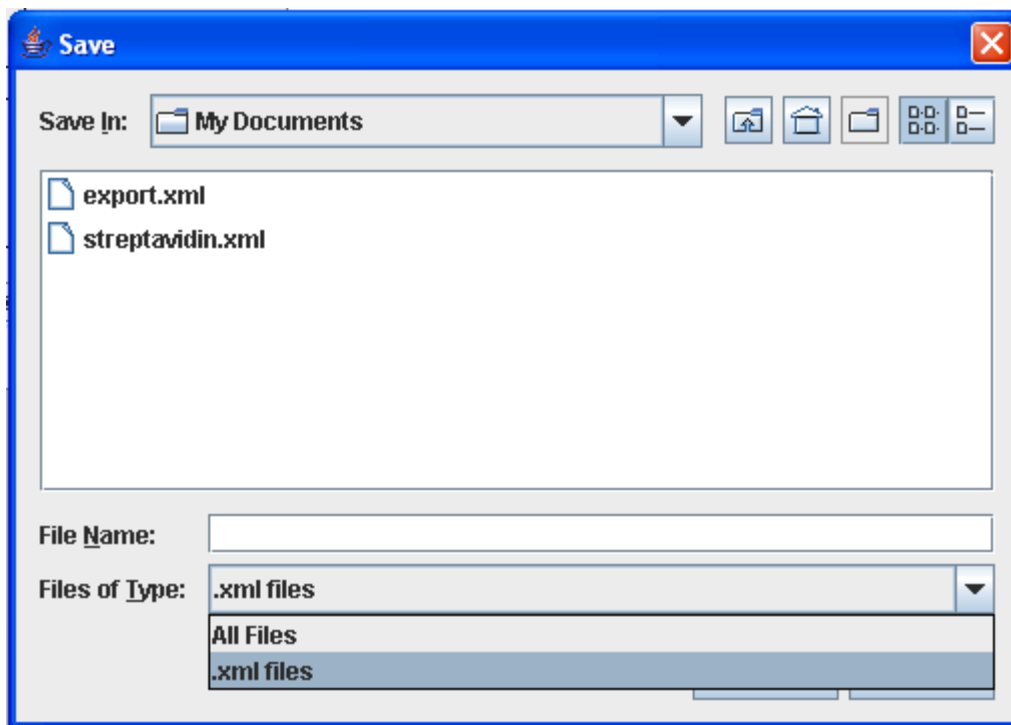


Figure 4. Save Dialog for Parameter Export Showing .xml File-type Filter

Future Consideration

As stated above, the decision to use XML as the means of parameter persistence makes the potential for data exchange simply a matter of implementing the needed transforms for converting data to or from the Gene Designer format. The simple XML format provided by the properties class makes transformation a simple matter of matching keys and extracting the values. The primary exception to this is in the case of restriction lists which have been implemented as single element instances of lists of name/sequence pairs. Transformation of these elements would require some form of tokenizing around the ':' and ',' delimiters of these lists.

The use of the Properties class also simplifies possible future implementation of new program input parameters which can simply be included as new setProperty and getProperty statements in the Gene Designer code without concern for the XML format apart from assigning a valid unique key for the new parameter.

If there are any questions regarding these implementations, or any need for further work such as the development of XSLT transforms or expanding the saved parameters set, please feel free to contact me at s.bradley@computer.org.