Technical report, June 2011

Retrofit Learning Management System To Use SCORM

Bachelor's Thesis in Computer Engineering

Mingkun Yang

School of Information Science, Computer and Electrical Engineering, Halmstad University

Details

First Name, Family Name:	Mingkun Yang
University:	Halmstad University, Sweden
Degree Program:	Computer Engineering
Title of Thesis:	Retrofit Learning Management System To Use SCORM
Academic Supervisor:	Wagner Ourique de Morais

Abstract

The existing Learning Management System (LMS), “esTracer”, from Entergate company has been updated to be compliant with Sharable Content Object Reference Model (SCORM). By following the SCORM standard, new and existing packages of course materials could be imported and exported to SCORM compliant systems. In the test, one package from ADL is used for importing and presentation. The implementation only applies to “esTracer” LMS, but the methodology is general and could be used in other LMS as well. This project is implemented using ASP.NET MVC framework. ASP.NET MVC framework is one great framework, which enables developers to stand on a high level to see the whole picture. Future work needs to be done to make “esTracer” complete SCORM compliant.

Acknowledgements

First of all I want to thank my supervisor Wagner Ourique de Morais for his suggestions, advices and help in the whole process of doing this project.

Secondly, I want to thank Cyrus Daneshmir for providing such great oppotunity to conduct this project.

Lastly, I want to thank Johan Simonsson for all the elaborate explanation he did whenever I was in doubt.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, introduction to this project is covered, including background, objectives, limitations, etc.

1.1 Background

Essentially, E-learning is the transferring of skills and knowledge through computer and networks. With the rapid development of Internet, E-learning is becoming more and more popular and practical. E-learning applications include Web-based learning, computer-based learning, virtual classroom opportunities and digital collaboration. This project is proposed by Entergate company (www.entergate.se), and used to extend the original functionality of their product “esTracer”, which is the Learning Management System from the company.

1.2 Related work

When it comes to E-learning, Advanced Distributed Learning has to be mentioned. The Advanced Distributed Learning (ADL) Initiative is the result of collaborative efforts to harness the power of information technologies to deliver high-quality, easily accessible, adaptable, and cost-effective education and training [1]. More information about ADL will be covered when the origin of Sharable Content Object Reference Model (SCORM), which is one standard related to E-learning and will be explained in detail in next chapter, is addressed.

The most prestigious company providing SCORM related solution is Rustici Software [2]. They make SCORM easy by creating the best SCORM products on the market. In fact, ADL is sponsoring them to conduct the Tin Can research, which is meant to be the successor of SCORM. Further information can be found in section 2.5.

1.3 Objectives

Without following the standards, endless of time and money will be wasted for “reinventing the wheel”. Considering SCORM is the de facto standard in E-learning system, the objective of this project is to make “esTracer” SCORM compliant. This project focuses on importing and exporting SCORM compliant packages, and present the content of packages without providing communication with the Learning Management System(LMS). This project is one preliminary work of making “esTracer” SCORM compliant.

The whole mission is broken into four tasks:

- Importing whole package
  This is used when the whole packaged is meant to be launched by LMS directly.
- Importing all SCOs in this package
  This will enable course content authors to create complete tests out of all the SCOs, which could then be exported as SCORM compliant packages or launched by LMS.
- Exporting package
  This could be used when different LMSs want to share packages.
- Launching SCO
  Via this, users can interact with the SCO. (In this project, the communication between SCO and LMS is not implemented, so no records will be save after finishing this SCO.)

1.4 Limitation

Considering E-learning is one quite broad concept, this project will only concern about Web-based learning E-learning system, specifically the product from Entergate company. The implementation only applies to “esTracer”, but the methodology is general, which can be used in other systems.

Make one Learning Management System SCORM compliant requires launching the content in one frameset or one new windows, then to provide the APIs to enable the communication between the Learning Management System and the content. This project only covers the importing, exporting and previewing the content of SCO without providing the communication APIs.

1.5 Outline

In Web-based E-learning System and SCORM chapter, E-learning and SCORM is addressed to give the audiences the fundamental concepts concerning E-learning and the de facto standard. The standard is explained thoroughly because this project depends on it heavily.

In the Method chapter, the working environment and methodology adopted is presented in detail. The “esTracer” is explained from a technical perspective so that modification could be made in order to extend its functionality. Afterwards, how each task is solved, and why one particular approach is chosen are explained.

In the Result chapter, one package from ADL is used to demonstrate the use of functionality concerning SCORM in “esTracer”, such as importing and presentation.

In the Discussion and Conclusion chapter, personal opinion about various technology used in this project is expressed. Since this project is just one preliminary task of making “esTracer” SCORM compliant, future work would be done, some of which are discussed.

Chapter 2

Web-based E-learning System and SCORM

In this chapter, some technology related to E-learning system is explained, which paved the way for utilizing them in this project.

2.1 Learning Management Systems

A Learning Management System (LMS) is a software package that is typically accessible via the Internet and is used to manage one or more courses to one or more learners. In other words, an LMS allows learners to authenticate themselves, register for courses, access learning content, take assignments, and then store the learners grade records [3]. Apparently, this model is based on physical administration process.

2.2 Learning Content Management Systems

An Learning Content Management System (LCMS) is a development of the LMS. It is a multi-tier environment where learning materials developers may create, store, reuse, manage, and deliver digital learning content from a central content repository [4]. To some extent, LCMS can be treated as one combination of the original LMS and Authoring Tools, which will be explained immediately.

2.3 Authoring Tools

Authoring tools are software applications used to develop E-learning products. They generally include the capabilities to create, edit, review, test, and configure E-learning packages. These tools support learning, education, and training by using distributed E-learning system that is cost-efficient to produce, and that facilitates incorporating effective learning strategies and delivery technologies into the E-learning system. Authoring tools themselves are not SCORM-certified, but many are designed to create SCORM compliant E-learning course materials [5].

Since it is very convenient to use LCMS, LMS and authoring tools do not appear as standalone applications any more. In this project, the term “LMS” is used, even though, in fact, “LCMS” is concerned.

2.4 History of SCORM

In the late 1990s, as computer was getting more and more popular, everyone was using computer in their daily life and work. The United States Department of Defense realized that they were procuring the same training content many times over and over again, but could not reuse it across departments, because each department had its own Learning Management System. In those days, each LMS had its own proprietary content format, which discouraged content sharing. In 1999 an executive ordered a small research laboratory, Advanced Distributed Learning(ADL), to “develop common specifications and standards for E-learning.” Rather than starting from scratch, ADL harmonized the work of existing standards organizations like the Aviation Industry Computer-Based Training Committee (AICC) [6], IMS [7] and the IEEE Learning Technology Standards Committee (LTSC) [8] into a cohesive reference model, which becomes SCORM later. SCORM was released in 2001 and was quickly adopted by both government and industry. Today it is the de facto standard for E-learning interoperability [9].

2.5 SCORM Variation

With the technology of Internet and computer gets more and more mature, people has improved SCORM so that it could adapt to various situations. At present, there are three major players on the stage.

- SCORM 2004
  This is the current version. It is based on new standards for API and content object-to-runtime environment communication, with many ambiguities of previous versions resolved. It contains ability to specify adaptive sequencing of activities that use the content objects, and the ability to share and use information about success status for multiple learning objectives or competencies across content objects and across courses for the same learner within the same learning management system [10].

- SCORM Cloud
  Using SCORM Cloud, learners can be reached where they are rather than forcing them to go to the LMS. In addition to supplying best of class SCORM, SCORM Cloud serves as a connection point between one LMS and the rest of the Internet [11].

- Tin Can
  Over the last decade, two phenomena have happened. First, as the adoption of SCORM soared, people wanted to utilize it in new ways, ways that stretched the limits of what SCORM was designed to do. Second, technology has evolved at a rapid pace. In comparison with present technologies, the current SCORM structure is antiquated and inadequate. Both of these trends demand the need for a more modern and powerful communication framework [12].

The biggest advantage of SCORM Cloud over SCORM 2004 is that it does not require users to go to the LMSs. Instead, users can be connected and tracked anywhere in the web. In this case, users are asked to come to the LMS, and take the test in this LMS, so SCORM 2004 suffices. The weakest point of Tin Can is that this project hasn't finished yet. According their announcement [12], this project consists of three phases, and they are still in the first phase when this project is conducted. Therefore, the most appropriate solution for this case is SCORM 2004. In the following, SCORM 2004 is introduced.

2.6 SCORM 2004

SCORM specifiers that content should:

- Be packaged in a ZIP file.
- Have one XML to describe the package
- Communicate with the LMS via JavaScript.
- Define sequence using rules in XML.

SCORM consists of three sub-specifications:

- The Content Packaging section specifies how content should be packaged and described. It is based primarily on XML. The content should be packaged in a ZIP file.
- The Run-Time section specifies how content should be launched and how it communicates with the LMS. It is based primarily on ECMAScript (JavaScript). The LMS should launch content in a web browser, either in a new window or in a frameset.
- The Sequencing section specifies how the learner can navigate between parts of the course. It is defined by a set of rules and attributes written in XML.

Considering the particular tasks in this project, content packaging is focused.

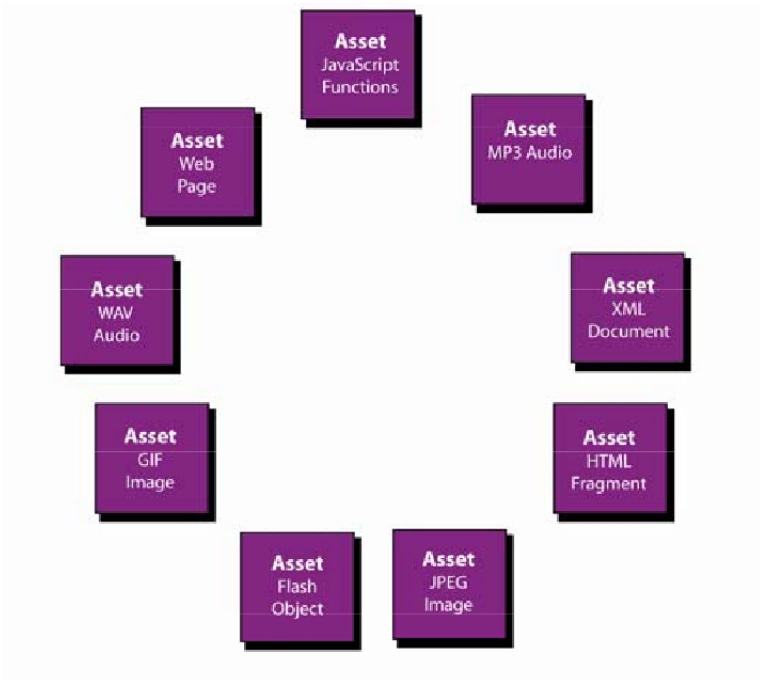


Figure 2.1: Examples of asset [13].

2.7 The SCORM Content Aggregation Model

The SCORM Content Aggregation Model (CAM) represents a learning-taxonomy neutral way for designers and implementers of instruction to aggregate learning resources for the purpose of delivering a desired learning experience [13].

The SCORM CAM consists of the following:

- Content Model: Define the content components of a learning experience.
- Content Packaging: How to structure and exchange learning content.
- Metadata: A mechanism for describing specific instances of the components of the content model.
- Sequencing and Navigation: A rule-based model for defining a set of rules that describes the intended sequence and ordering of activities.

In this project, only Content Model and Content Packaging are used. In the following, they are explained thoroughly.

2.7.1 Content Model

The SCORM Content Model explains the SCORM components used to construct a learning experience from learning resources. The content model also defines how these

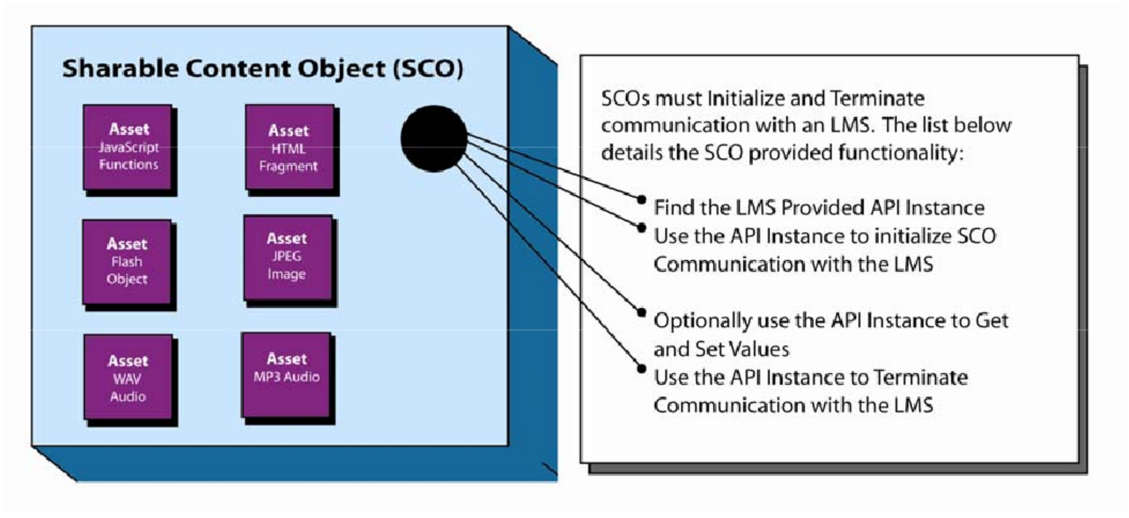

Sharable Content Object (SCO)
Asset JavaScript Functions
Asset HTML Fragment
Asset Flash Object
Asset JPEG Image
Asset WAV Audio
Asset MP3 Audio
SCOs must Initialize and Terminate communication with an LMS. The list below details the SCO provided functionality:
Find the LMS Provided API Instance
Use the API Instance to initialize SCO Communication with the LMS
Optionally use the API Instance to Get and Set Values
Use the API Instance to Terminate Communication with the LMS

Figure 2.2: Conceptual markup of a SCO [13].

low level sharable, learning resources are aggregated and organized into higher-level units of instruction. The SCORM Content Model is made up of assets, sharable content objects (SCOs), activities, a content organization and content aggregations [13].

- Asset
  The asset is the basic building block of a learning resource. Assets are an electronic representation of media, such as text, images, sound, assessment objects or any other piece of data that can be rendered by a Web client and presented to a learner.

  Typical examples of asset are shown in Figure 2.1.

- SCO
  A SCO is a collection of one or more assets that represent a single launchable learning resource that uses the SCORM RTE to communicate with an LMS. Figure 2.2 illustrate the markup of SCO. The only difference between a SCO and an asset is that the SCO communicates with LMS using javascript.

- Activities
  A learning activity may be loosely described as a meaningful unit of instruction; it is conceptually something the learner does while proceeding through instruction.

  The activities represented in a content organization may consist of other activities (sub-activities), which may themselves consist of other activities. There is no limit to the number of levels of nesting for activities. Activities that do not consist of other activities (leaf activities) will have an associated learning resource (SCO or asset) that is used to perform the activity.

- Content Organization

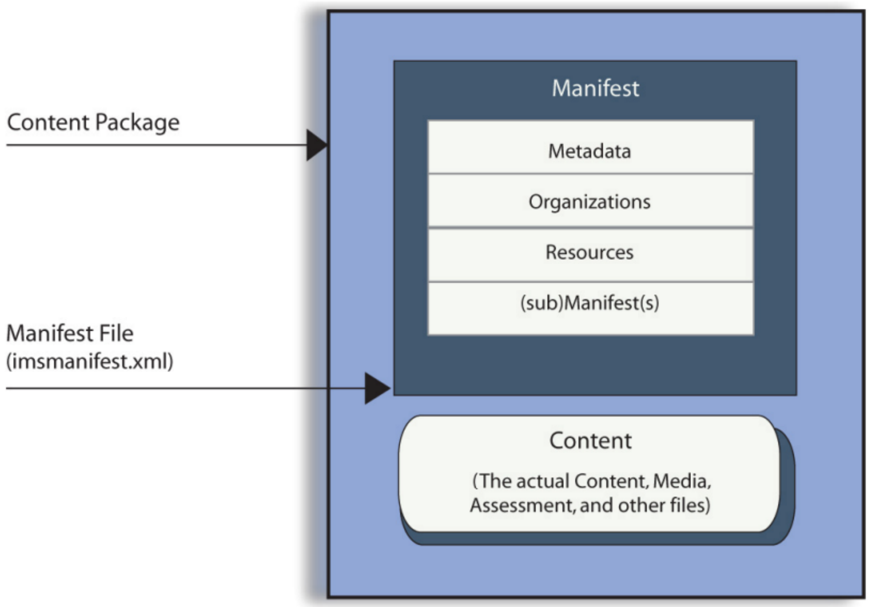


Figure 2.3: Components of a content package [13].

A content organization is a representation or map that defines the intended use of the content through structured units of instruction (activities). The map shows how activities relate to one another. In other words, it represents the sequencing information.

The LMS is responsible for interpreting the sequencing information described in the content organization and applying sequencing behaviors to control the actual sequence of the learning resources at run-time.

- Content Aggregation
  Content aggregation can be used as both an action and as a way of describing a conceptual entity. Sometimes, this term is loosely used to describe content package.

2.7.2 Content Packaging

The purpose of the content package is to provide a standardized way to exchange learning content between different systems or tools.

Content Package Component

A content package contains two major components [13], as presented in Figure 2.3 and described in below:

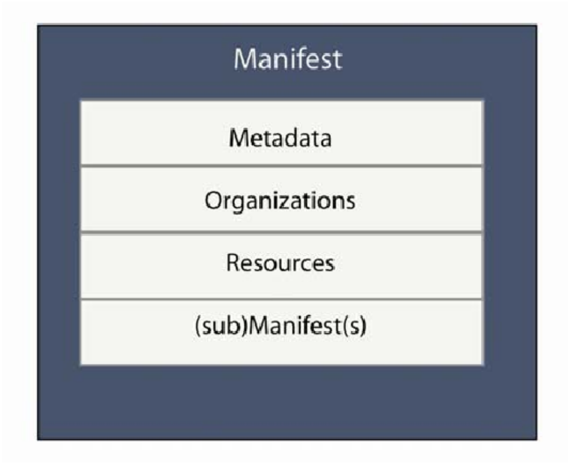


Figure 2.4: Components of a Manifest [13].

- A special XML document describing the content structure and associated resources of the package called the manifest file (imsmanifest.xml). A manifest is required to be present at the root of the content package.

- The content (i.e., physical files) making up the content package.

Components of a Manifest

The general rule is that one single manifest file exists in the root directory, describing this package. At present, ADL recommends not to use (sub)manifests until the completion of the corresponding work. The manifest consists of four major components [13], as presented in Figure 2.4 and explained in below:

- Metadata: Data describing the content package as a whole, which enables the search and discoverability of the package itself.

- Organizations: Contains the content structure or organization of the learning resources making up a stand-alone unit or units of instruction. A definition of sequencing intent can be associated with the content structure.

- Resources: Defines the learning resources bundled in the content package.

- (sub)Manifest(s): Describes any logically nested units of instruction (which can be treated as stand-alone units).

This section will be reviewed in detail with one example in next chapter.

SCORM Content Package Application Profiles

SCORM Content Package Application Profiles describe how the IMS Content Packaging Specification will be applied within the overall context of SCORM. There are currently two SCORM Content Package Application Profiles, which describe how to package CAM components: [13]

- Resource Content Packages
  The SCORM Resource Content Package Application Profile defines a mechanism for packaging assets and SCOs without having to provide any organization, learning context or curricular taxonomy.
- Content Aggregation Content Packages
  Its main purpose is to be used to deliver content to an end user (i.e., typically through an LMS).

Chapter 3

Method

In this chapter, various methods involved in this project are explained. They are arranged in the way that is corresponding to the four tasks.

3.1 Environment

3.1.1 Visual Studio

While working with Visual Studio, developers should keep one thing in mind, that there is only one solution at one time, and this solution will always have one or more projects [14]. This might feel a little bit strange for new users, but it is quite reasonable if one thinks about this organization especially when one task is too complicated and immense to fit in as one project.

This project is based on the ASP.NET MVC Web Application technology. While creating one such application with Visual Studio, it automatically adds a number of files and directories to the project, as is shown in Figure 3.1. Although this project is based on one existing project (no need to create new project), understanding the file organization is critical, for the existing project keeps the original layout somehow.

One new ASP.NET MVC project by default has six top-level directories: [15]

/Controllers	Where we put Controller classes that handle URL requests
/Models	Where we put classes that represent and manipulate data
/Views	Where we put UI template files that are responsible for rendering output
/Scripts	Where we put JavaScript library files and scripts (.js)
/Content	Where we put CSS and image files, and other non-dynamic/non-JavaScript content
/App_Data	Where we store data files we want to read/write.

Table 3.1: File organization.

ASP.NET MVC does not require this structure. However, developers working on

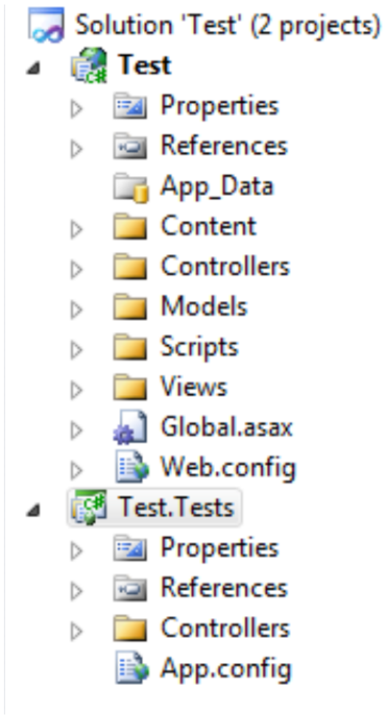


Figure 3.1: MVC layout.

large applications will typically partition the application up across multiple projects to make it more manageable (for example: data model classes often go in a separate class library project from the web application which is exactly the case in this project). The default project structure, however, does provide a nice default directory convention that developers can use to keep the application concerns clean.

The directory layout used by in this project is based on the default, with more complex arrangements capable of dealing with different situations.

3.1.2 ASP.NET MVC

The Table 3.2 illustrates several typical URL routing examples. The natural starting

URL	Controller Class	Action method	Passed parameter
/	Home Controller	Index()	None
/Home	Home Controller	Index()	None
/Home/About	Home Controller	About()	None

Table 3.2: Examples of routing.

point to learn ASP.NET MVC is to understand the execution flow. The understanding

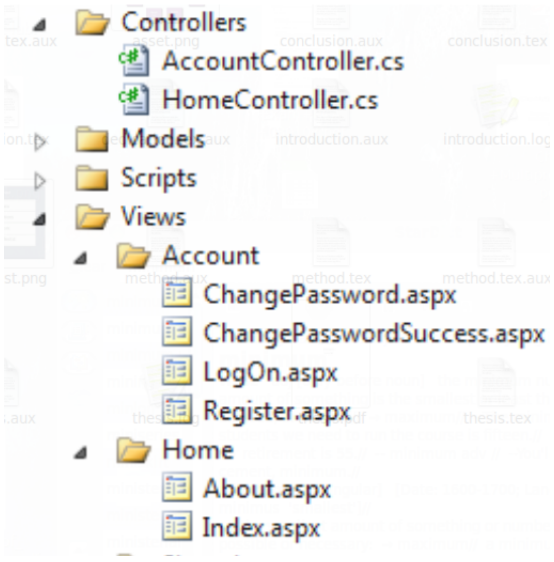


Figure 3.2: URL routing.

of execution flow begins with the URL routing.

With traditional web frameworks (classic ASP, PHP, ASP.NET Web Forms, etc), incoming URLs are typically mapped to files on disk. For example: a request for a URL like "/Books.aspx" or "/Books.php" might be processed by a "Books.aspx" or "Books.php" file.

ASP.NET MVC includes a powerful URL routing engine that provides a lot of flexibility in controlling how URLs are mapped to controller classes. It allows developers to completely customize how ASP.NET MVC chooses which controller class to create, which method to invoke on it, as well as configure different ways that variables can be automatically parsed from the URL/Query string and passed to the method as parameter arguments.

By default, new ASP.NET MVC projects come with a pre-configured set of URL routing rules already registered. This enables developers to easily get started on an application without having to explicitly configure anything. Routing configuration is loaded when the application starts. The default routing rule registrations can be found within the "Application" class of projects which can be opened by double-clicking the "Global.asax" file in the root of the project. This could be confirmed by setting one breakpoint on that line, and observe that this line is executed before any URL is interpreted.

This is one example of URL routing, based on the default routing rule, which is "/controller/action/id", where "controller" is the name of the controller class to instantiate, "action" is the name of a public method to invoke on it, and "id" is an optional parameter embedded within the URL that can be passed as an argument to the method. According the default rule, Controller = "Home", Action = "Index", and Id = "", if no argument is provided [16].

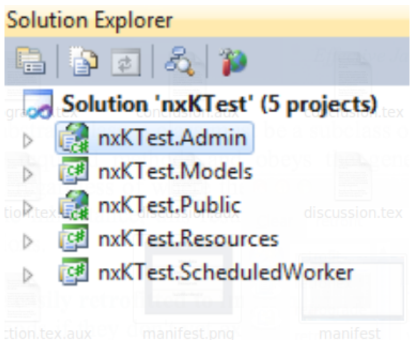


Figure 3.3: Application Layout.

3.1.3 nxtTest

This application is divided into five projects, as showed in the Figure 3.3. Only the first three will be used in this project. Since this project is the extension of one almost mature product, fully studying and understanding of it is critical before any non-trivial modification can be done. Consequently, the three projects are described briefly to illustrate how this project can be integrated into it.

The "Admin" project is responsible for the administration of LMS or tests. With proper right, users can create questions or import questions, create tests by adding existing questions, and grade tests. The administrators of LMS can assign the right to users. The importing and exporting function belongs to this project. The file organization is shown in Figure 3.5a.

The "Model" project is the heart of this application, as is the heart of MVC architecture. A plethora of files exist in this project to reflect the various types of questions and tests. The file organization is shown in Figure 3.5b. In this project, only a small subset is used.

The names of most tables are self-explaining, but some of them might be a little confusing. QuestionSCOSetting is used to store the settings particularly for SCO questions; for SCO questions, the QuestionSetting is not much used.

Tests are built out of questions, which will be saved in TestQuestion. The answers to tests are stored in TestAnswerSet. The relation between question and answer is stored in TestAnswerSetQuestion and TestAnswerSet, respectively. The reason for that is that sometimes the questions presented to users are randomly chosen from the question pool of the test.

QuestionRight is used to keep track of who has the right to use one particular question.

The "Public" project is where to see how the questions look like, which indicates that it is the interface to end users who are taking the tests, and question and test authors,

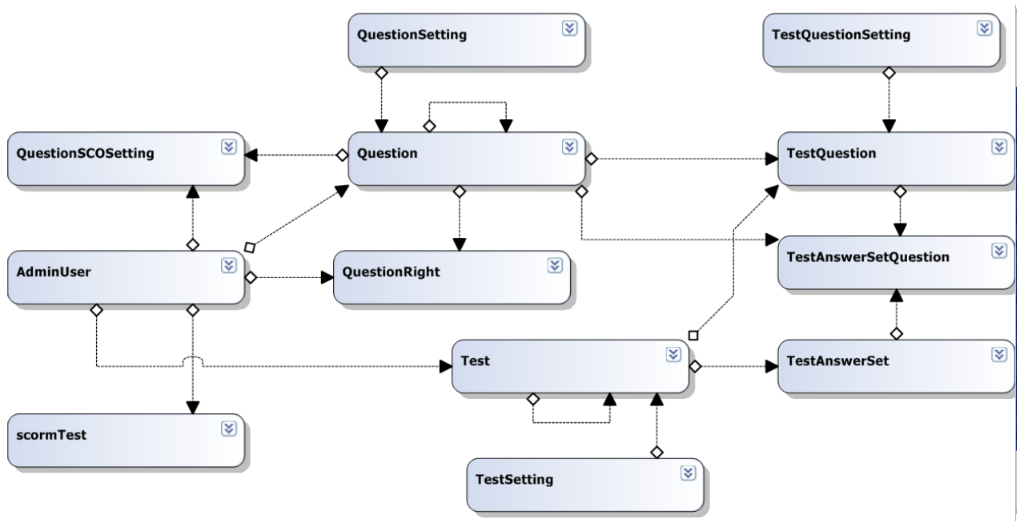


Figure 3.4: Table Relation Diagram.

who might want to have one preview of this question and test. The file organization is shown in Figure 3.5c. The Launching functionality belongs to this project.

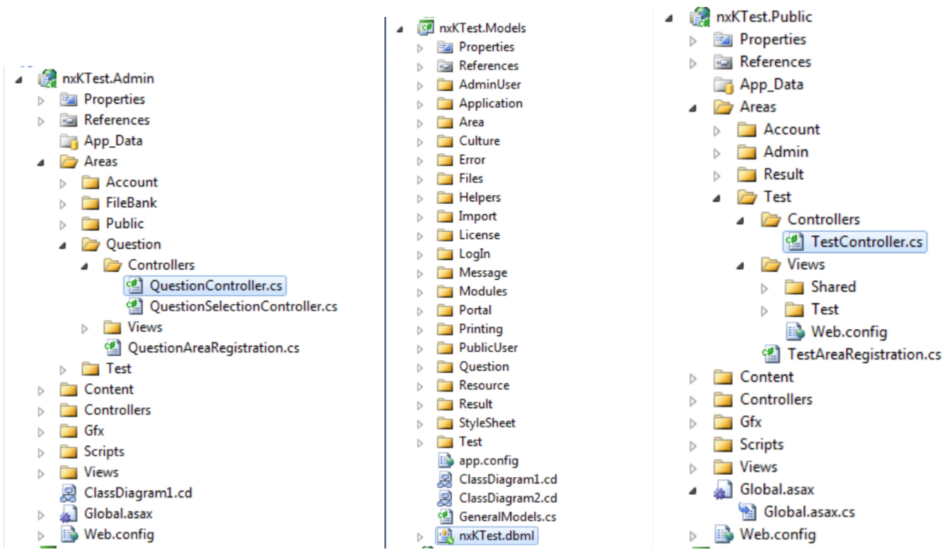
3.2 Import whole package

In the first case, import the package as one whole test, so that the LMS can launch it directly. Work needs to be done is to unzip this file, and save the content to one folder. The name of this folder is generated as one Globally Unique Identifier(GUID), which can prevent users guessing what other tests exist in this LMS. SharpZipLib [17] is used to uncompress the zip file.

After uncompressing, several pieces of information need to be written to database: folder name(GUID), identifier(the attribute of manifest in imsmanifest.xml) and the name of the user importing this package. The reason for saving the identifier of this package is to prevent importing the same package more than once.

According to SCORM 2004 standard [13], there is no guarantee that identifier is different from one to another. However, this attributes describe some information about this package, so it is reasonable to draw the conclusion that two packages with distinct content do not have the exact same identifier. In addition, this project is the preliminary work; improvement can be made afterwards.

In the second case, all the SCOs in this package will be extracted so that course/test content developers could build tests using different combinations. In addition to what have been done for the first case, extra work needs to be taken, such as locating different SCOs and their associated information, save them at different directories, etc.



(a) Admin Project (b) Models Project (c) Public Project

Figure 3.5: Projects.

3.3 Import single SCOs

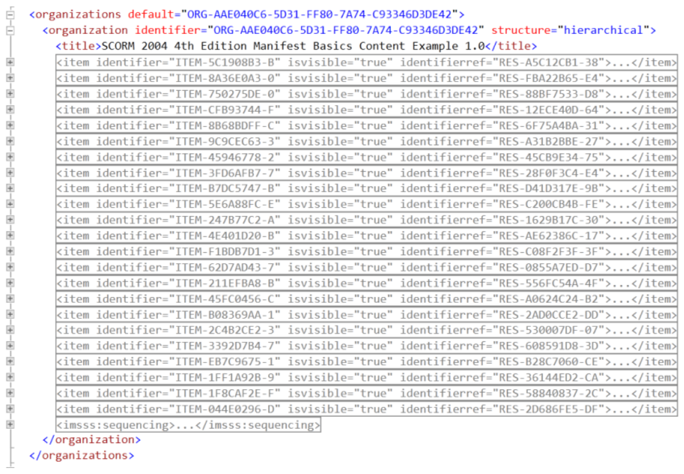
This package will be parsed in three phrases. Firstly, identify each SCO indicated in XML file. Secondly, identify all resources in this XML file. Finally, associate the resources with each SCO, copy the files to the appropriate locations, and save the information to the database.

In this project, Manifest Basics Content Example Content Package [18] is used. In the highest level, two elements are of interest: <organizations >and <resources >. Figure 3.6 and Figure 3.7 is the screenshot of organization and resource of the manifest file in this package, respectively. These two work in parallel, constituting the whole package. When this manifest file is parsed, information concerning organization and resource is saved and their corresponding relation.

According to the content model, the <item >element represents an activity in the content organization. Therefore, in the first step, the <item >element in this XML file is kept track of, and associated with “identifierref” attribute, title, metadata information. The “identifierref” attribute works like one pointer, pointing to the resources this activity relies on.

One class called “Item” is created to save the information related to one item, and one Class called “Items” is created to keep track of all the items in this XML file, which works like one container.

In the second step, try to locate the resources indicated by “identifierref” in the previous step. Because of the existence of <dependency >element, the exact information



```xml
<organizations default="ORG-AAE040C6-5D31-FF80-7A74-C93346D3DE42">
  <organization identifier="ORG-AAE040C6-5D31-FF80-7A74-C93346D3DE42" structure="hierarchical">
    <title>SCORM 2004 4th Edition Manifest Basics Content Example 1.0</title>
    <item identifier="ITEM-5C1908B3-B" isvisible="true" identifierref="RES-A5C12CB1-38">...</item>
    <item identifier="ITEM-8A36E0A3-0" isvisible="true" identifierref="RES-FBA22B65-E4">...</item>
    <item identifier="ITEM-750275DE-0" isvisible="true" identifierref="RES-88BF7533-D8">...</item>
    <item identifier="ITEM-CFB93744-F" isvisible="true" identifierref="RES-12ECE40D-64">...</item>
    <item identifier="ITEM-8B68BDFF-C" isvisible="true" identifierref="RES-6F75A4BA-31">...</item>
    <item identifier="ITEM-9C9CEC63-3" isvisible="true" identifierref="RES-A31B2BBE-27">...</item>
    <item identifier="ITEM-45946778-2" isvisible="true" identifierref="RES-45CB9E34-75">...</item>
    <item identifier="ITEM-3FD6AFB7-7" isvisible="true" identifierref="RES-28F0F3C4-E4">...</item>
    <item identifier="ITEM-B7DC5747-B" isvisible="true" identifierref="RES-D41D317E-9B">...</item>
    <item identifier="ITEM-5E6A88FC-E" isvisible="true" identifierref="RES-C200CB4B-FE">...</item>
    <item identifier="ITEM-247B77C2-A" isvisible="true" identifierref="RES-1629B17C-30">...</item>
    <item identifier="ITEM-4E401D20-B" isvisible="true" identifierref="RES-AE62386C-17">...</item>
    <item identifier="ITEM-F1BDB7D1-3" isvisible="true" identifierref="RES-C08F2F3F-3F">...</item>
    <item identifier="ITEM-62D7AD43-7" isvisible="true" identifierref="RES-0855A7ED-D7">...</item>
    <item identifier="ITEM-211EFBA8-B" isvisible="true" identifierref="RES-556FC54A-4F">...</item>
    <item identifier="ITEM-45FC0456-C" isvisible="true" identifierref="RES-A0624C24-B2">...</item>
    <item identifier="ITEM-B08369AA-1" isvisible="true" identifierref="RES-2AD0CCE2-DD">...</item>
    <item identifier="ITEM-2C4B2CE2-3" isvisible="true" identifierref="RES-530007DF-07">...</item>
    <item identifier="ITEM-3392D7B4-7" isvisible="true" identifierref="RES-608591D8-3D">...</item>
    <item identifier="ITEM-EB7C9675-1" isvisible="true" identifierref="RES-B28C7060-CE">...</item>
    <item identifier="ITEM-1FF1A92B-9" isvisible="true" identifierref="RES-36144ED2-CA">...</item>
    <item identifier="ITEM-1F8CAF2E-F" isvisible="true" identifierref="RES-58840837-2C">...</item>
    <item identifier="ITEM-044E0296-D" isvisible="true" identifierref="RES-2D686FE5-DF">...</item>
    <imsss:sequencing>...</imsss:sequencing>
  </organization>
</organizations>
```

Figure 3.6: Manifest organization.

what files constitute this resource is probably unknown when the “xmlreader” reach this point (It is possible that this resource depends on another resource, and that resource is defined way below this one). Therefore, keeping track of what resources exist in this XML file, and the content for each resource is necessary.

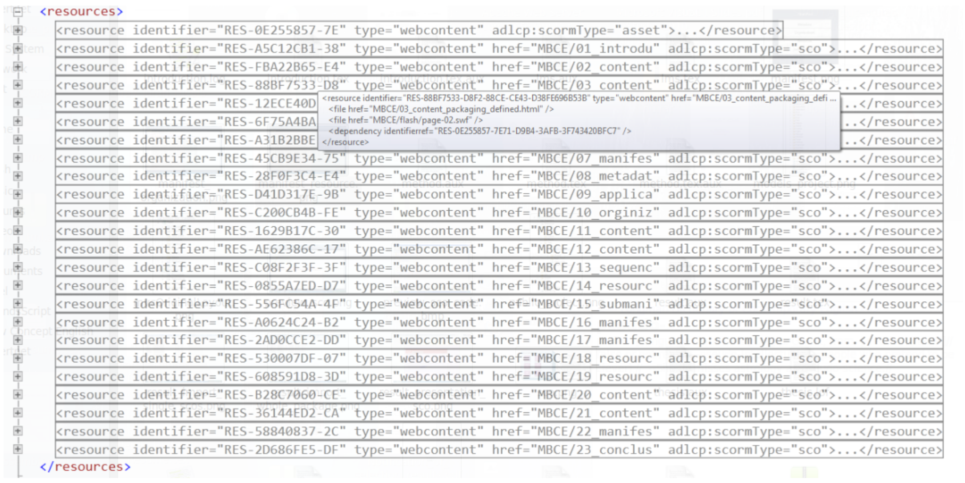
The implementation for resource is similar to above: “Resource” and “Resources” are created to save this information for later processing.

In the last step, “interpret” the resources, and figure out what files constitute each resource. The “translation” is done using recursion. The reason why recursion is used is that the level of including is unknown. In other words, it is possible that one resource depends on another resource, which depends on another resource, which depends on another resource, ... Until now, the relation between SCO and its related files is established. The rest is clear; copy the related files to the fold represented by this SCO, and save the information to the database.

Specifically, the information will be saved to QuestionSetting table (even though there is not much to do, since the settings are stored in QuestionSCOSetting), Question, QuestionSCOSetting and QuestionRight tables.

3.4 Exporting

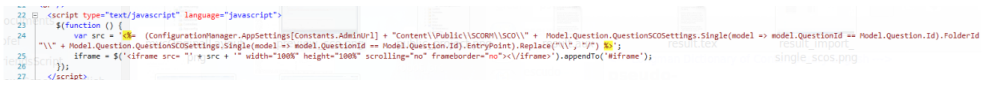
Exporting SCORM compliant package is the reverse of the above process, technically. Once the importing is fully understood, this section should be quite straightforward. This part is finished by Johan Simonsson from Entergate company.



```
<resources>
    <resource identifier="RES-0E255857-7E" type="webcontent" adlcp:scormType="asset">...</resource>
    <resource identifier="RES-A5C12CB1-38" type="webcontent" href="MBCE/01_introdu" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-FBA22B65-E4" type="webcontent" href="MBCE/02_content" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-88BF7533-D8" type="webcontent" href="MBCE/03_content" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-12ECE40D                                                                    ...</resource>
    <resource identifier="RES-6F75A4BA                                                                    ...</resource>
    <resource identifier="RES-A31B2BBE                                                                    ...</resource>
    <resource identifier="RES-45CB9E34-75" type="webcontent" href="MBCE/07_manifes" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-28F0F3C4-E4" type="webcontent" href="MBCE/08_metadat" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-D41D317E-9B" type="webcontent" href="MBCE/09_applica" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-C200CB4B-FE" type="webcontent" href="MBCE/10_orginiz" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-1629B17C-30" type="webcontent" href="MBCE/11_content" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-AE62386C-17" type="webcontent" href="MBCE/12_content" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-C08F2F3F-3F" type="webcontent" href="MBCE/13_sequenc" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-0855A7ED-D7" type="webcontent" href="MBCE/14_resourc" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-556FC54A-4F" type="webcontent" href="MBCE/15_submani" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-A0624C24-B2" type="webcontent" href="MBCE/16_manifes" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-2AD0CCE2-DD" type="webcontent" href="MBCE/17_manifes" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-530007DF-07" type="webcontent" href="MBCE/18_resourc" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-608591D8-3D" type="webcontent" href="MBCE/19_resourc" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-B28C7060-CE" type="webcontent" href="MBCE/20_content" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-36144ED2-CA" type="webcontent" href="MBCE/21_content" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-58840837-2C" type="webcontent" href="MBCE/22_manifes" adlcp:scormType="sco">...</resource>
    <resource identifier="RES-2D686FE5-DF" type="webcontent" href="MBCE/23_conclus" adlcp:scormType="sco">...</resource>
</resources>
```

```
<resource identifier="RES-88BF7533-D8F2-88CE-CE43-D38FE696B53B" type="webcontent" href="MBCE/03_content_packaging_defi ...
  <file href="MBCE/03_content_packaging_defined.html" />
  <file href="MBCE/flash/page-02.swf" />
  <dependency identifierref="RES-0E255857-7E71-D9B4-3AFB-3F743420BFC7" />
</resource>
```

Figure 3.7: Manifest resource.



```
<script type="text/javascript" language="javascript">
    $(function () {
        var src = '<%=  (ConfigurationManager.AppSettings[Constants.AdminUrl] + "Content\\Public\\SCORM\\SCO\\" +  Model.Question.QuestionSCOSettings.Single(model => model.QuestionId == Model.Question.Id).FolderId
"\\" + Model.Question.QuestionSCOSettings.Single(model => model.QuestionId == Model.Question.Id).EntryPoint).Replace("\\", "/") %>';
        iframe = $('<iframe src= "' + src + '" width="100%" height="100%" scrolling="no" frameborder="no"><\/iframe>').appendTo('#iframe');
    });
</script>
```

Figure 3.8: Code sample of PreviewSCORMQuestion.ascx

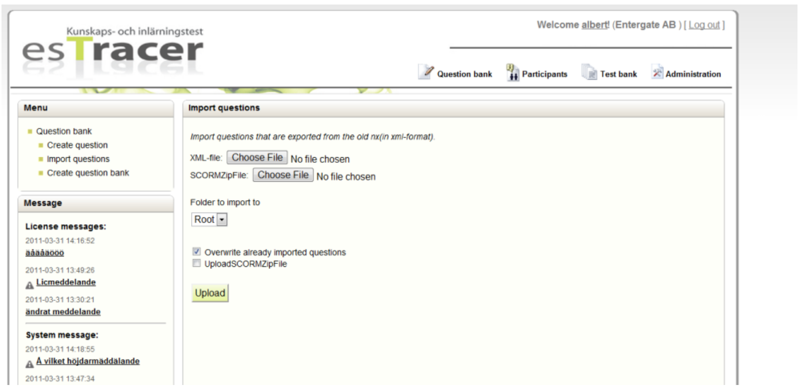
3.5 Presenting Single SCO

In this section, the work will be around "Public" project. One file named "PreviewSCORMQuestion.ascx" is responsible for the presentation of all the SCOs. Figure 3.8 illustrates the content of it. The content of the SCO is inserted to one <iframe >using javascript for the presentation. Since the communication is not implemented for now, all the work is just to find the path to the entry point of the SCO from the database, and pass it to the javascript.

Chapter 4

Result

In this chapter, result of this project is presented by using one package from ADL. In this way, this system can take advantage of existing SCORM compliant packages. The user interface for such functionality is shown below.



License messages:
2011-03-31 14:16:52
ååäåäooo
2011-03-31 13:49:26
Licmeddelande
2011-03-31 13:30:21
ändrat meddelande

System message:
2011-03-31 14:18:55
Å vilket höjdarmäddälande
2011-03-31 13:47:34

Import questions that are exported from the old nx(in xml-format).

XML-file: No file chosen
SCORMZipFile: No file chosen

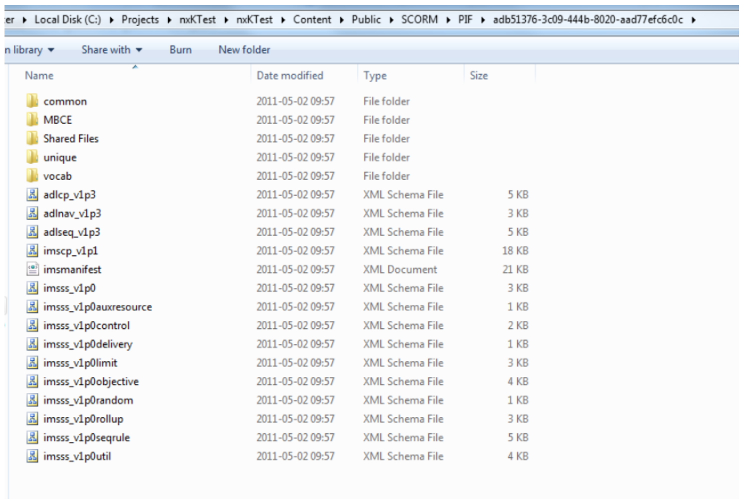
Folder to import to
Root

[✓] Overwrite already imported questions
[ ] UploadSCORMZipFile

Figure 4.1: Importing User Interface.

4.1 Import whole package

Manifest Basics Content Example Content Package [18] is used. Figure 4.1 illustrates the interface for importing. The check box "UploadSCORMZipFile" is used to indicate whether user wants to upload SCORM zip file or just XML file. After importing, one folder was created for this package. The name of this folder is one GUID, which can prevent guessing and name-conflict. Figure 4.2 shows the content of this new created folder, which coincides with the content of this zip file. In other words, if the package is unzipped, the content is exactly the same to what is contained in this folder.



er ▸ Local Disk (C:) ▸ Projects ▸ nxKTest ▸ nxKTest ▸ Content ▸ Public ▸ SCORM ▸ PIF ▸ adb51376-3c09-444b-8020-aad77efc6c0c ▸

Name	Date modified	Type	Size
common	2011-05-02 09:57	File folder	
MBCE	2011-05-02 09:57	File folder	
Shared Files	2011-05-02 09:57	File folder	
unique	2011-05-02 09:57	File folder	
vocab	2011-05-02 09:57	File folder	
adlcp_v1p3	2011-05-02 09:57	XML Schema File	5 KB
adlnav_v1p3	2011-05-02 09:57	XML Schema File	3 KB
adlseq_v1p3	2011-05-02 09:57	XML Schema File	5 KB
imscp_v1p1	2011-05-02 09:57	XML Schema File	18 KB
imsmanifest	2011-05-02 09:57	XML Document	21 KB
imsss_v1p0	2011-05-02 09:57	XML Schema File	3 KB
imsss_v1p0auxresource	2011-05-02 09:57	XML Schema File	1 KB
imsss_v1p0control	2011-05-02 09:57	XML Schema File	2 KB
imsss_v1p0delivery	2011-05-02 09:57	XML Schema File	1 KB
imsss_v1p0limit	2011-05-02 09:57	XML Schema File	3 KB
imsss_v1p0objective	2011-05-02 09:57	XML Schema File	4 KB
imsss_v1p0random	2011-05-02 09:57	XML Schema File	1 KB
imsss_v1p0rollup	2011-05-02 09:57	XML Schema File	3 KB
imsss_v1p0seqrule	2011-05-02 09:57	XML Schema File	5 KB
imsss_v1p0util	2011-05-02 09:57	XML Schema File	4 KB

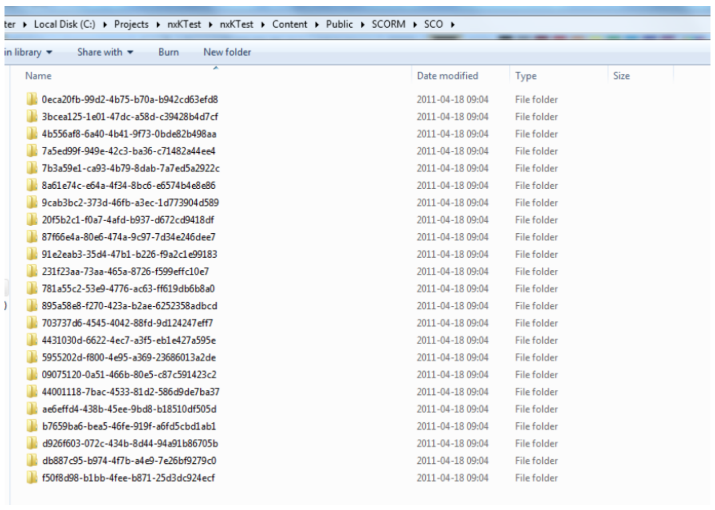
Figure 4.2: Result of importing whole package.

4.2 Import single SCOs

Manifest Basics Content Example Content Package [18] is used. The user interface is the same. The way to distinguish the two different importing(whole package or single SCO) has not implemented on the interface. It could be implemented as one check box. However, after consulting the company, I decided to hardcode the if condition, since someone in the company is particularly responsible for the user interface designing. In other words, the expression to evaluate within “if” is always true if I want to import the whole package, or always false if I want to import the SCOs. After importing, a few folders are created for this package. They have one-to-one map to the SCOs in this package. Figure 4.3 shows all the new created folders.

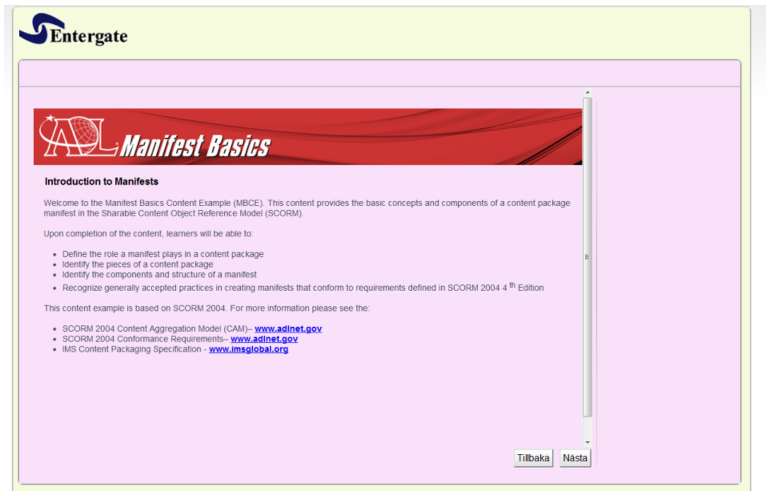
4.3 Presentation of SCO

The “Introduction to Manifests” SCO in Manifest Basics Content Example Content Package [18] is used, which is one very simple SCO, for it contains only one HTML page. Usually, one SCO will contain more than one HTML pages, then the SCO is responsible for the internal navigation. While navigating from one HTML page to another, the SCO might communicate with the LMS, which requires the LMS to implement all the APIs. Figure 4.4 presents the content of this SCO.


ter ▸ Local Disk (C:) ▸ Projects ▸ nxKTest ▸ nxKTest ▸ Content ▸ Public ▸ SCORM ▸ SCO ▸
in library ▾
Share with ▾
Burn
New folder

Name	Date modified	Type	Size
0eca20fb-99d2-4b75-b70a-b942cd63efd8	2011-04-18 09:04	File folder	
3bcea125-1e01-47dc-a58d-c39428b4d7cf	2011-04-18 09:04	File folder	
4b556af8-6a40-4b41-9f73-0bde82b498aa	2011-04-18 09:04	File folder	
7a5ed99f-949e-42c3-ba36-c71482a44ee4	2011-04-18 09:04	File folder	
7b3a59e1-ca93-4b79-8dab-7a7ed5a2922c	2011-04-18 09:04	File folder	
8a61e74c-e64a-4f34-8bc6-e6574b4e8e86	2011-04-18 09:04	File folder	
9cab3bc2-373d-46fb-a3ec-1d773904d589	2011-04-18 09:04	File folder	
20f5b2c1-f0a7-4afd-b937-d672cd9418df	2011-04-18 09:04	File folder	
87f66e4a-80e6-474a-9c97-7d34e246dee7	2011-04-18 09:04	File folder	
91e2eab3-35d4-47b1-b226-f9a2c1e99183	2011-04-18 09:04	File folder	
231f23aa-73aa-465a-8726-f599effc10e7	2011-04-18 09:04	File folder	
781a55c2-53e9-4776-ac63-ff619db6b8a0	2011-04-18 09:04	File folder	
895a58e8-f270-423a-b2ae-6252358adbcd	2011-04-18 09:04	File folder	
703737d6-4545-4042-88fd-9d124247eff7	2011-04-18 09:04	File folder	
4431030d-6622-4ec7-a3f5-eb1e427a595e	2011-04-18 09:04	File folder	
5955202d-f800-4e95-a369-23686013a2de	2011-04-18 09:04	File folder	
09075120-0a51-466b-80e5-c87c591423c2	2011-04-18 09:04	File folder	
44001118-7bac-4533-81d2-586d9de7ba37	2011-04-18 09:04	File folder	
ae6effd4-438b-45ee-9bd8-b18510df505d	2011-04-18 09:04	File folder	
b7659ba6-bea5-46fe-919f-a6fd5cbd1ab1	2011-04-18 09:04	File folder	
d926f603-072c-434b-8d44-94a91b86705b	2011-04-18 09:04	File folder	
db887c95-b974-4f7b-a4e9-7e26bf9279c0	2011-04-18 09:04	File folder	
f50f8d98-b1bb-4fee-b871-25d3dc924ecf	2011-04-18 09:04	File folder	

Figure 4.3: Result of importing single SCOs.



Introduction to Manifests

Welcome to the Manifest Basics Content Example (MBCE). This content provides the basic concepts and components of a content package manifest in the Sharable Content Object Reference Model (SCORM).

Upon completion of the content, learners will be able to:

- Define the role a manifest plays in a content package
- Identify the pieces of a content package
- Identify the components and structure of a manifest
- Recognize generally accepted practices in creating manifests that conform to requirements defined in SCORM 2004 4 th Edition

This content example is based on SCORM 2004. For more information please see the:

- SCORM 2004 Content Aggregation Model (CAM)– www.adlnet.gov
- SCORM 2004 Conformance Requirements– www.adlnet.gov
- IMS Content Packaging Specification - www.imsglobal.org

Figure 4.4: Presentation of single SCO.

Chapter 5

Discussion and Conclusion

In this chapter, author's personal opinion is presented towards some technologies in this project. In addition, the future work is discussed so that someone can continue with this.

5.1 Modular programming

Thanks to modular programming applied in the system architecture, it is possible to integrate the new function to it without fully understanding all of it. The biggest challenge in such project is to find the right place, and insert the new function.

In this project, the traditional approach is taken to find the portion of code responsible for the import function. Based on the knowledge of MVC routing, several breakpoints are set to check the execution flow in order to get one high view of this system architecture. This approach will always work so long as the developer choose the appropriate abstraction level.

After detecting the right place, inserting the new function is straightforward especially if the function is not very complex. However, great care needs to be taken to ensure the modularity. This requirement is quite subtle, and there is no absolute criteria to determine whether it is right or not.

Because of the constrain of ASP.NET MVC, it is not easy to add one new function. Therefore, the original function is expanded so that it has the new functionality. Hopefully, this will keep the modularity of this system.

5.2 ASP.NET VS ASP.NET MVC

Because the name "ASP.NET MVC" contains the "ASP.NET", the first impression would be that ASP.NET MVC is based on ASP.NET technology. In fact, ASP.NET MVC is one extension of the original framework, but in the end user point of view, the two do not share much, neither in syntax nor design. Microsoft has tried to make windows form model development for web application development. Consequently, this approach breaks the stateless nature of web, which makes this approach not suitable for large and complex project [19]. In “ASP.NET MVC”, “ASP” means the ASP engine, to generate the HTML files on the fly, “.NET” means the .NET framework, which is the software framework of almost all Microsoft product, “MVC” means the model-view-controller pattern.

From my experience of learn them, ASP.NET MVC is more clear and easier to pick up. It gives the developer one high view to see the whole picture without being bother by all kinds of details. In other words, it is more like the top-down programming paradigm.

5.3 MVC architect

This architect is particular suitable for large application, for the “separation of concerns” is expressed to great extent. Among the three components, model is the “heart” of the application, which is also the most fundamental and complex one. I didn’t realize that until I encountered much obstacles just reading the code without fully understanding how the model works, which reflects how tables in the database relate to each other. In fact, it makes perfect sense that model is the core, for controllers retrieve information from model and send it view and get information from view and send it to model. It’s impossible for developers to understand the behavior of controllers without knowing the structure of model.

5.4 C Sharp Programming Language

After using C Sharp in this project for approximately two month, my personal opinion is that it is really an enterprise-level language. By enterprise-level, I mean it is quite concise compared to some other languages.(Imagine that one language can finish the work using five lines while another will have to use ten lines. The most significant part is that all these do not break readability.) At first, it might seem alien language, but once one get used to the syntax, he or she can almost generate working code nearly as fast as he or she can type.

Microsoft went great length to make this language concise without breaking the logic of the whole picture. Two examples will be presented in here. The first example will be “properties”. Properties are type members that provide functionality that is a cross between fields and methods. One can read and write to a property just like a field. Additionally, code could be defined that runs whenever someone reads to or writes from a property, similar to methods [20]. The counterpart in Java would be “setters” and “getters”, but what could be easier than the “=” symbol?

The second examples would be the extension method [21]. Extension methods enable developers to “add” methods to existing types without creating a new derived type, recompiling, or otherwise modifying the original type. The counterparts of C Sharp have similar features. Take Java as one example, such functionality would be achieved by using composition of super class. However, this syntactic sugar becomes more obvious with LINQ (Language Integrated Query) [22], which utilizes extension method a lot.

However, LINQ will probably not be that powerful as it seems. While developing one system, that interacts with database, developers usually set up a abstraction layer of APIs so that this system could cope with various databases. Technically, LINQ is one more sophisticated way achieving it. Unfortunately, most systems using LINQ are using SQL servers from Microsoft, which hides this power to some extent.

Influenced by C++, Java, C Sharp inherits the merits of them and improves the pitfalls. With the help of a plethora of libraries in ".NET framework", C Sharp is indeed one competitive language.

5.5 Future Work

This project only covers the very first step to make one LMS SCORM compliant. In fact, there are three high level tasks to undertake. Firstly, developers will need to create an import mechanism to ingest SCORM compliant content into the system. This involves parsing an XML document called the imsmanifest and it's associated metadata to discover the structure of the content. This is covered in this project. Secondly, developers will need to develop a launching and tracking mechanism to deliver the content. The heart of this system is an ECMAScript (JavaScript) API that allows the content to persist and retrieve data according to the CMI data model. In other words, these APIs enable enable the communication between LMS and SCO. The final step is to develop an implementation of the SCORM / IMS sequencing engine that controls the navigation between parts of a SCORM compliant course (this last step is the trickiest).

The last two tasks are untouched in this project. Therefore, there is still a long way to go before "esTracer" is fully SCORM compliant.

Bibliography

[1] *Advanced Distributed Learning* http://www.adlnet.gov/About/Pages/Default.aspx

[2] *Rustici Software* http://scorm.com/

[3] *Learning Management System* http://www.adlnet.gov/Technologies/Lab/default.aspx

[4] *Learning Content Management System* http://www.adlnet.gov/Technologies/Lab/default.aspx

[5] *Authoring Tools* http://www.adlnet.gov/Technologies/Lab/default.aspx

[6] *Aviation Industry Computer-Based Training Committee* http://www.aicc.org/

[7] *IMS Global Learning Consortium* http://www.imsglobal.org/

[8] *IEEE Learning Technology Standards Committee* http://www.ieeeltsc.org/

[9] *SCORM Explained* http://scorm.com/scorm-explained/business-of-scorm/

[10] *SCORM 2004* http://scorm.com/scorm-explained/technical-scorm/scorm-2004-overview-for-developers/

[11] *SCORM Cloud* http://scorm.com/scorm-solved/scorm-cloud/

[12] *Tin Can* http://scorm.com/tincan/

[13] *Sharable Content Object Reference Model (SCORM) 2004 4th Edition Content Aggregation Model (CAM) Version 1.1, 2009.* Advanced Distributed Learning (ADL)

[14] Joe Mayo *Microsoft Visual Studio 2010 - A Beginner's Guide* ISBN: 978-0-07-166896-5

[15] *NerdDinner Tutorial* http://nerddinnerbook.s3.amazonaws.com/Part1.htm

[16] *NerdDinner Tutorial* http://nerddinnerbook.s3.amazonaws.com/Part4.htm

[17] *C#ziplib* http://www.icsharpcode.net/opensource/sharpziplib

[18] *Manifest Basics Content Example Content Package* http://www.adlnet.gov/Technologies/scorm/SCORMSDocuments/2004 4th Edition/Content Examples.aspx

[19] Shiju Varghese *ASP.net MVC Vs ASP.net Web Form* http://weblogs.asp.net/shijuvarghese/archive/2008/07/09/asp-net-mvc-vs-asp-net-web-form.aspx

[20] *The C Sharp Station Tutorial* http://www.csharp-station.com/Tutorials/Lesson10.aspx

[21] *The C Sharp Programming Guide* http://msdn.microsoft.com/en-us/library/bb383977.aspx

[22] *LINQ:.NET Language-Integrated Query* http://msdn.microsoft.com/library/bb308959.aspx