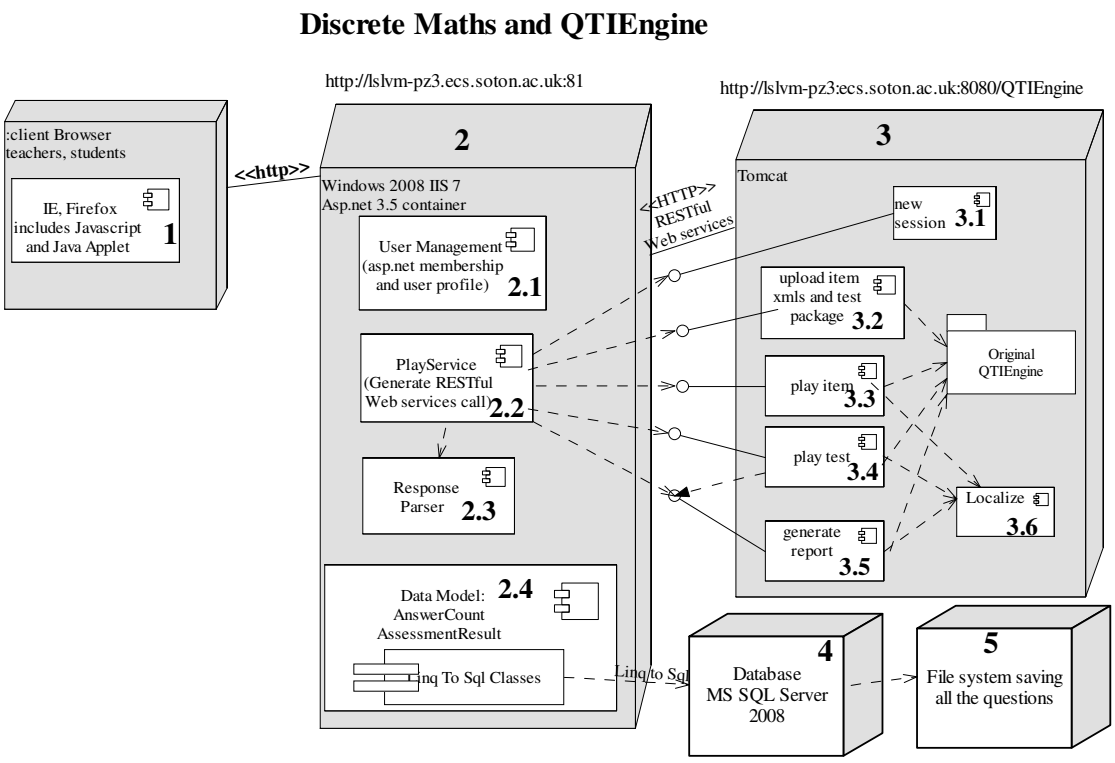


1. Client Browser

This is a web-based application. What end users need, is a browser with Javascript and Java applet enabled. See Known_issue.docx about the java applet issues.

2. Discrete Maths User Management

This is a Web application locating at http://lslvm-pz3.ecs.soton.ac.uk:81. The server is Windows 2008 IIS 7 with asp.net 3.5 installed. The development also uses dlls such as asp.net mvc, but I have copied all the necessary dlls with the project when it is deployed. This application is implemented in asp.net mvc 2, see WebFramwork.docx for detail.

2.1 User management system

Setup

This section is implemented automatically by asp.net membership and user profile. Asp.net membership and profile provides an easy way to create database for user, role, group and profile management. What you have to do is create new asp.net membership database using aspnet_regsql command (http://weblogs.asp.net/scottgu/archive/2005/08/25/423703.aspx) and then do some configuration work in web.config:

Connection String:

```
<connectionStrings>
        <add name="MathsConnecctionString" connectionString="Data
Source=LSLVM-PZ3\SQLEXPRESS;Initial Catalog=Maths;User
ID=sa;Password=12345678" providerName="System.Data.SqlClient"/>
    </connectionStrings>
```

Authentication: (in system.web tag)

```
<authentication mode="Forms">
            <forms loginUrl="~/Account/LogOn" timeout="2880"/>
</authentication>
```

Membership Provider: (in system.web tag)

```
<membership defaultProvider="MathsMembershipProvider">
            <providers>
                <add name="MathsMembershipProvider"
type="System.Web.Security.SqlMembershipProvider"
connectionStringName="MathsConnecctionString" enablePasswordRetrieval="false"
enablePasswordReset="true" requiresQuestionAndAnswer="false" applicationName="DiscreteMaths"
requiresUniqueEmail="true" passwordFormat="Hashed" maxInvalidPasswordAttempts="5"
minRequiredPasswordLength="7" minRequiredNonalphanumericCharacters="0"
passwordAttemptWindow="10" passwordStrengthRegularExpression=""/>
            </providers>
        </membership>
```

Usage

Get current User:

```
MembershipUser user = Membership.GetUser(HttpContext.User.Identity.Name)
```

Get current User Id:

```
(Guid)user.ProviderUserKey
```

Controller Access Control:

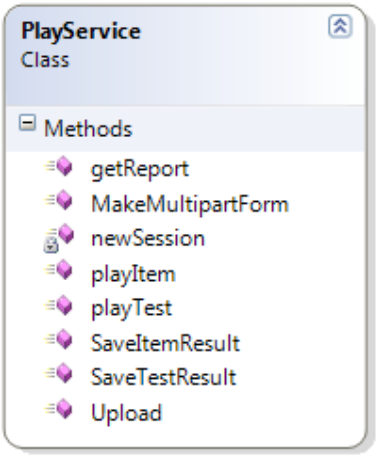
Sometimes for some action in controller, we need to specify that only users in role “teacher” and “administrator” can access. So we can add the a class RequiresRoleAttribute.cs, see ~/Services/RequiresRoleAttribute.cs”.

We require that only teacher or administer can create a new question, so we add “RequiresRole” tag before Create() method:

```
[RequiresRole(RolesToCheckFor = new string[] { "teacher", "administrator" })]
        public ActionResult Create()
        {
            ViewData["CategoryId"] = getCategoryIds();
            return View();
        }
```

2.2 PlayService

PlayService (“~/Services/PlayService.cs”) is a class contains method to make httpWebRequest call to QTIEngine RESTful webservices. It also saves response result for items or tests into database.



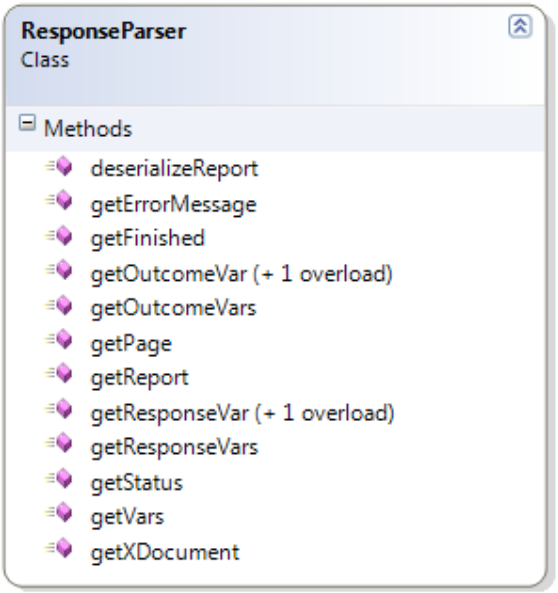
getReport(), newSession(), playItem(), playTest(), Upload() correspond to the method name of RESTful Web services in QTIEngine. The usage of this method can be found in playItem.docx and playTest.docx.

MakeMultipartForm() method will generate the post string sent with httpWebRequest.

SaveItemResult() method saves the item response into database.

SaveTestResult() method saves the test report into database.

2.3 ResponseParser



ResponseParser (“~/Services/ResponseParser.cs”) parses the response xml. The example of

return xml string can be found in Section 3.

2.4 Data Model

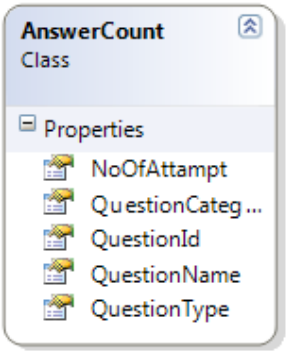
All the classes for data model are at: ~/Models . Most of the classes in data model are The file of auto-generated classes is at: ~/Models/Maths.dbml.

AccountModel.cs
Auto-generated class for user management in asp.net MVC.

Maths.dbml
Generated automatically by Linq to SQL. It is a mapping to SQL Server database. Please open it…and see the class diagram. The Question Class has two subclasses, Item and Test. The type of a Question instance is either an Item or Test. The "Class" field in Question class specifies which class this Question is. The value of "Class" field is either "Item" or "Test" as String.

In order to let asp.net MVC auto-generate list for questions and user answers. We also created some views such as: vw_UserAnswerQuestion and vw_QuestionUserCategory. If we want to list all the questions, only the QuestionCategoryId will be listed on the screen, but what user want to see is the CategoryName (not the id of category). So the view can join the Question and QuestionCategory together. Then asp.net MVC can create table automatically for the fields or columns in vw_QuestionUserCategory (we can also do it manually, but not necessary)

AnswerCount.cs
This class is written by hand. We need to display how many times a certain user (or current logged in user) have done this item or test and this information needs to be displayed along with the question information such as Question Name, Category etc. But unfortunately, there's no such class in database, so it is very difficult to use asp.net MVC to display this table. So we create this class.



Currently, this class is only used in /UserQuestion/Index.aspx. Following code is the usage of this class in UserQuestionController.cs

```
public ActionResult Index()
        {
            MembershipUser user = Membership.GetUser(HttpContext.User.Identity.Name);
```

```
            var uaqs = from uaq in db.vw_UserAnswerQuestions
                       where uaq.UserId == (Guid)user.ProviderUserKey
                       group uaq by uaq.QuestionId into g
                       select new DiscreteMaths.Models.AnswerCount
                       {
                           QuestionId = g.Key,
                           QuestionName = db.Questions.Single<Question>(q => q.QuestionId ==
g.Key).QuestionName,
                           QuestionType = db.Questions.Single<Question>(q => q.QuestionId ==
g.Key).Class,
                           QuestionCategoryName = db.Questions.Single<Question>(q =>
q.QuestionId == g.Key).QuestionCategory.QuestionCategoryName,
                           NoOfAttampt = g.Count()
                       };

            return View(uaqs);
        }
```

AssessmentResult.cs

Similar to AnswerCount.cs, AssessmentResult.cs is created to display test report. The report retruned by QTIEngine RESTful Web Services can be deserialized to the instance of this class. This class is generated by xsd.exe from "~/XSD/AssessmentResult.xsd". AssessmentResult.xsd is part of ims qti 2.1 specificaion http://www.imsglobal.org/xsd/imsqti_v2p1.xsd. The use of this class can be found in playTest.docx.

3 QTIEngine Server

It is a tomcat server running at http://lslvm-pz3.ecs.soton.ac.uk:8080/QTIEngine. The original QTIEngine dosen't provide any web service interface for remote application to retrieve. What we have done is adding RESTful Web services to use the functions provided by QTIEngine. These methods can be found at QTIEngine/grails-app/RestController.groovy

3.1 newSession() method

Physical Location: newSession()
Sample Url: http://lslvm-pz3.ecs.soton.ac.uk:8080/QTIEngine/rest/newSession
Input params: none
http post sample: none
Output: a session id as String. 5F471331466D8A6E0E83D1440CCC972B

3.2 Upload File()

Physical Location: upload()
Sample Url:
http://lslvm-pz3.ecs.soton.ac.uk:8080/QTIEngine/rest/upload;jsessionid=D8417904066FEEE26E9584A2CCC81528?actionUrl=http://localhost:51237/Question/PlayItem/21&imagesUrl=http://localhost:51237/Content/Questions/Images/
Input params: jsession (required), actionUrl (required) ,imageUrl (optional, if it's necessary to specify image url.)
http post sample: uploadPost.txt
Output: upload.xml
actionUrl and imageUrl are sent to QTIEngine RESTful Web services through url QueryString

3.3 Play Item

Physical Location: playItem()
Sample Url:
http://lslvm-pz3.ecs.soton.ac.uk:8080/QTIEngine/rest/playItem/0;jsessionid=D8417904066FEEE26E9584A2CCC81528
Input params: jsession (required) ,actionUrl(requried)
http post sample: playItemPost.txt
Output: playItem.xml
actionUrl will be a parameter in the post form

3.4 Play Test

Physical Location: playTest()
Sample Url:
http://lslvm-pz3.ecs.soton.ac.uk:8080/QTIEngine/rest/playTest/0;jsessionid=7AAA0AD5209623A7E943B5013892A068
Input params: jsession (required), actionUrl (required) ,imageUrl (optional, if it's necessary to specify image url.)
http post sample: playTestPost.txt
Output: playTest.xml, if the test is finished: playTestFinished.xml

3.5 Generate Report (only for test, not item)

Physical Location: report()
Sample Url:
http://lslvm-pz3.ecs.soton.ac.uk:8080/QTIEngine/rest/playTest/0;jsessionid=7AAA0AD5209623A

7E943B5013892A068
Input params: jsession (required, in query string), actionUrl(required, we insert it in the post form). There should be a parameter with key as "report"
http post sample: reportPost.txt
Output: report.xml

The "exit" command is similar to report.

3.6 Localize (private)

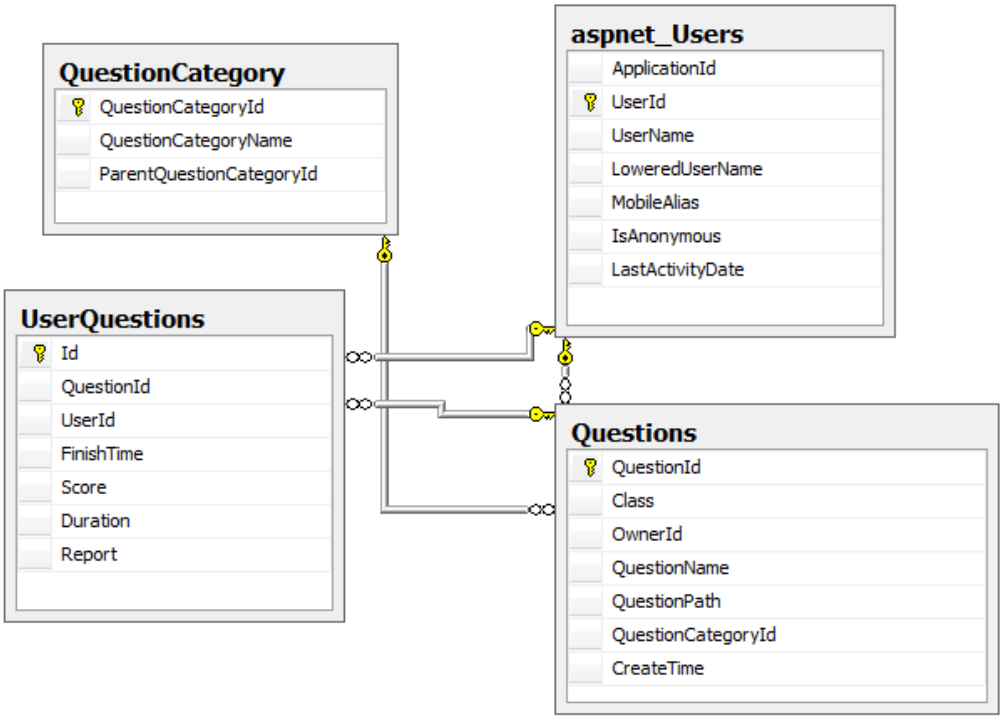
Localize is not a public interface. It can only been used inside QTIEngine. This method is in RestService.groovy in QTIEngine/grails-app/services/. The returned xhtml page from QTIEngine has many problems. A lot of @href and @src urls (link src, action url, image url, etc. See Known Issues.docx) are not correct. This method uses simple string replacement to replace the urls with the correct ones.

Explanation about localize:
The original QTIEngine can return an xhtml page. But unfortunately, it's only a page without any status information about current response. In addition, the answered form will be posted directly to the default service in QTIEngine, which means we cannot insert anything in this process. For example, if a user is doing a test, and I want to write his or answer information into database EVERY TIME HE OR SHE SUBMITS THE ANSWER, it's impossible to do it. But if we change the form action to some server programme (.asp or .php), we can insert the functions we want to execute before the http post send data to QTIEngine. That's why we need to replace the action filed in form.

When we play an item, we just send the item xml file. Some items also have images to display. But current QTIEngine API doesn't support sending image files with http post. So the users have to upload the image somewhere accessible on internet and specify the image url in the http post. Then the localize() method will replace the img src with the url users pass to QTIEngine. It's not necessary to do it in the test, because test will upload the whole package into QTIEngine including the image. In this case, we just need to replace the img src with a server path, so users don't need to pass imageUrl.

4 SQL Server 2008 database

The database name is Maths. Username "sa" , password "12345678". Asp.net membership and user profile has been setup in the database.



5 File system

All the questions are saved in a file directory on the server. In database, we only save the relative path of the item xml or test package. We did this because some items need to display one or more images. It's possible to save images in the database, but in the item xml, the image has been specified by a reference to a relative path of the item xml file. So if we save images into database, we also have to change the image references in item xml, which is very inconvenient. So we decide that we still save item and test files on the file system.

The path of all the files is ~/Content/Questions.
In order to avoid name conflict when uploading item or test files, we change the file name when the files are saved on the server. The actual file name will be: original file name + guid + file extension.