DART Developer's Manual

Team 2: Jingtao Tong, Xinwen Wang, Guoxiong Xie, Mengzhe Li

Table of Contents

List of Figures

1. What is DART?

It is often necessary for a group of possibly non co-located stakeholders to collaboratively assess and ultimately prioritize project risks. Unfortunately, with worsens sharply with increasing numbers of stakeholders, risk assessment becomes nearly impossible when stakeholders are not all in the same place at the same time. The Distributed Assessment of Risks Tool (DART) is a means to aid risk assessment efforts by providing a web based interface for gathering stakeholder assessments, aggregate results, generate a top-n risk list, and track reassessments.

DART is used to provide a means to address these issues with respect to project risk assessment and tracking. The tool provides an easy to use interface for project stakeholders to continuously perform and monitor top-n risk assessments. This includes acquiring risk exposure estimates for distributed stakeholders, reporting current risk assessments and priorities, gathering risk mitigation and impact information, and graphing risk exposure changes. The information can be easily exported to external tools such as Excel and text based CSV format.

For the whole system's information security, our system set up three different log in status, they are: TA, Regular User and Project Manager. They have different functions and authorities to this system, so after they log in with their own status, it will be different user interface.

2. How Do I Get Started With DART Development?

The Developer's Guide will provide you information about the current database design and technologies used in the application supplemented by screen-shots.

The code base is available to fork or download from https://github.com/GuoxiongXie/DART. For more information about the github repository, please contact Guoxiong Xie at felixxie1218@gmail.com.

3. General Technologies used

Frontend: HTML, CSS, JavaScript (for form evaluation and verification).

Backend: PHP (for controller mechanism), MySQL (for database),
MyPhpAdmin (set up database manually),
ADOdb (database abstraction library for PHP)

We would recommend the developer to use XAMMP stack as their development environment. XAMPP is an Apache distribution containing MySQL, PHP and Perl. XAMPP is very easy to install and to use - just download, extract and start: http://www.apachefriends.org/en/xampp.html

4. Database Description and Design

The DART system has three user modes: TA, Project Manager and Regular User. TA can set up a project, assign a manager, view all projects' assessment results, and export vote data into CSV format. Manager can add stakeholders, edit stakeholders, add risks, edit risks, modify project information, do risk assessment, close a voting session, view assessment results (the project he/she involved in) and export as a CSV file. In regular user mode, the regular user can view his/her project information, add risks, do risk assessment, view risk assessment and export it as a CSV file.

Since stakeholders in the project are divided into three groups and they have different authorities, we maintain a database keeping track of these three types of users:

Type 1: Teaching Assistance

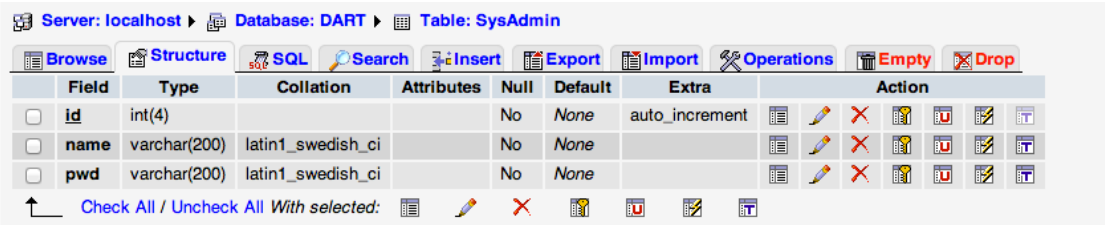

Figure 1. SysAdmin Table

Figure 1 SysAdmin Table is the table keeping track of the current TA. The table has three attributes:

Attribute Name	Description
id	Primary key of the table; auto incremented.
name	The name of the TA
Pwd	Password of the TA

Type 2: Project Manager

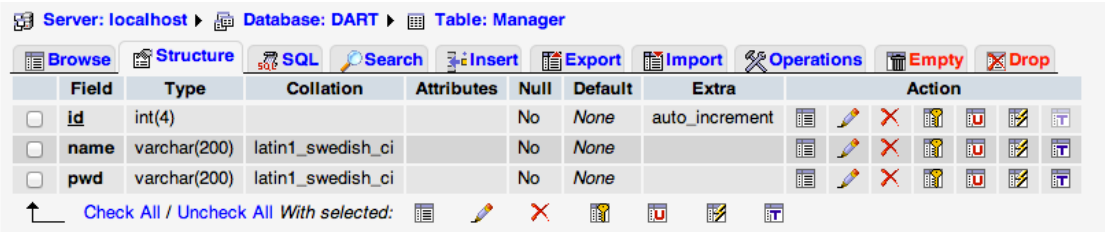

Figure 2. Manager Table

Figure 2 Manager Table is the table keeping track of the current project manager. The table has three attributes:

Attribute Name	Description
id	Primary key of the table; auto incremented.
name	The name of the manager.
Pwd	Password of the manager.

Type 3: Regular Users

Server: localhost ▸ Database: DART ▸ Table: RegularUser

Browse | Structure | SQL | Search | Insert | Export | Import | Operations | Empty | Drop

Field	Type	Collation	Attributes	Null	Default	Extra	Action
id	int(4)			No	*None*	auto_increment	
name	varchar(200)	latin1_swedish_ci		No	*None*		
pwd	varchar(200)	latin1_swedish_ci		No	*None*		

Check All / Uncheck All *With selected:*

Figure 3. RegularUser Table

Figure 3 RegularUser Table is the table keeping track of the current project regular user. The table has three attributes:

Attribute Name	Description
id	Primary key of the table; auto incremented.
name	The name of the regular user.
Pwd	Password of the regular user.

Since stakeholders are involved in projects, we have a table keeping track of current projects:

Server: localhost ▸ Database: DART ▸ Table: Project

Browse | Structure | SQL | Search | Insert | Export | Import | Operations | Empty | Drop

Field	Type	Collation	Attributes	Null	Default	Extra	Action
id	int(4)			No	*None*	auto_increment	
projectname	varchar(200)	latin1_swedish_ci		No	*None*		
projectdesc	varchar(800)	latin1_swedish_ci		No	*None*		
closed	tinyint(1)			No	0		
lastAssessmentDate	varchar(200)	latin1_swedish_ci		No	*None*		

Check All / Uncheck All *With selected:*

Figure 4. Project Table

Figure 4 Project Table is the table keeping track of the current project . The table has five attributes:

Attribute Name	Description
id	Primary key of the table; auto incremented.
projectname	The name of the project.
projectdesc	The description of the project.
closed	A flag that signifies whether the project's voting session has been closed.
lastAssessmentDate	The last date a voting session is closed.

From the design above, we still do not know who is involved in which project. Thus, we added one more table—ProjMem—to maintain the mapping between projects and stakeholders involved in the project.

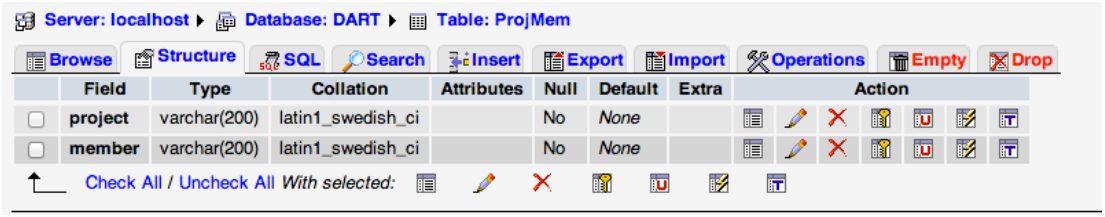

Figure 5. ProjMem Table

This application is a voting system where each members in the project vote on a risk item. Thus, there are some instances we need to keep track of, such as risk, probability of Undesired Outcome (PUO) and Size of Loss of Undesired Outcome (LUO) and each member's votes.

To keep track of risk items, we maintain a ProjRiskDesc table:

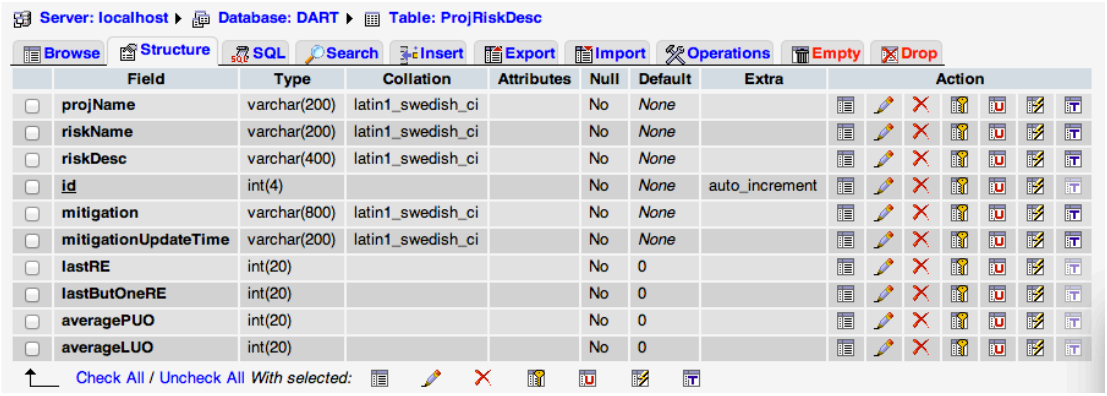

Figure 6. ProjRiskDesc Table

Figure 6 ProjRiskDesc Table has ten attributes:

Attribute Name	Description
id	Primary key of the table; auto incremented.
projName	The name of the project to which a risk item belongs.
riskName	The name of the risk item.
riskDesc	Description of the risk; enter when user creates a risk.
mitigation	A potential solution to the risk item.
MitigationUpdateTime	Indicates the latest date a mitigation plan is entered.
lastRE	The Risk Exposure value from last voting session.
lastButOneRE	The Risk Exposure value from last but one voting session.
averagePUO	The average value of PUO in a voting session.
averageLUO	The average value of LUO in a voting session.

The IindividualVote Table keeps track of each stakeholder's vote on a particular risk item:

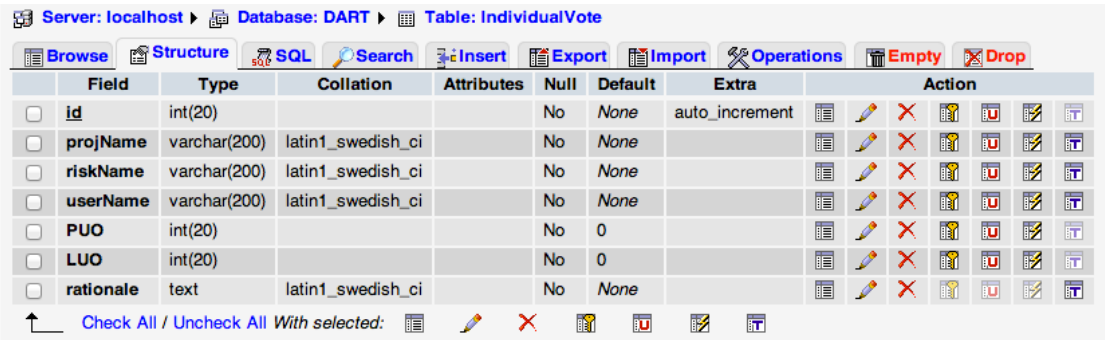

Figure 7. IndividualVote Table

Figure 7 IndividualVote Table has seven attributes:

Attribute Name	Description
id	Primary key of the table; auto incremented.
projName	The name of the project to which a risk item belongs.
riskName	The name of the risk item.
userName	The name of the stakeholder.
PUO	The PUO value a stakeholder votes on the risk item.
LUO	The LUO value a stakeholder votes on the risk item.
rationale	The rationale behind the votes.

4.1. ADOdb

We use ADOdb to maintain our database. ADOdb is a database abstraction library for PHP. It provides a set of API that make information accessing easy. For a complete lost of functions, please refer to http://adodb.sourceforge.net/.

Before running the application, please make sure you have specified the server in both conn.php and connect_to_mysql.php files located in include/ directory. For conn.php, you need to enter the server information in line 5 following the commented instructions in line 4 (Figure 8):

```
1 <?php
2     include_once("adodb5/adodb.inc.php");
3     $conn = ADONewConnection('mysql');
4     //You should change the line 5 below and set up database accordingly before running
5     $conn->PConnect('someHostName','someUserName','somePassword','someDatabaseName') or die('connection error');
6     $conn->Execute('set names gb2312');
7     $ADODB_FETCH_MODE = ADODB_FETCH_BOTH;
8 ?>
```

Figure 8. Instructions for entering server info in conn.php

Similarly, you need to provide server information in connect_to_mysql.php from line 2 to line 9 following the commented instructions (Figure 9):

```
1 <?php
2 // Place the host name for the MySQL database here
3 $db_host = "someHostName";
4 // Place the username for the MySQL database here
5 $db_username = "someUserName";
6 // Place the password for the MySQL database here
7 $db_pass = "somePassword";
8 // Place the name for the MySQL database here
9 $db_name = "someDatabaseName";
```

Figure 9. Instructions for entering server info in connect_to_mysql.php

To access the database, be sure to include conn.php file located in include/ directory as shown in Figure 10. The conn.php file specifies the server on which we run the application. Once conn.php file is included, we can access the database with the variable $conn.

```
<?php
//This action happens after manager clicks on "close session" button.
//note that this file is only for manager to close voting period.
//The authority will be checked immediately.

include_once 'include/conn.php';
session_start();
```

Figure 10. Example of including conn.php

5. Check Authority using Session Variables after Logging In

Since we have three different authorities in DART—admin(TA), manager, and user (regular user), it is important to check their authorities before proceeding. When someone logs in to our system, we store his/her username and authority in in $_SESSION variable. If we want to access them, we can do $_SESSION['username'] to get the username and $_SESSION['authority'] to get the authority. Note: the authority for TA is "admin", that for project manager is "manager", and that for regular user is "user". Figure 11 shows an example of checking the authority of the log in person:

```php
<?php
//This action happens after manager clicks on "close session" button.
//note that this file is only for manager to close voting period.
//The authority will be checked immediately.

include_once 'include/conn.php';
session_start();

$role = $_SESSION['authority']; //manager, admin, user; here it only can be manager

if ($role != "manager"){
    echo "<script>alert('Sorry, but you have to be one of the project managers to close a voting period!');</script>";
    echo "<script language='javascript'>window.location.href='setup.html';</script>";
}
```

Figure 11. Example of Checking Authority

6. Other Documentations

Most of the files in the code base are well commented. There is a description about the file and purpose at the beginning of each file. For example, in closeVotingPeriod.php, we have

```
<?php
//This action happens after manager clicks on "close session" button.
//note that this file is only for manager to close voting period.
//The authority will be checked immediately.

include_once 'include/conn.php';
session_start();
```

Figure 12. Example of Comments at the Beginning of Each File

We also provide a user's manual for this application, which can be downloaded from https://github.com/GuoxiongXie/DART.

For more information about DART, please contact Guoxiong Xie at felixxie1218@gmail.com.