

# Technical Reference

## Zodiac.NET Survey Engine Toolkit Release 1.0.0.0

© 2009 MentorLogic Ltd.  
All rights reserved.

MentorLogic is a leading software provider for software utilities, reusable components and libraries.

This document is part of the company products documentation series.

For more information or ordering of this product or others products, please contact:  
[support@mentor-logic.com](mailto:support@mentor-logic.com)

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electrical, mechanical, photocopying, recording, scanning or otherwise except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher.

Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where MentorLogic Ltd. is aware of a claim, the product names appear in initial Capital or ALL CAPITAL LETTERS.

Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

# Contents

Scope .....	5
About Zodiac.NET .....	5
1. Architecture Overview .....	5
1.1 Assemblies .....	6
1.2 Components.....	6
1.2.1 Survey Viewer Web Control.....	6
1.2.2 Questions Web Controls.....	7
1.2.3 Survey Object Models.....	8
1.2.4 Repositories.....	8
2. XML Definition File .....	9
2.1 Meta Element .....	10
2.2 Pages Element .....	10
2.3 Questions Element.....	10
2.3.1 Text Question .....	11
2.3.2 Paragraph Question .....	11
2.3.3 Single Choice Question.....	12
2.3.4 Multi Choice Question .....	13
2.3.5 Matrix Question.....	13
3. Survey Themes.....	14
About MentorLogic.....	21



## Scope

This document was designed for the purpose of providing a reference for architects, developers and software engineers wishing to develop and customize Zodiac.NET Survey Engine in their own projects and applications. It is recommended that the reader of this document have a reasonable grasp of Microsoft .NET technology, ASP.NET, XML and Client Scripting.

## About Zodiac.NET

Zodiac.NET is a .NET web based survey and form engine software toolkit for Microsoft .NET written in pure managed C#. It enables NET developers to install and host their own online survey applications written in state of the art Microsoft ASP.NET C#. Using Zodiac.NET, developers are able to create questionnaires, surveys and exams wizards in their own .NET web projects in an easy and customizable way. The engine provides you with several types of questions to fit the different needs of the survey makers. Zodiac.NET combines XML and ASP.NET to create a customized component to fit the different needs and requirements of the different .NET projects.

## 1. Architecture Overview

Zodiac.NET support developers with the ability to create surveys with different types of questions. Developers are able to create single choices questions, multi choices question, text/essay questions and matrix questions. Defining of the questions details are made through XML which enhance the integration capability of the engine with the existing projects and systems.

User responses are collected in XML files as well which allow software developers to have custom analysis on the generated files and easily integrate the engine output with their projects flow and logic.

The separation of the presentation, business logic and the storage handlers was the main methodology in designing the engine. The engine presentation layer consists of a collection of user controls for rendering the survey wizards and questions. This layer is represented by the main survey viewer web control with a collection of helper user controls used by the viewer to render each question type in a different way.

To provide an extensible strategy for data storage, the data storage is handled by XML repository to load the XML definition file and construct the object model. The repository is used by the user controls to read the survey XML definition files and render the contents of the survey to the end user. The following diagram illustrates the different layers and the main building blocks of Zodiac.NET.

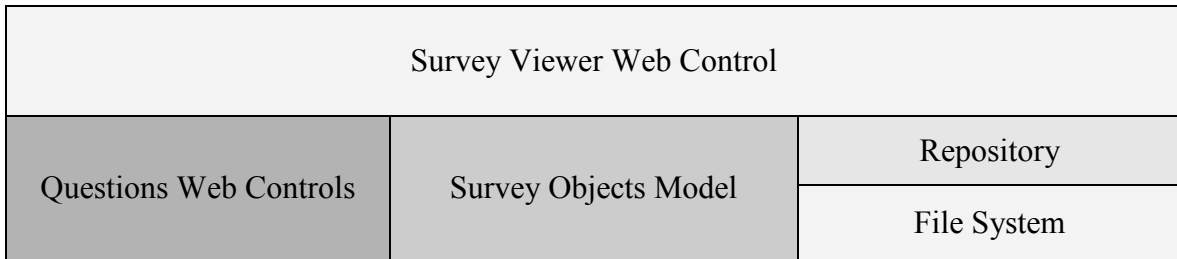


Figure 1: Zodiac.NET Architecture Design

## 1.1 Assemblies

Zodiac.NET shipped in two different .NET assemblies. The assemblies can be found in the bin folder after decompressing the purchased package. The following table lists the details of each assembly:

Assembly	Description
MentorLogic.Common	This assembly contains the common customized user controls used within the engine. The assembly contains for example customized HTML controls for radio buttons lists, checkbox lists and web helpers classes.
MentorLogic.Engines.Zodiac	This assembly contains the engine core logic and components. It contains the survey web controls, the object model and the repositories. It contains also the core classes for reading and parsing the survey XML files.

## 1.2 Components

### 1.2.1 Survey Viewer Web Control

The survey viewer – *SurveyViewerWebControl.ascx* - is a web control which can be used in any ASP.NET web page to render the survey XML definition file. The web control is part of the MentorLogic.Engines.Zodiac.Web namespace. The following sample shows you how to use this control in ASP.NET web page.

```
<Zodiac:SurveyViewerWebControl
    ID="ucSurveyWebControl"
    XmlFilePath="C:\\zodiac\\samples\\demo.xml"
    OutputFolder="C:\\zodiac\\samples\\Responses\\"
    CssClass="bluesky"
    runat="server">
</Zodiac:SurveyViewerWebControl>
```

### 1.2.1.1 Properties

Property	Description
ID	The web control unique ID.
XmlFilePath	The file path of the XML definition file.
OutputFolder	The folder path in which to save the user responses.
CssClass	The CSS class name to be applied to the survey viewer web control.

### 1.2.1.2 Events

Event	Description	Arguments
OnResponseRecieved	This event is raised when a user has finished the survey and his response is ready to be saved.	SurveyResponseArgs: Contains the user response XML representation and the response ID.

## 1.2.2 Questions Web Controls

The Questions Web Controls are a collection of custom user control internally used by the survey viewer user control to render and generate the UI of the survey.

Each question type has its own web user control. The following is a list of the different questions controls:

- TextQuestionWebControl
- ParagraphQuestionWebControl
- SingleChoiceQuestionWebControl
- MultiChoiceQuestionWebControl
- MatrixQuestionWebControl

Each question web control inherit the abstract custom control *QuestionWebControl* and override the rendering methods according to each question type (e.g. the text question inputs would be rendered in different way than the multi choice question inputs. The text question inputs will be rendered as text box to receive the user inputs however the multi choice questions input will be rendered as a list of checkboxes). This custom control is responsible to render the common elements in all the question types like the question body and remarks.

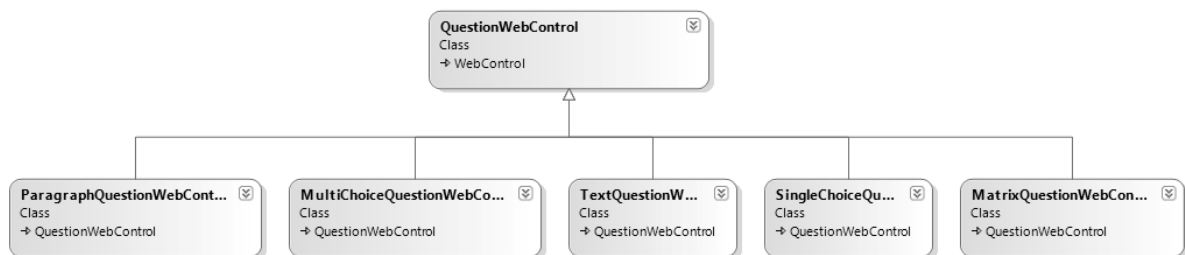


Figure 2: Questions Web Controls Class Diagram

Each question web control override two methods in *QuestionWebControl*:

Method	Description
CreateQuestionInputsArea	The method which renders the question inputs. Each question type will render its inputs in different way.
CreateQuestionFooterArea	The method which renders the question footer and validation controls/scripts. Each question type may have its own footer elements and validation controls.

### 1.2.3 Survey Object Models

The object model hierarchy of the survey engine is similar to the web controls hierarchy. As illustrated in the class diagram, there is a separate class model for each question type. Each class of the question types is inherited from *QuestionBase* abstract class. The question base contains the common properties and methods served by all the questions types (e.g. the rendering of the question text body and the remarks are both common for each question. Each question will have its question text body and remarks text). The following diagram represents the class diagram of the engine object model.

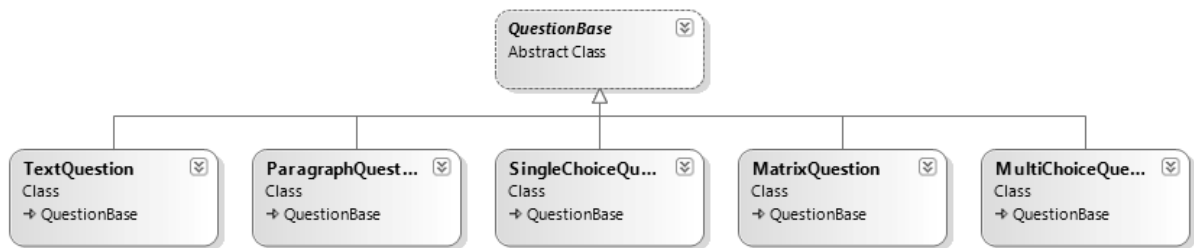


Figure 3: Object Model Class Diagram

### 1.2.4 Repositories

Zodiac.NET uses the repository pattern to save and retrieve the survey objects. The engine provides an out-of-the-box repository for storing and saving the surveys as XML file: *XmlSurveyRepository*. However, you can extend the survey repository to save in different medium like database by implementing *ISurveyRepository* interface and use it instead of the default *XMLSurveyRepository* to store and retrieve the survey object model. (Note: You will need to have the professional license to modify the source code.)

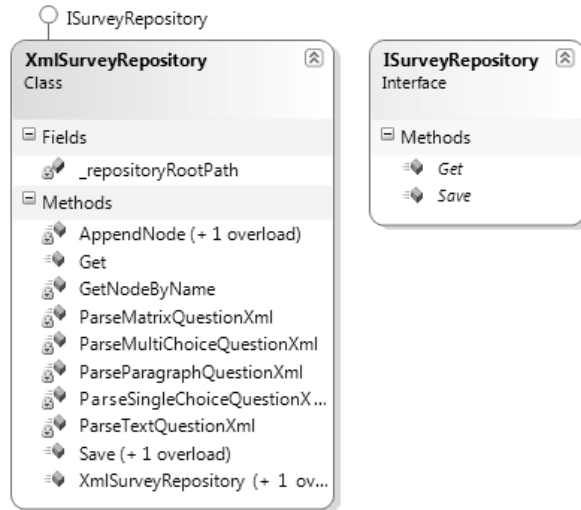


Figure 4: Repository Class Diagram

## 2. XML Definition File

The power of Zodiac.NET is in defining the survey details as XML. The survey details and questions can be defined in one solid and structured XML format. The following XML show how the survey data is structured and defined.

```
<?xml version="1.0" encoding="utf-8" ?>
<survey>
  <meta>
    <id/>
    <title/>
    <description/>
    <endtitle/>
    <enddescription/>
  </meta>
  <pages>
    <page/>
    <page/>
    <page/>
  </pages>
  <questions>
    <question/>
    <question/>
    <question/>
    <question/>
  </questions>
</survey>
```

## 2.1 Meta Element

The Meta element contains the main details of the survey such as survey name and description. The following table lists the children elements of the Meta tag:

Element	Description
id	This is a unique ID for the survey. It's up to the developer to set it to the suitable identification he may flow in his own application.
title	The title of the survey. This title will be displayed on the survey first page.
description	The description of the survey. The description will be displayed on the survey first page.
endtitle	The title which will be displayed for the user in the last page after finishing the survey.
enddescription	The description text which will be displayed for the user in the last page after finishing the survey.

## 2.2 Pages Element

The pages element contains the details of the survey pages. The pages can be considered as the wizard steps of the survey. Each survey should have at least one page. Each page will contains one or more questions. The following table lists the children elements of the Pages tag:

Element	Description
Page	The page element represents a wizard step in the survey. The page element contains 3 attributes: <ul style="list-style-type: none"><li>- id: a unique identification for the page.</li><li>- title: the page title which be displayed to the user at the top of the page (before the questions).</li><li>- description: the page description which will be displayed to the user at the top of the page (before the questions).</li></ul>

## 2.3 Questions Element

The questions element contains the details of the survey questions including the question body, choices, constraints and validation criteria. Zodiac.NET provides five types of questions: Text Question, Paragraph Question, Single Choice Question, Multi Choice Question and Matrix Question. All the question types can be defined using the Question Element.

```
<question id="007" type="text" page="3"/>
```

The following table contains the different Question Element attributes.

Attribute	Description
id	Unique identifier for the question. It can be string or numerical.
type	It should be set to the type of the question. The possible values are: <i>text</i> , <i>paragraph</i> , <i>singlechoice</i> , <i>multichoice</i> and <i>matrix</i>
page	It should be set to the page

Now we will go through the different types of the questions that can be defined in the survey XML file and how to use the Question Element to define each question type.

### 2.3.1 Text Question

This represents a question in which you get a single line of text as a response from the user. The question can be defined using the Question Element as following:

```
<question id="007" type="text" page="3">
  <body>Would you please introduce yourself?</body>
  <remarks>Please enter your name in the following field</remarks>
  <footer>We don't distribute any personal details.</footer>
  <mandatory>>false</mandatory>
</question>
```

#### 2.3.1.1 Elements

Element	Description
body	The text of the question.
remarks	The remarks to be added to the bottom of the question body.
footer	The footer text to be added to the bottom of the question input.
mandatory	True or false. Specify if this question is mandatory or not.

### 2.3.2 Paragraph Question

This represents a question in which you get the user response as an essay or multi-line text.

```
<question id="004" type="paragraph" page="2">
  <body>What is required to enhance the product?</body>
  <remarks>Please feel free to provide your ideas</remarks>
  <footer>Your opinion is important</footer>
  <mandatory>>false</mandatory>
</question>
```

### 2.3.2.1 Elements

The following table states the allowed attributes and elements in the question element of type paragraph.

Element	Description
body	The text of the question.
remarks	The remarks to be added to the bottom of the question body.
footer	The footer text to be added to the bottom of the question input.
mandatory	True or false. Specify if this question is mandatory or not.

### 2.3.3 Single Choice Question

This represents a question in which the user chooses a single choice from a defined list. The user is also able to choose “other” option his choice is not defined in the list.

```
<question id="006" type="singlechoice" page="3">
  <body>How did you know about this product?</body>
  <remarks></remarks>
  <otherenabled>true</otherenabled>
  <dropdownlist>false</dropdownlist>
  <footer></footer>
  <mandatory>true</mandatory>
  <choices>
    <choice>Friend</choice>
    <choice>Search Engines</choice>
    <choice>Advertisment</choice>
    <choice>Blog Post</choice>
  </choices>
</question>
```

#### 2.3.3.1 Elements

The following table states the allowed attributes and elements in the question element of type “single choice”.

Element	Description
body	The text of the question.
remarks	The remarks to be added to the bottom of the question body.
otherenabled	True or False. Specify if it’s allowed for the user to enter “Other” option other than the ones in the choices list,
dropdownlist	True or False. Specify if the choices should be rendered as a list of check boxes or as a drop down list.
footer	The footer text to be added to the bottom of the question
mandatory	True or false. Specify if this question is mandatory or not.
choices	The choices of the question. The element contains one or more of the <choice> elements.

### 2.3.4 Multi Choice Question

This represents a question in which the user chooses zero or more choices from a defined list. The user is also able to choose “other” option in case his choice is not in the predefined list.

```
<question id="002" type="multichoice" page="1">
  <body>How will you use the product?</body>
  <remarks>Please specify how you wil use the product.</remarks>
  <otherenabled>true</otherenabled>
  <footer></footer>
  <mandatory>true</mandatory>
  <choices>
    <choice>Educational/Exams/Research</choice>
    <choice>Human Resources/Employee Statisfaction</choice>
    <choice>Customer Satisfaction/Feedback</choice>
    <choice>Market Research</choice>
  </choices>
</question>
```

#### 2.3.4.1 Elements

The following table states the allowed attributes and elements in the question element of type “multi choice”.

Element	Description
body	The text of the question.
remarks	The remarks to be added to the bottom of the question body.
otherenabled	True or False. Specify if it’s allowed for the user to enter “Other” option other than the ones in the choices list,
footer	The footer text to be added to the bottom of the question
mandatory	True or false. Specify if this question is mandatory or not.
choices	The choices of the question. The element contains one or more <choice> elements.

### 2.3.5 Matrix Question

This represents a question in which the user have a vertical list of aspects/items and horizontal ranking columns. The user is able to rank each vertical item versus each horizontal rank.

```

<question id="003" type="matrix" page="2">
  <body>How would you rate the product versus the following?</body>
  <remarks>The higher rank is better.</remarks>
  <footer>*Your ranking is important for.</footer>
  <mandatory>>false</mandatory>
  <columns>
    <column>Very Bad</column>
    <column>Bad</column>
    <column>Good</column>
    <column>Very Good</column>
    <column>Excellent</column>
  </columns>
  <rows>
    <row>Features</row>
    <row>Simplicity</row>
    <row>Integration</row>
    <row>Extensibility</row>
    <row>Customization</row>
    <row>License/Price</row>
  </rows>
</question>

```

### 2.3.5.1 Elements

The following table states the allowed attributes and elements in the question element of type “matrix”.

Element	Description
body	The text of the question.
remarks	The remarks to be added to the bottom of the question body.
footer	The footer text to be added to the bottom of the question
mandatory	True or false. Specify if this question is mandatory or not.
columns	This element represents the ranking to be listed in the matrix in horizontal alignment.
rows	This element represents the items to be ranked. The items are listed in vertical rows.

## 3. Survey Themes

Zodiac.NET engine generates the HTML elements with a predefined CSS classes. This means that each generated question HTML elements will have an assigned CSS class so that the developer is able to modify the look and feel of the surveys in their applications.

The package is shipped with a sample CSS theme containing the main CSS classes used by the engine when rendering the survey HTML elements. The CSS sample can be used as a reference for creating similar themes or even modifying it to fit the hosting application theme.

The following sample lists a sample theme called: bluesky which contains the main CSS styles used by the engine when rendering the HTML elements.

```

.bluesky
{
    margin: 80px;
    padding: 30px;
    background-color: #ffffff;
    display: block;
    border-style: solid;
    border-color: #a7d3ff;
    border-width: 30px;
    font-family: Arial, Verdana
}

.bluesky .survey-intro { }

.bluesky .survey-title
{
    display:block;
    font-size: 24px;
    padding:10px;
}

.bluesky .survey-desc
{
    display:block;
    font-size: small;
    padding:10px;
}

.bluesky .survey-page { }

.bluesky .seperator
{
    display:block;
    clear: both;
    border-top-style: solid;
    border-top-width: 1px;
    border-top-style: dotted;
    border-color: #000000;
    padding-bottom: 40px;
}

.bluesky .survey-question
{
    margin-bottom: 20px;
    font-size: 12px;
}

.bluesky .survey-question input[type="text"]
{
    border-style: solid;
    border-width: 1px;
    border-color: #c0c0c0;
    background-color: #f0f0f0;
    color: #505050;
    width: 300px;
}

```

```

/* --- Input Area --- */
.bluesky .paragraph-question-inputarea textarea
{
    display: block;
    width: 400px;
    height: 100px;
    margin-top: 20px;
    margin-bottom: 20px;
    border-style: solid;
    border-width: 1px;
    border-color: #c0c0c0;
    background-color: #f0f0f0;
    color: #505050;
}

.bluesky .singlechoice-question-inputarea input[type="text"]
{
    width: 100px;
    margin-left: 10px;
}

.bluesky .singlechoice-question-inputarea ul li
{
    list-style-type: none;
    display: block;
}

.bluesky .multichoice-question-inputarea ul li
{
    list-style-type: none;
    display: block;
}

.bluesky .multichoice-question-inputarea input[type="text"]
{
    width: 100px;
    margin-left: 10px;
}

.bluesky .matrix-question-inputarea { }
/* --- End Input Area --- */

.bluesky .question-body
{
    font-size: 16px;
    color: #83a5c8;
    font-weight: bold;
    display: block;
}

.bluesky .question-remarks
{
    display: block;
    font-size: 14px;
    color: #a0a0a0;
    border-width: 0px;
}

```

```

border-color: #b2b2b2;
border-bottom-style: dotted;
padding-top: 2px;
padding-bottom: 2px;
margin-bottom: 10px;
}

.bluesky .question-footer
{
display: block;
font-size: small;
color: #606060;
padding-bottom: 20px;
margin-top: 10px;
}

.bluesky .question-suffix
{
font-size: x-small;
color: #ff0000;
display: block;
padding-top: 10px;
padding-bottom: 20px;
}

.bluesky .question-choices
{
display: block;
border-top-width: thin;
padding-top: 20px;
}

.bluesky .question-radio
{
display: block;
}

.bluesky .question-radio-horz
{
list-style-type: none;
display: inline;
}

.bluesky .question-radio-horz li
{
display: block;
float: left;
width: 100px;
height: 20px;
padding: 10px;
font-weight: bold;
font-size: 12px;
color: #000000;
background-color: #f0f0f0;
}

.bluesky .question-radio-horz .rowvalue

```

```

{
    display:block;
    width:100px;
    text-align: center;
    font-weight: bold;
    color: #000000;
}

.bluesky .question-radio-horz-header
{
    display: inline;
}

.bluesky .question-radio-horz-header li
{
    display:inline;
    width:100px;
    height: 20px;
    padding: 10px;
    float:left;
}

.bluesky .question-radio-horz-header li label { }

.bluesky .question-radio-horz-header .rowtitle
{
    display:block;
    width:100px;
    text-align: center;
    background-color: #ffffff;
    font-weight: bold;
    font-size: 12px;
    color: #000000;
}

.bluesky .question-radio-horz-header .rowtitle-alt
{
    display:block;
    width:100px;
    text-align: center;
    background-color:#ffffff;
    font-weight: bold;
    font-size: 12px;
    color: #000000;
}

.bluesky .question-radio-horz-alternate
{
    list-style-type: none;
    display:inline;
}

.bluesky .question-radio-horz-alternate ul
{
}

```

```

.bluesky .question-radio-horz-alternate li
{
    display:block;
    float:left;
    width:100px;
    height: 20px;
    font-weight: bold;
    font-size: 12px;
    color: #000000;
    background-color: #d0d0d0;
    padding: 10px;
}

.bluesky .question-radio-horz-alternate .rowtitle
{
    display:block;
    width:100px;
    float:left;
    font-weight: bold;
    font-size: 12px;
    color: #000000;
}

.bluesky .question-radio-horz-alternate .rowvalue
{
    display:block;
    width:100px;
    text-align: center;
    font-weight: bold;
    color: #000000;
}

.bluesky .clear
{
    clear: both;
    display:block;
    height: 1px;
}

.bluesky .survey-end-title
{
    display:block;
    font-size: large;
    padding:10px;
}

.bluesky .survey-end-desc
{
    display:block;
    font-size: small;
    padding:10px;
}

.bluesky .buttons-bar
{

```

```
        text-align: center;
    }

    .bluesky .next-button
    {
        margin:5px;
        width: 100px;
        text-align: center;
    }

    .bluesky .prev-button
    {
        margin:5px;
        width:100px;
        text-align: center;
    }

    .error
    {
        color: #ff0000;
    }
}
```

## About MentorLogic

MentorLogic is a leading vendor of components for Microsoft .NET technologies, solutions and Products. MentorLogic helps customers build applications with unparalleled richness, responsiveness and interactivity. Created with passion, MentorLogic products help thousands of developers every day to be more productive and deliver reliable applications under budget and on time.

Website: <http://www.mentor-logic.com>

Support: [support@mentor-logic.com](mailto:support@mentor-logic.com)

Sales: [sales@mentor-logic.com](mailto:sales@mentor-logic.com)