This information is located at...

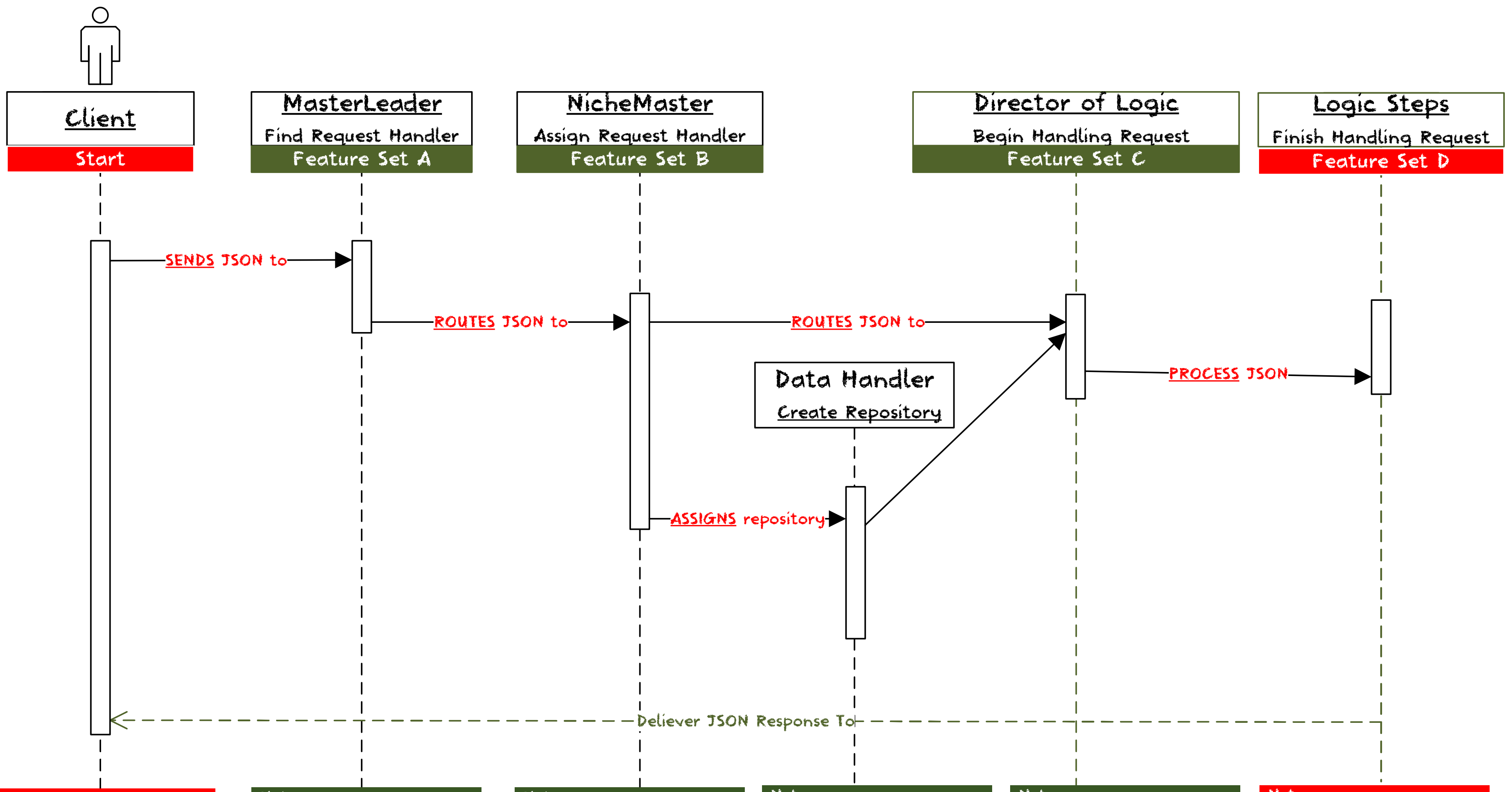
- C:\Local Drive | Google Drive | Dropbox | Cloud Storage Service
  - RTG
    - Productivity
      - 2. Teach
        - Mark Richardson
          - Strategic Planning
            - Goal Setting

The Automation Story 2-3 – Learn BaseDI - 1-1 - BaseDI.io _ Visual Studio Management - Strategic Planning _ Goal Setting - 1.0.vsdx

Taking the 1st Step with BaseDI for C#/Typescript Developers

GOAL – To learn the big picture and to take our FIRST STEP with BaseDI

A Quick Glimpse of BaseDI Core



Notes

- REMEMBER ANY client is referred to internally as basectl or Base Control

Notes

- SEARCH for MasterLeader in project to see SOURCE CODE example

Notes

- SEARCH for NicheMaster in project to see SOURCE CODE example

Notes

- SEARCH for LocalFile in project to see SOURCE CODE example

Notes

- REVIEW LESSON 6-D for list repositories available in BaseDI

Notes

- SEARCH for Director Of in project to see DIRECTOR SOURCE CODE example

Notes

- SEARCH for Experience The in project to see EXPERIENCE SOURCE CODE example

Notes

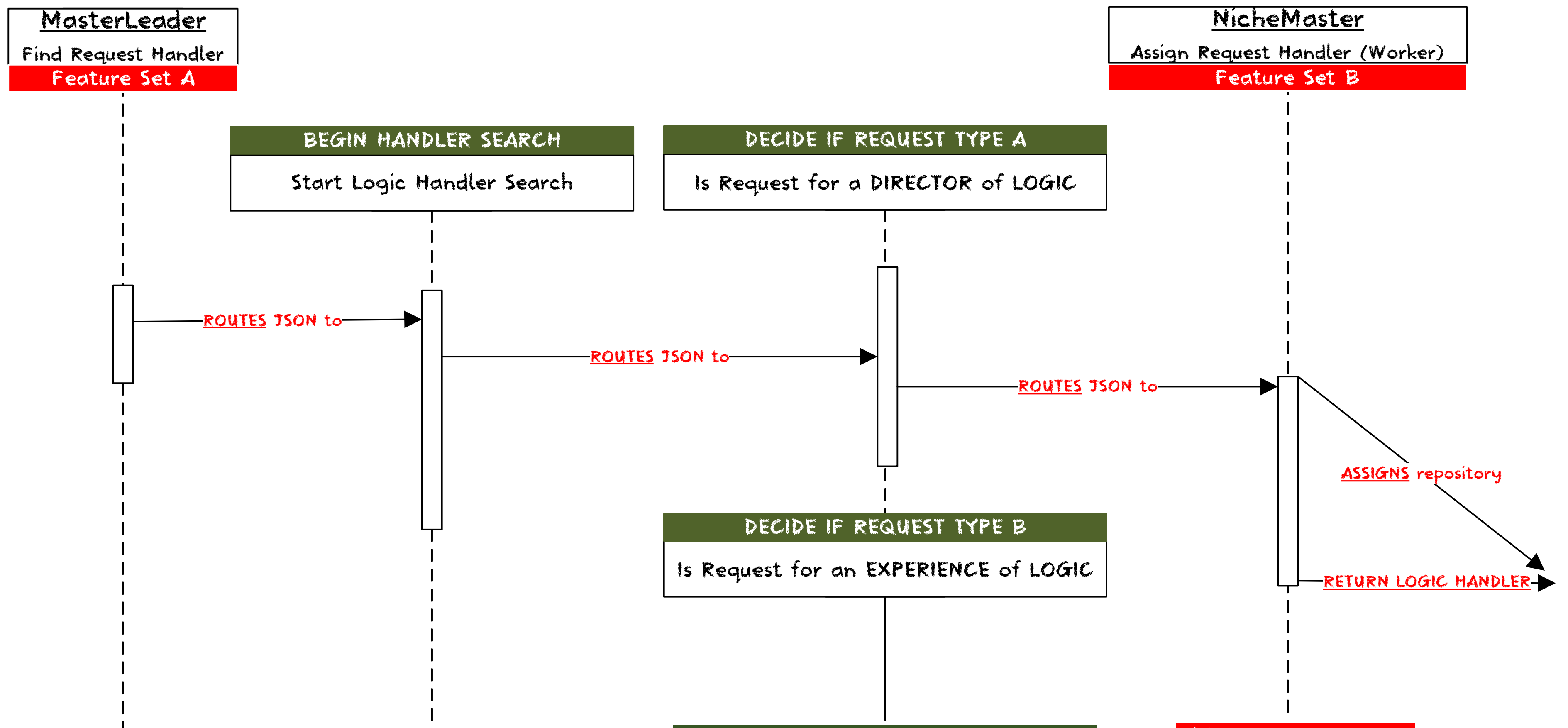
- SEARCH for Page in project to see SOURCE CODE example

Notes

- A "SERIES" of STEPS is called a Chapter Page. Like a page of a book.

A Quick Glimpse of The BaseDI Core Request Inspector

INSPECT FILE: ProgrammingStudioAdministrator_MasterLeader_12_2_1_0.cs



Notes

- SEARCH for MasterLeader in project to see SOURCE CODE example

Notes

- SEARCH for Use DesignPattern Builder Chapter 12_2_1_0 in MasterLeader file to see SOURCE CODE example

Notes

- SEARCH for Implement DesignPattern Builder Chapter 12_2_1_0 in MasterLeader file to see SOURCE CODE example

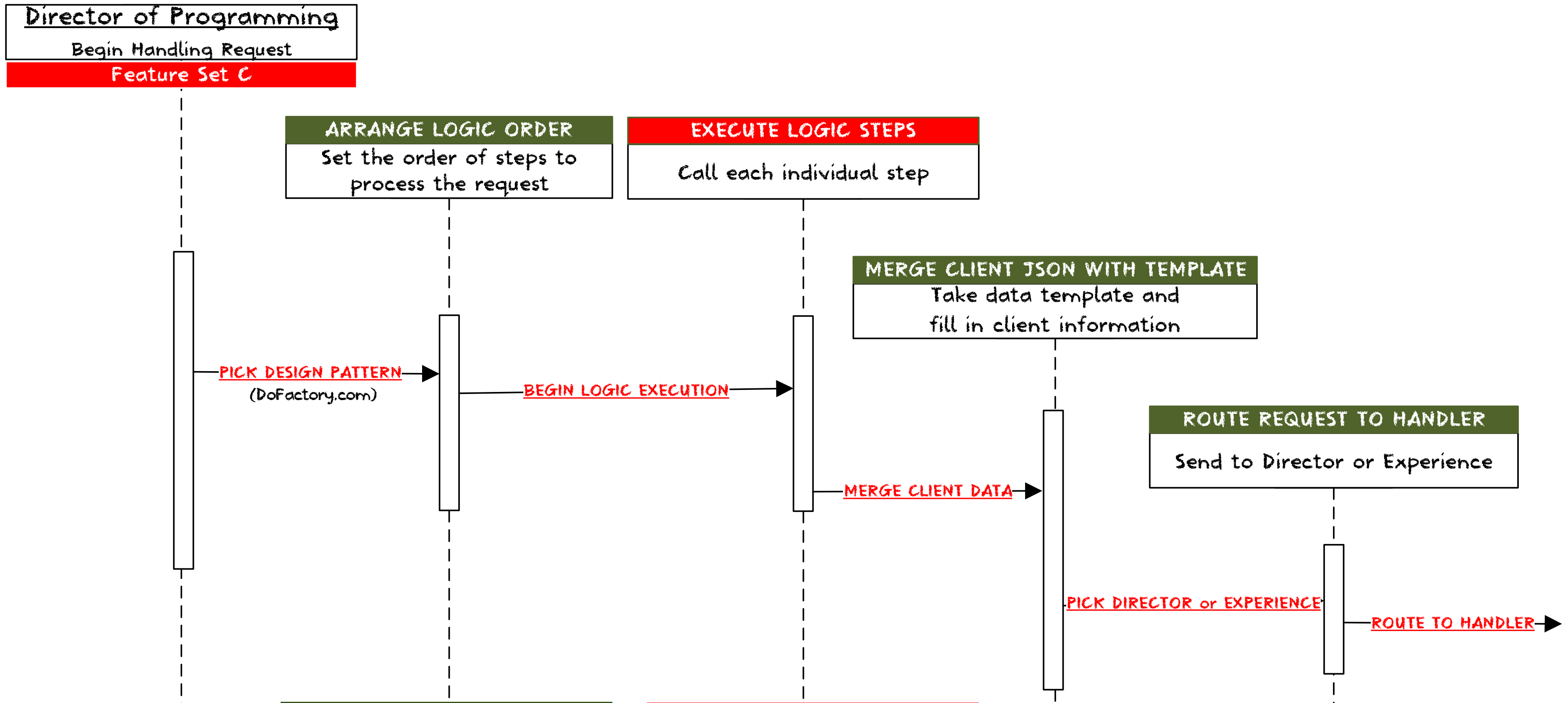
Notes

- SEARCH for NicheMaster in project to see SOURCE CODE example

A Quick Glimpse of The BaseDI Core Request Handler

INSPECT FILE: ProgrammingFactoryImplementer_NicheMaster_12_2_1_0.cs

GOTO DEFINITION: Director_Of_Programming_Chapter_12_2_Page_1_RequestHandler_1_0.cs



Notes

- SEARCH for RequestHandler in project to see SOURCE CODE example

Notes

- A DIRECTOR OF something is apart of the WORKER family

Notes

- AN EXPERIENCE OF something is apart of the WORKER family

Notes

- SEARCH for Use_DesignPattern in RequestHandler file to see SOURCE CODE example

Notes

- SEARCH for Implement_DesignPattern in RequestHandler file to see SOURCE CODE example

Notes

- SEARCH for Page_1_9_Verify_Process_12_2_1_0 in Project to see SOURCE CODE example

Notes

- SEARCH for Page_1_10_End_Process_12_2_1_0 in Project to see SOURCE CODE example

Step 0-A

HOW TO DOWNLOAD THE BASEDI TEST PROJECT

A **DOWNLOAD** the LATEST version of the UNIT TEST project from GitHub.com

REQUIRED: TO USE EXACT PATH AS LISTED BELOW...BASEDI NEEDS A PREDICTABLE PATH

C **LEARN**

What is BaseDI?

BaseDI is a programming framework used to help us successfully execute our goals

- It focuses on the AUTOMATION of GENERATING LEADS and SALES.

  (TO INCREASE REVENUE)
- It focuses on the AUTOMATION of ADMINISTRATION and MANAGEMENT TASK.

  (TO DECREASE EXPENSES)
- It focuses on the UNIFICATION of a TEAM.

  This means we want you to code in a PREDICTABLE way.

  This means giving a group of programmers a common set of GUIDELINES to follow.

  (TO IMPROVE COMMUNICATION)

B **VERIFY** FOLDER STRUCTURE AFTER DOWNLOAD

- Local Disk (C:)
  - Programming
    - https://github.com/NerdyGroupAffiliate/000.BaseDI.SourceCode.Community (PUBLIC REPO)
    - https://github.com/NerdyGroupAffiliate/000.BaseDI.SourceCode.Professional (PRIVATE REPO)
    - BaseDI Project Code Structure
      - 0. Script
      - 1. Storyline
      - 2. Character
      - 3. Setting
      - 4. Experience
      - 5. Chapter
      - 6. State
      - 7. Director
      - 8. Testing
      - 9. Organizing
    - https://github.com/NerdyGroupAffiliate/000.BaseDI.Assets.Community (PUBLIC REPO)
    - https://github.com/NerdyGroupAffiliate/000.BaseDI.Assets.Professional (PRIVATE REPO)
    - BaseDI Project Assets Structure
      - 1. Data Storage
      - 2. Data Movement
      - 3. Client
      - 4. CI-CD

Step 0-B

HOW TO TAKE OUR FIRST STEP WITH BASEDI

A **PREPARE** to learn about the BaseDI PROOF OF CONCEPT UNIT TEST TEMPLATE

B **REMEMBER** that BaseDI is HIGHLY STRUCTURED and ORANIZED

D **LEARN**

What is the TEMPLATE?

The template help us code proof of concepts
• The purpose of the template is to take our ideas and turn them into PROOF OF CONCEPTS (UNIT TEST).
• The template also has STRUCTURED SECTIONS of where we should place our code.

What are the LETTERS?

The letters in the file name descriptions	
G	This is the Goal Number. (SEE LESSON 1)
N	This is the Niche Number. (SEE EZINES.com for List)
P	This is the Story Page (Task) Number. (SEE LESSON 5)
V_V	This is the Version Number

NOTE: This pattern will be used going forward

C **OBSERVE** THAT THE SELECTED FILE IS THE UNIT TEST TEMPLATE FILE NAME

Name
Template_Director_Of_Niche_Chapter_12_3_Page_1_CreateWebDevelopmentForWebsite_Handler_1_0_Test <- EXAMPLE
Template_Director_Of_Niche_Chapter_G_N_Page_P_CRUDValue-Niche-Preposition-Noun-SmallDescription_Handler_V_V_Test
Template_Experience_The_Group_SmallDescription_SubGroup_G_N_V_V_Test.template
• Niche = Ezine.com Niche • G_N = Goal Number_Niche Number (See LESSION 1, Step 1-B for Goal List)
• Page_P = Page (Task) Number
• Underscore _ = Word Separator • Dash - = Word Concatenation
• Preposition = English Grammar (See List @ https://www.english-grammar-revolution.com/list-of-prepositions.html) • Noun = English Grammar (See List @ https://www.english-grammar-revolution.com/list-of-nouns.html • Group = (Pick one of the classifications from Lesson 4) • SubGroup = (Pick one of the classifications from Lesson 4) • SmallDescription = (Another Noun of Your Choice)
• V_V_Test = Version Number
• Template_ = REMOVE prefix when ready to code • .template = RENAME to .cs or .ts for typescript

Step 0-C

HOW TO TAKE OUR SECOND STEP WITH BASEDI

A PREPARE to learn about the BaseDI PROOF OF CONCEPT TEMPLATE REGIONS

B REMEMBER that each word inside the {} will be replaced based on an idea

C LEARN

What are we UPDATING?

We are filling in the following

- The Class Name
- The Class Constructor Name
- The Unit Test Method Name

Remember the LETTERS?

The letters to the right mean

G	This is the Goal Number. (SEE LESSON 1-B)
N	This is the Niche Number. (SEE EZINES.com for List)
P	This is the Story Page Number. (SEE LESSON 5)
V_V	This is the Version Number

NOTE: This pattern will be used going forward

C OBSERVE THAT THE HIGHLIGHTED AREAS IS WHAT YOU WILL BE CHANGING

```
public class  Direct_Niche_Chapter_G_N_Page_P_
              {CRUDValue}-{Niche}-{Preposition}-{Noun}-
              {SmallDescription}_V_V_Test
{
    #region 2. Ready

    public    Direct_Niche_Chapter_G_N_Page_P_
              {CRUDValue}-{Niche}-{Preposition}-{Noun}-
              {SmallDescription}_V_V_Test

    {

    }
    #endregion
    #region 4. Action

    [Test]
    public void  Did_Niche_Chapter_G_N_Page_P_
    {            {CRUDValue}-{Niche}-{Preposition}-{Noun}-
                 {SmallDescription}_V_V_Work()


    }
    #endregion
}
```

Step 0-D

HOW TO UNDERSTAND THE TEST REGIONS I

A REMEMBER that "EVERY" UNIT TEST will always have "4 DEFAULT REGIONS".

C LEARN

What are REGIONS?

Each region will start with a "#"

- A region's sole purpose is to GROUP and ORGANIZE code that share a similar purpose.

B NOTICE WHAT GOES INSIDE EACH REGION

Region "1. Assign"

- private JObject storylineDetails; is used for our BaseDI ARM Template JSON
- private JObject storylineDetails_Parameters; is the unique client information.
- private string baseDIArmTemplateSchemaEmbeddedResource;
  is used to set the CLIENT data SCHEMA.
- private string baseDIArmTemplateSchemaParametersEmbeddedResource;
  is used to set the actual CLIENT data .

Region "2. Ready"

- Sub Region "1. Assign"
  - Add any VARIABLES that are SHARED FOR ALL unit TEST "IN FILE".
- Sub Region "2. Action"
  - Call any METHODS that are SHARED FOR ALL unit TEST "IN FILE".
- Sub Region "3. Observe"
  - Leave blank for now.

Region "3. Set"

- Sub Region "1. Assign"
  - Add any INITIALIZATION that is SHARED FOR ALL unit TEST "IN FILE".
- Sub Region "2. Action"
  - Add any INITIALIZATION methods that is SHARED FOR ALL unit TEST "IN FILE".
- Sub Region "3. Observe"
  - Add any observations that are made from Region 2 that is SHARED FOR ALL unit TEST "IN FILE".

Step 0-E

HOW TO UNDERSTAND THE TEST REGIONS 2

Region "4. Action"

- Sub Region "1. Assign"
  - Add any INITIALIZATION that is for THIS unit TEST.
- Sub Region "2. Action"
  - This is where we add the code that calls and execute the Unit Test Logic

```
JObject armTemplateJSONOutput =
        new ProgrammingStudioAdministrator_MasterLeader_12_2_1_0<Director_Of_Programming_Chapter_12_2_Page_1_RequestHandler_1_0>()
            .SetupStoryline(_storylineDetails, _storylineDetails_Parameters).Result
            .Action().Result;
```

- Sub Region "3. Observe"
  - inspect and observe armTemplateJSONOutput JSON object
  - More Information about ARM Template Outputs Here https://blogs.msdn.microsoft.com/girishp/2015/06/16/azure-arm-templatestips-on-using-outputs/

Storyline Development

GOAL – To learn how to use the STRAGETY to HELP implement various FEATURES for an idea

Step 1-A

HOW TO KNOW HOW BASEDI REDUCES STRESS

A **PREPARE** to learn how BaseDI is designed to help us structure and organize our thoughts

B **REMEMBER** that BaseDI wants to REDUCE CHAOS in how we get things done

C **LEARN**

How to stay ORGANIZED

BaseDI has 9 main folders classifications
Classification 0 – The Script • Used to host various abstract classes, arguments, enumerations, extensions, interfaces and templates.
Classification 1 – The Storyline • Used to host a CENTRALIZED business logic request routing system.
Classification 2 – The Character • Used to host various code that represent people, animals and other things we assign LIFE to.
Classification 3 – The Setting • Used to host various code that represent various layouts and elements for locations like a blog or address.
Classification 4 – The Experience • Used to host various code about THINGS, SHARED ui and business logic. (DUMB LOGIC/SCREENS)
Classification 5 – The Chapter • Used to host THE STEPS of PROPRIETARY business logic.
Classification 6 – The State • Used to host various types of repositories.
Classification 7 – The Director • Used to host various ENTRY POINTS into executing business logic.
Classification 8 – The Testing Area • Used to "QUALITY" test our work

BaseDI enforces source code folder and file naming standards

```
Local Disk (C:)
Programming
  BaseDI.SourceCode.FreeVersion
    0. Script
    1. Storyline        Core
    2. Character        Who, What
    3. Setting          Where
    4. Experience       How, Why, When (EMOTIONAL)
    5. Chapter          How, Why, When (LOGICAL)
    6. State            Models
    7. Director         Controllers
    8. Testing          Quality Assurance
    9. Organizing       (Used for times when you want to just save something random)
```

How, Why, When (LOGICAL)

Step 1-B

HOW TO KNOW THE BASEDI PERSPECTIVE

A **PREPARE** to learn how BaseDI sees problem solving.

B **REMEMBER** that BaseDI wants you to ask yourself **WHAT** is **YOUR** ultimate end goal

C **LEARN**

What is a STORYLINE?

BaseDI sees things like writing a storyline

Concept 1 – The Plot (Goal Vision)

- The plot is all about the concept of "I HAVE AN IDEA".

  The plot would be a MOVIE SCRIPT

Concept 2 – The Drama

- The drama are the obstacles that are in the way of making our idea a success.

  The obstacles would be the DRAMA of the movie.

Concept 3 – The Happy Ending

- The happy ending is our way of removing an obstacle that's in our way.

  This is basically a solution.

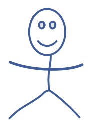
  Another word for solution is "feature".

  The solution would be the HAPPY ENDING of the movie

BaseDI has 12 main plots (END GOALS)

- Goal Number 1: To Generate Brand Awareness (Advertising)
- Goal Number 2: To Generate Brand Trust (Friendship)
- Goal Number 3: To Generate Optin (List Building)
- Goal Number 4: To Sell Low Ticket Offer (Sales)
- Goal Number 5: To Sell High Ticket Offer (Sales)
- Goal Number 6: To Sell Subscription Offer (Sales)
- Goal Number 7: To Sell Commission Offer (Sales)
- Goal Number 8: To Account Gain or Loss (Accounting)
- Goal Number 9: To Improve Customer Experience (Customer Service)
- Goal Number 10: To Perform a Manual Task (Management)
- Goal Number 11: To Automate a Manual Task (Programming)
- Goal Number 12: Other

Example - Goal 3: To Generate Optin (List Building)

A FITNESS STORYLINE	THE PLOT (END GOAL)	THE DRAMA	THE HAPPY ENDING
 Nutrition Seller	I want to sell my supplement products	I need a way to COLLECT LEADS in EMAIL FORMAT	I will implement an online lead capture form.

Step 1-C

HOW TO DESIGN IDEA STORYLINES

A **PREPARE** to learn how BaseDI uses a <u>CENTRALIZED REQUEST</u> system to <u>EXECUTE</u> an idea

B **REMEMBER** that BaseDI wants to make things predictable

C **LEARN**

How to design STORIES

BaseDI enforces file naming rules

<u>Rule 1 – Find the MasterLeader file</u>

- Search for "MasterLeader" under 1. Storyline folder

<u>Rule 2 – Find the Correct Region</u>

- Open the <u>6. Action Implementation Region</u> file of "MasterLeader" file

<u>Rule 3 – IF NOT already added</u>

- Follow the existing pattern to ADD a new NICHE MASTER

<u>Rule 4 – IF ADDING new NICHE MASTER</u>

- Implement base class aClass_Programming_ScriptNicheMaster_G_N_V_V

<u>Rule 5 – Save the niche master</u>

- We must use the following naming rules for saving a new niche master.
  - This new class will be saved under the correct <u>SUB FOLDER</u> located in the <u>1. Storyline Folder</u>
  - The name of the new class will follow the pattern of.
    - {Ezines.com Niche} +
    - FactoryImplementer_ +
    - NicheMaster_ +
    - Goal Category Number _ +
    - Goal Niche Number _ +
    - X_X = Version Number

EXAMPLE FILE NAME BELOW

AdvertisingFactoryImplementer_NicheMaster_1_1_1_0

BaseDI uses a CENTRALIZED REQUEST system to execute certain scenes (IDEAS) of a storyline

JSON REQUEST

{"key":"value"} → PLAY SCENE IN MOVIE

1 **1. Call The Request Inspector**

ProgrammingStudioAdministrator_MasterLeader_G_N_V_V.cs

SetupStoryline(jsonObject1, jsonObject2, extraDataObject, request = "")

2 **2. Call The Request Handler Creator**

[Niche]FactoryImplementer_NicheMaster_G_N_V_V.cs

Action(requestObject, jsonObject1, jsonObject2, request = "")

3 **3A. Call The Director Request Handler Type**

Director_Of_[Niche]_Chapter_G_N_Page_P_CRUDValue-OneWordDesc_V_V.cs

Action()

OR

3A. Call The Experience Request Handler Type

Experience_The_[Group]_[SmallDescription]_[SubGroup]_G_N_V_V.cs

Action()

Step 1-D

HOW TO DESIGN STORYLINE SCRIPTS

A **PREPARE** to learn how BaseDI wants us to design various SCRIPTS for our ideas

B **REMEMBER** to know that BaseDI wants us to code guidelines and standards

D **LEARN**

How to design SCRIPTS

Scripts are used to create templates and standards

Rule 1 – Pick Script Type

- Abstract Class
- Enumerations
- Extensions
- Interfaces
- Parameters

Rule 2 – Save the new script

- We must use the following naming rules for saving a new script.
  - This new script will be saved under the correct SUB FOLDER located in the 0. Script Folder
  - The name of the new script will follow the pattern of.
    - IF Abstract Class
      - aClass_{Ezines.com Niche}_ +
    - IF Enumeration
      - eEnumerations_{Ezines.com Niche}_ +
    - IF Extension Methods
      - eMethods_{Ezines.com Niche}_ +
    - IF Interfaces
      - iContract_{Ezines.com Niche}_ +
    - IF Parameter
      - aParameter_{Ezines.com Niche}_ +
    - SmallDescription_ +
    - G_N_V_V

EXAMPLE FILE NAME BELOW

eEnumerations_Programming_MasterLeader_12_2_1_0

C **KNOW** that the information below will make more sense as you read the playbook

NOTE: ALL 3rd Party Components are to be ENCAPSUALTED into an "EXTENSION" method

Method Naming Formula 1 to (Try Something) for Extension, Interface or Base Class

- Extensions or Interface or Inherited_ +
- Try_ +
- CRUDValue (Create | Read | Update or Delete)_ +
- SomeDescription_ + (OPTIONAL SomeSmallDescription2_) + V_V

EXAMPLE: Extension_Try_Read_RepositoryType_1_0

Method Naming Formula 2 to (Work with XML, HTML or JSON) for Extension, Interface or Base Class

- Extension or Interface or Inherited_ +
- CRUDValue (Create | Read | Update or Delete)_ +
- DataValue (XML | HTML or JSON) _ +
- IF Single Item
  - Node
- IF List
  - Nodes | Nodes_In_List or Nodes_List
- + _SomeSmallDescription_ + (OPTIONAL SomeSmallDescription2_) + V_V

EXAMPLE: Interface_Read_JSON_Node_1_0

EXAMPLE: Extension_Create_HTML_Nodes_List_1_0

Method Naming Formula 3 to (Work with Chapters and Experiences) (DEFAULT STANDARD)

- Step_GN_SN_Custom_ +
- CRUDValue (Create | Read | Update or Delete)_ +
- (Subcategory + Character) OR (Category + Setting) OR (Category + Subcategory + Experience) OR (ChapterPage + _G_N_Page_P) _
- SomeSmallDescription_ +
- SomeSmallDescription2_ + (OPTIONAL SomeSmallDescription3_) + V_V

EXAMPLE: Step_1_0_Read_BlogSetting_ArticleByID_1_0

EXAMPLE: Step_1_0_Update_ExperienceDataTransferMovement_ToFacebook_1_0

Method Naming Formula 4 to (Work with Chapters and Experiences) (YOUR STANDARD)

- Step_GN_SN_Custom_ +
- SomeSmallDescription1_ +
- SomeSmallDescription2_ + (OPTIONAL SomeSmallDescription3_) + V_V

EXAMPLE: Step_1_0_Custom_Find_JSONPlaceHolderNodes_1_0

NOTES

- GN = "Group Number"
- SN = "Step Number"
- (Notice that we used the word FIND instead of one of the CRUD names)

Step 1-E

HOW TO UNDERSTAND THE BASDI REQUEST SYSTEM

1. The Master Leader (Request Insepctor)

Centralized Request Inspection System (Request Entry Point)

USE CASES

- To INSPECT & ROUTE ALL web service or unit test REQUEST to a NICHE MASTER
- To centralize "logging" and "exception management".
- To centralized "configuration" settings.
- To handle backups and recoveries.

INSPIRED BY

- Kubernetes Master Node
    - SEE: https://kubernetes.io/docs/concepts/#kubernetes-control-plane

2. The Niche Master (Request Handler Creator)

Centralized Request Handler Creation System

USE CASES

- To ACCEPT ALL web service or unit test REQUEST
- To CREATE a DIRECTOR or EXPERIENCE LOGIC WORKER)

INSPIRED BY

- BaseDI Founders
- Ezines.com

3. The Logic Worker (Director or Experience) (Request Handler)

Entry Point to Handle Logic

USE CASES

- To HANDLE & PROCESS ALL web service or unit test REQUEST for a certain NICHE MASTER
- To RESPOND to all request

INSPIRED BY

- Kubernetes Worker Node
    - SEE: https://kubernetes.io/docs/concepts/architecture/nodes/

Character Development

GOAL – To learn how to setup the **PEOPLE** who are involved in an idea

Step 2-A

HOW TO KNOW WHO IS OUR IDEA ABOUT

A **PREPARE** to learn how BaseDI gets us to think about WHO our idea is about

B **REMEMBER** to ask yourself WHO are we solving the problem for in terms of ROLES

C **LEARN**

Who is our idea ABOUT?

BaseDI has 6 CHARACTER role classifications

Character 1 – The Buyer – Target Market
- The Buyer can be an EXTERNAL person such as a person buying an item off of Amazon.
- The Buyer can be an INTERNAL person such as an employee who works in another department.
- The Buyer is the target "AUDIENCE" of our story.

Character 2 – The Seller – Idea Owner
- The Seller is an "ACTOR" acting in our story.

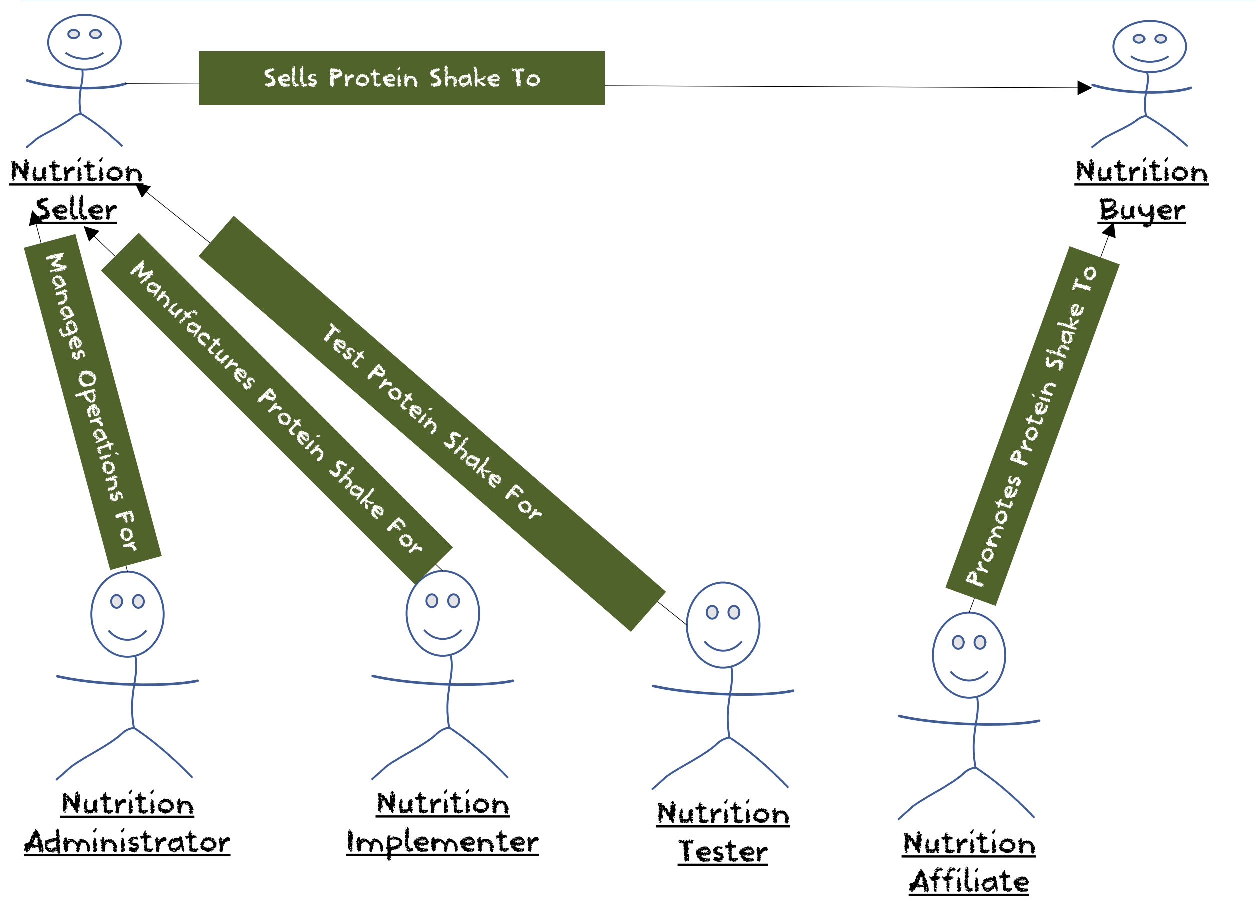
Character 3 – The Implementer – Idea Builder
- The Implementer is an "ACTOR" acting in our story.

Character 4 – The Tester – Idea Qualifier
- The Tester is an "ACTOR" acting in our story.

Character 5 – The Administrator – Task Manager
- The Administrator is an "ACTOR" acting in our story.

Character 6 – The Affiliate – Idea Promoter
- The Affiliate is an "ADVOCATE" and "PROMOTER" in our story.

BaseDI gives us 6 CHARACTER classifications to work with



Step 2-B

HOW TO DESIGN CHARACTERS FOR OUR IDEA

A **PREPARE** to learn where BaseDI wants us to save code that focuses on WHO

B **REMEMBER** to ask yourself WHO are we solving the problem for in terms of ROLES

C **LEARN**

How to design CHARACTERS

Characters are basically ROLES

Rule 1 – Find Abstract Class

- Search for "aClass_Programming_ScriptCharacter" under the 0. Script folder

Rule 2 – Implement Abstract Class

- Implement abstract class aClass_Programming_ScriptCharacter_G_N_V_V

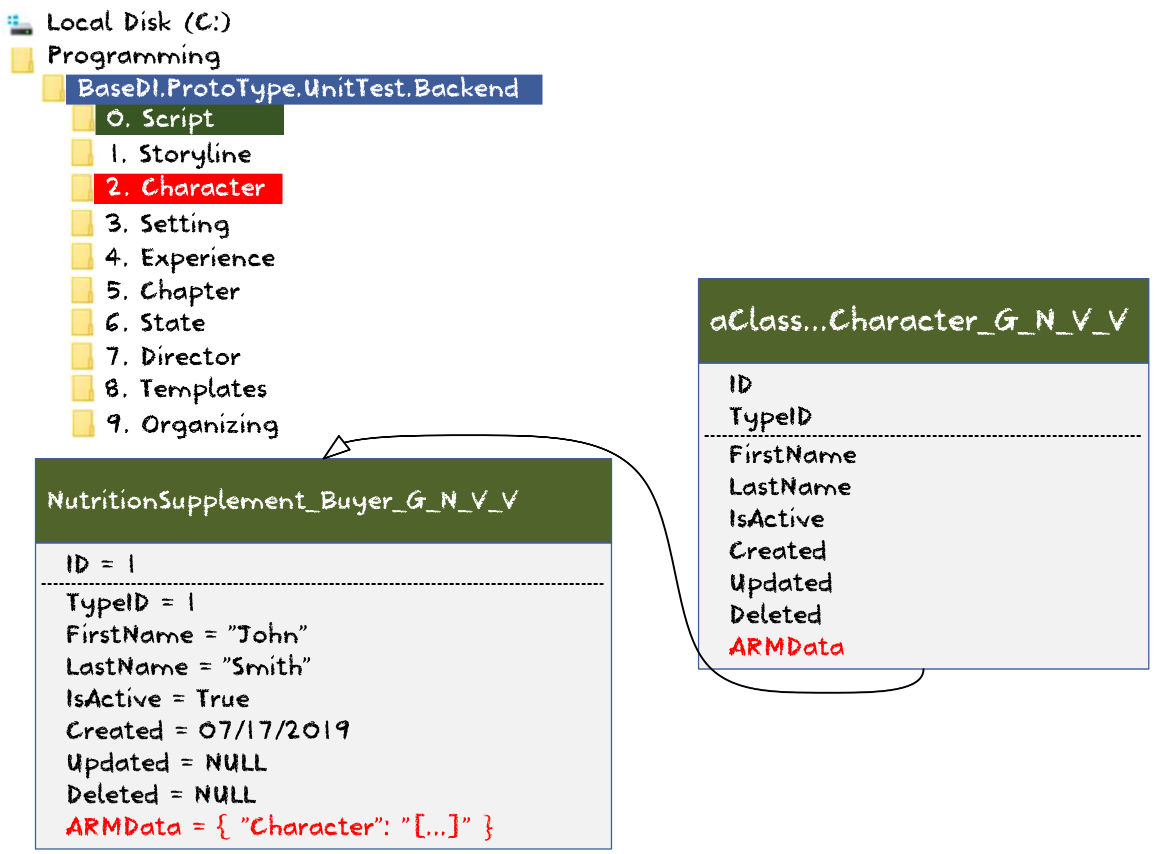
Rule 3 – Save the new Character POCO

- We must follow the following naming rules for creating a new characters.
  - This new charaqcter will be saved under a SUB FOLDER located in the 2. Character Folder
  - The name of the new character will follow the pattern of.
    - {Ezines.com Niche} +
    - {SMALL Description} +
    - _ +
    - {Pick 1 of the 6 Character Groups}
      - Buyer
      - Seller
      - Implementer
      - Tester
      - Administrator
      - Affiliate +
    - G_N_V_V

EXAMPLE FILE NAME BELOW

ProgrammingSystem_Administrator_12_2_1_0

BaseDI enforces source code folder and file naming standards



```
Local Disk (C:)
Programming
  BaseDI.ProtoType.UnitTest.Backend
    0. Script
    1. Storyline
    2. Character
    3. Setting
    4. Experience
    5. Chapter
    6. State
    7. Director
    8. Templates
    9. Organizing
```

aClass...Character_G_N_V_V

- ID
- TypeID
- FirstName
- LastName
- IsActive
- Created
- Updated
- Deleted
- ARMData

NutritionSupplement_Buyer_G_N_V_V

- ID = 1
- TypeID = 1
- FirstName = "John"
- LastName = "Smith"
- IsActive = True
- Created = 07/17/2019
- Updated = NULL
- Deleted = NULL
- ARMData = { "Character": "[...]" }

Setting Development

GOAL – To learn how to setup WHAT time and WHERE an idea takes place

Step 3-A

HOW TO KNOW WHERE THE IDEA TAKES PLACE

A **PREPARE** to learn how BaseDI gets us to think about WHERE someone will experience our idea

B **REMEMBER** to ask yourself WHERE will the idea takes place and deliver an experience

C **LEARN**

Where is idea LOCATED

BaseDI has 9 setting classifications

Setting 1 – Social Media
- Attracts an audience to deliver various forms media via online platforms such as Facebook, YouTube, etc.

Setting 2 – Blog
- Attracts an audience to deliver various forms of media. (NEWS INCLUDED)

Setting 3 – Podcast
- Attracts an audience to deliver media in audio format only. (NEWS INCLUDED)

Setting 4 – Movies TV
- Attracts an audience to deliver media in mostly audio and video format. (NEWS INCLUDED)

Setting 5 – Ecommerce
- Attracts an audience to sell them something in an online medium.

Setting 6 – Home
- Focuses on one of the character types personal home such as an Address

Setting 7 – Commercial Property
- Attracts an audience to a physical place such as shopping mall, retail store or office space.

Setting 8 – Land
- Attracts an audience to a piece of land like a concert, baseball game or some type of event.

Setting 9 – Software
- Focuses on something happening inside a computer program like a service.

BaseDI gives us 9 SETTING classifications to work with

A CHARACTER SETTING RELATIONSHIP EXAMPLE

Nutrition Seller
EcommerceNutrition_Seller

Step 3-B

HOW TO DESIGN SETTINGS FOR OUR IDEA

A **PREPARE** to learn where BaseDI wants us save code that focuses on WHERE

B **REMEMBER** to ask yourself WHERE will someone experience our ideas

C **LEARN**

How to design SETTINGS

Settings are basically ADDRESSES

Rule 1 – Find Abstract Class
- Search for "aClass_Programming_ScriptSetting" under the 0. Script folder

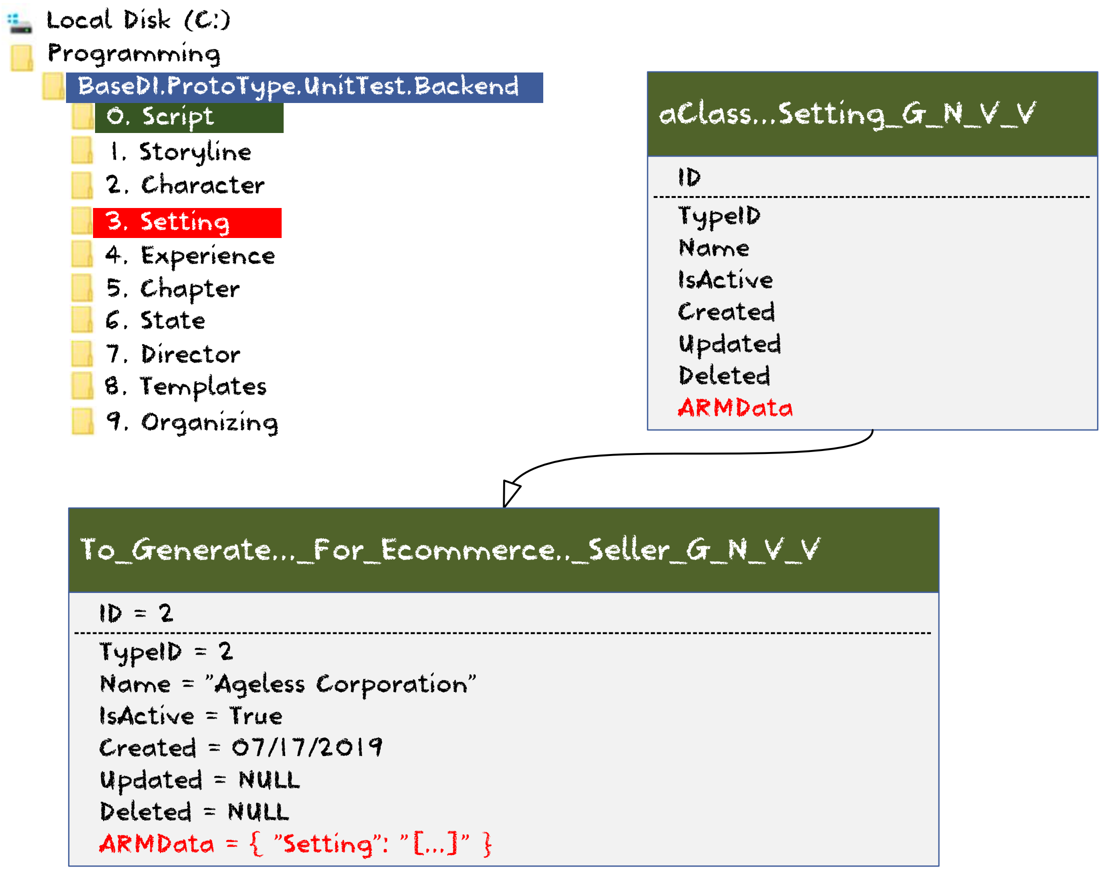
Rule 2 – Implement Abstract Class
- Implement abstract class aClass_Programming_ScriptSetting_G_N_V_V

Rule 3 – Save the new Setting POCO
- We must follow the following naming rules for creating new settings.
  - This new setting will be saved under a SUB FOLDER located in the 3. Setting Folder
  - The name of the new setting will follow the pattern of.
    - {Blog} or
      - {Podcast} or
      - {Movies TV} or
      - {Ecommerce} or
      - {Home} or
      - {Commercial Property} or
      - {Land} or
      - {Software} +
    - Setting_To + {1 of the 12 End Goals} + _For_ +
    - {Ezines Niche} +
    - _ +
    - {Pick 1 of the 6 Character Groups}
      - Buyer
      - Seller
      - Implementer
      - Tester
      - Administrator
      - Affiliate +
    - G_N_V_V

EXAMPLE FILE NAME BELOW

EcommerceSetting_To_GenerateBrandTrust_For_Nutrition_Seller_4_2_1_0

BaseDI enforces source code folder and file naming standards



Local Disk (C:)
- Programming
  - BaseDI.ProtoType.UnitTest.Backend
    - 0. Script
    - 1. Storyline
    - 2. Character
    - 3. Setting
    - 4. Experience
    - 5. Chapter
    - 6. State
    - 7. Director
    - 8. Templates
    - 9. Organizing

aClass...Setting_G_N_V_V

ID
TypeID
Name
IsActive
Created
Updated
Deleted
ARMData

To_Generate..._For_Ecommerce.._Seller_G_N_V_V

ID = 2
TypeID = 2
Name = "Ageless Corporation"
IsActive = True
Created = 07/17/2019
Updated = NULL
Deleted = NULL
ARMData = { "Setting": "[...]" }

Step 3-C

HOW TO DESIGN BASEDI SETTING LAYOUTS

A **PREPARE** to learn how BaseDI also has a LAYOUT system

B **REMEMBER** to ask yourself HOW will we layout various settings

C **LEARN**

What are layout ZONES?

Zones are basically shopping aisles

Example 1 – The Shopping Mall

- Let's say you walk into a clothing store in a shopping mall.

  The store can have 3 ZONES of Customer Service, Product Catalog and Checkout

Example 2 – The Website

- Let's say you went to Amazon.com

  The website can also have 3 ZONES of Customer Service, Product Catalog and Checkout

Designing a physical building

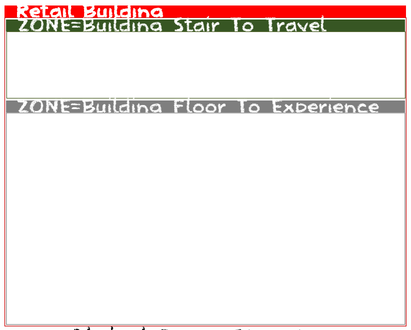
1. The foundation

2. The Stairs

3. The Floors

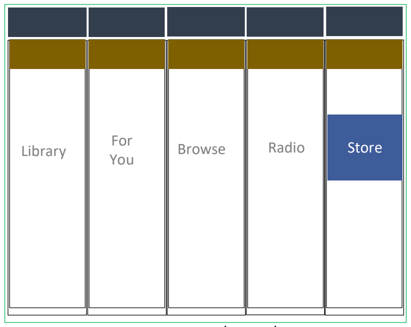
Designing a website or application

1. The foundation


Retail Building
ZONE=Building Stair To Travel
ZONE=Building Floor To Experience

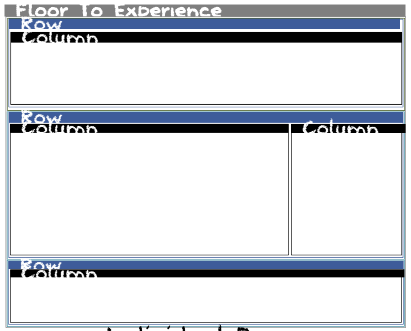
Global Page Structure

2. The Stairs



Page Navigation

3. The Floors



Individual Pages
(Home, About Us, Contact Us)

Experience Development

GOAL – To learn how to setup WHAT EXPERIENCES bring an idea to life

Step 4-A

HOW TO KNOW HOW TO BRING AN IDEA TO LIFE

A **PREPARE** to learn how BaseDI gets us to think about **HOW** to make the idea **REAL**

B **REMEMBER** to always know that our goals is to make this a **HUMAN EXPERIENCE**

C **LEARN**

What makes idea REAL?

BaseDI has 8 human experience classifications

Experience 1 – Touch
- Something that focuses of getting a person to touch something.
  - Call to Action/Data Entry/Gesture/Operate

Experience 2 - Movement
- Something that focuses of getting a person or something to flow and move.
  - Data Transfer/Transformation/Transition
  - Gesture/Content
  - Visual Animation/Transformation/Transition

Experience 3 - Smell
- Something that focuses of provoking the smell emotion in a person.
  - Content/Demo/Gesture

Experience 4 - Taste
- Something that focuses of provoking the taste emotion in a person.
  - Content/Demo/Gesture

Experience 5 - Sight
- Something that focuses of provoking the visual emotion in a person.
  - Content/Gesture

Experience 6 - Hear
- Something that focuses of provoking the listening emotion in a person.
  - Content/Gesture/Message

Experience 7 - Awareness
- Something that focuses of getting a person to become aware of something.
  - Advertisement/Status/Self/Content/Offer

Experience 8 - Classification
- Something that focuses of helping a person organize and make sense of their experience.
  - Group/Type/Behavior/Observation/State

NOTE: SEE LESSON 6 OF HOW TO DESCRIBE AN EXPERIENCE

Step 4-B

HOW TO DESIGN SHARED IDEA EXPERIENCES

A **PREPARE** to learn how BaseDI wants us to design various experiences for our idea.

B **REMEMBER** know that BaseDI always considers the **HUMAN EXPERIENCE**

C **LEARN**

How to design EXPERIENCES

Experiences can be visual and/or functional

Rule 1 – Find Abstract Class

- Search for "aClass_Programming_ ScriptExperience" under the 0. Script folder

Rule 2 – Implement Abstract Class

- Implement abstract class aClass_Programming_ScriptExperience_C_N_V_V

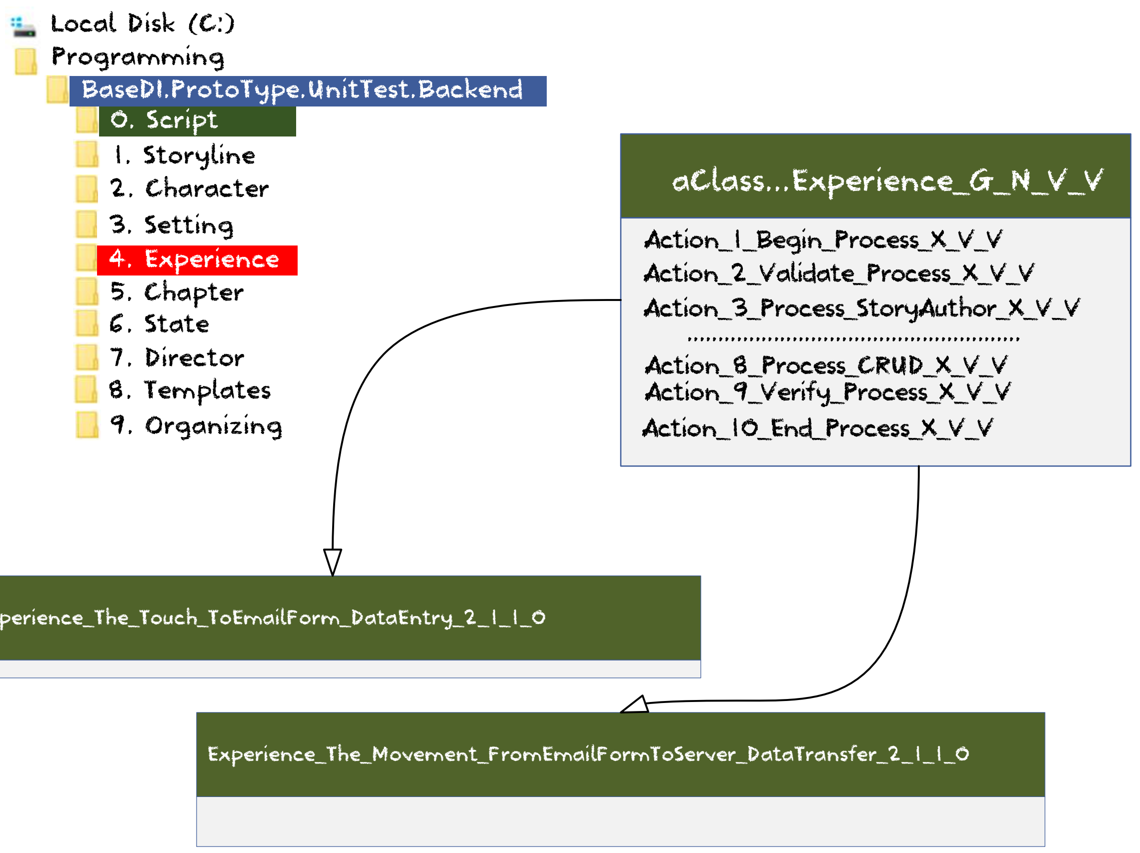
Rule 3 – Save the new shared EXPERIENCE

- We must follow the following naming rules for creating new experiences.
  - This new experience will be saved under a SUB FOLDER located in the 4. Experience Folder
  - The name of the new class will follow the pattern of.
    - Experience_The _ +
    - {Touch} or
      - {Movement} or
      - {Smell} or
      - {Taste} or
      - {Sight} or
      - {Hear} or
      - {Awareness} or
      - {Classification} +
  - _ {SMALL Description} +
  - _ +
  - {Pick 1 of the Sub Groups from STEP 4-A} +
  - _ +
  - G_N_V_V

EXAMPLE FILE NAME BELOW

Experience_The_Movement_FromEmailFormToServer_DataTransfer_2_1_1_0

BaseDI enforces source code folder and file naming standards


Local Disk (C:)
Programming
BaseDI.ProtoType.UnitTest.Backend
0. Script
1. Storyline
2. Character
3. Setting
4. Experience
5. Chapter
6. State
7. Director
8. Templates
9. Organizing
aClass...Experience_G_N_V_V
Action_1_Begin_Process_X_V_V
Action_2_Validate_Process_X_V_V
Action_3_Process_StoryAuthor_X_V_V
....
Action_8_Process_CRUD_X_V_V
Action_9_Verify_Process_X_V_V
Action_10_End_Process_X_V_V
Experience_The_Touch_ToEmailForm_DataEntry_2_1_1_0
Experience_The_Movement_FromEmailFormToServer_DataTransfer_2_1_1_0

Step 4-C

HOW TO SEE EXAMPLES OF EXPERIENCES

1A. Touch – Call to Action

Click here for 50% Off

1B. Touch – Data Entry

Please enter in email address

Submit

1C. Touch – Gesture

Swipe Left for No

Swipe Right for Yes

Do you like this beer?

Zombie Dust

2A. Movement – Data Transfer 1 of 2

Move Money to A Friend via Stripe

Step 1: Please submit payment amount.

$100.00

Submit

A vendor fee of $12.50 has been added
Your checkout cost is $112.50

Step 2: Stripe Service Takes Information

Call API Service

Step 3: Stripe Service Takes Transaction Fee

Transaction Fee = 12.5% = $12.50

$112.50 - $12.50 = $100.00

Step 4: Stripe Service Moves Money to Friend's Stripe Account

$100.00

Step 5: Stripe Service Sends Success or Failure Response Back

2B. Movement – Data Transfer 2 of 2

Move Video From Box to YouTube

API Service Calls

Step 1: Authenticate Into Box.com via API

Step 2: Authenticate Into YouTube.com via API

Step 3: Find Video on Box.com

Step 4: Download Video from Box.com

Step 5: Upload Video from YouTube.com

2C. Movement – Data Transformation

Move Amazon Product Info

Logic

Step 1: Find Amazon Product Detail Page.

https://amzn.to/2zK1vuG

Step 2: Load Page's HTML in Memory

<html>...</html>

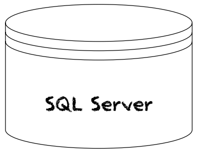
Step 3: Scrape Product Title

Step 4: Scrape Product Name

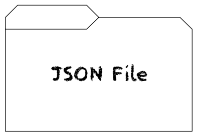
Step 5: Scrape Product Price

Data

Step 6: Move Product Details to Database



Step 7: Move Product Details to File on Box.com



2D. Movement – Data Transition

Move and Manage Website Traffic

Logic

Step 1: Analyze IP Address

10.123.123

Step 2: Decide if IP Address is Whitelisted

If...Then...

A: Yes...Then Allow

B: No...Then Deny

2E. Movement – Gesture

Move because of Database State

Logic

Step 1: Read IF Status is HOT

Step 2: Email image of HOT status to decision makers gesturing to them on what to do.

[IMAGE]

2F. Movement – Visual Animation

Watch Animated Gif

CONTENT OF ANIMATION HERE

3A. Smell – Content

CONTENT TO INVOKE SMELL EMOTION

2G. Movement – Visual Transformation

CONTENT OF TRANSFORMATION HERE

3B. Smell – Demo

AN ACTUAL DEMOSTRATION TO SMELL SOMETHING

2H. Movement – Visual Transition

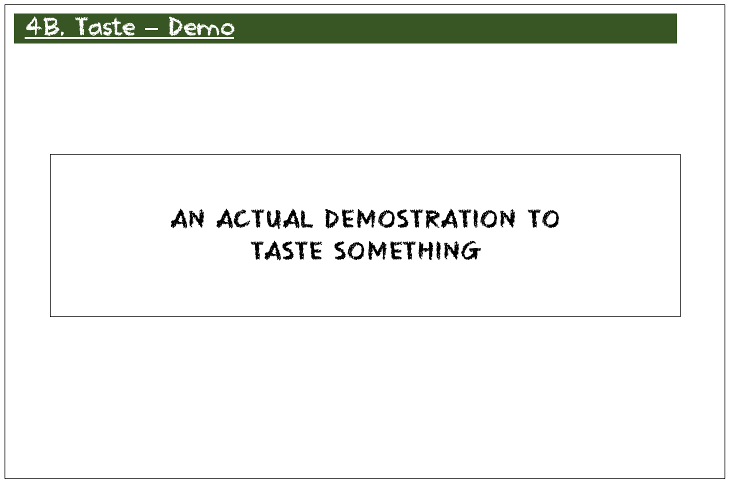
CONTENT OF TRANSITION HERE

3C. Smell – Gesture

ILLUSTRAING SOMETHING HAS A SMELL

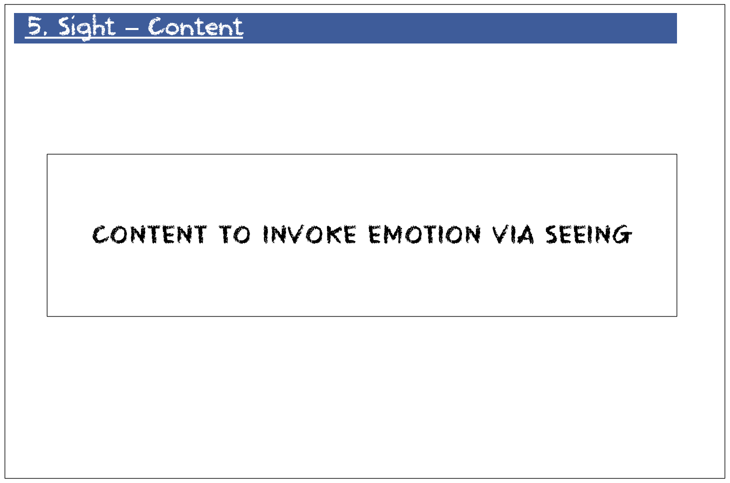
4A. Taste – Content

CONTENT TO INVOKE TASTE EMOTION



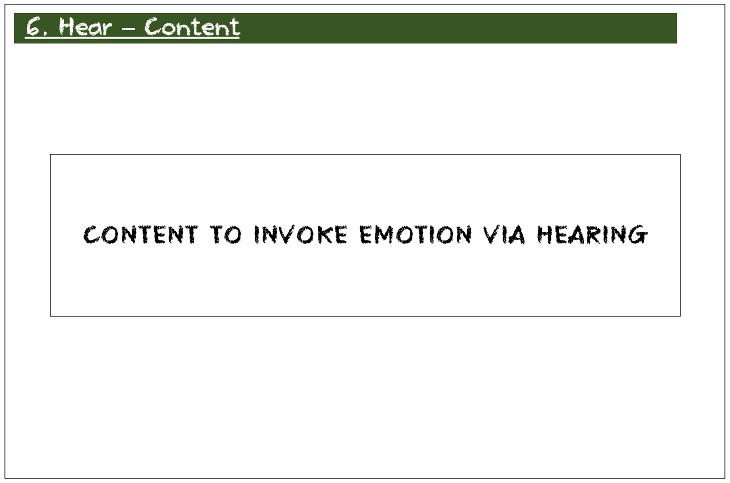
4B. Taste – Demo

AN ACTUAL DEMOSTRATION TO TASTE SOMETHING



5. Sight – Content

CONTENT TO INVOKE EMOTION VIA SEEING



6. Hear – Content

CONTENT TO INVOKE EMOTION VIA HEARING

6. Hear – Message – 1 of 3

Email Message

Logic

Step 1: Pull Email Message From

Step 2: Pull Email Message To

Step 3: Pull Email Message Subject

Step 4: Pull Email Message Body

Step 5: Send Email Message

6. Hear – Message – 2 of 3

Text Message

Logic

Step 1: Pull Text Message From

Step 2: Pull Text Message To

Step 3: Pull Text Message Body

Step 4: Send Text Message

6. Hear – Message – 3 of 3

Voice Message

Logic

Step 1: Pull Voice Message From

Step 2: Pull Voice Message To

Step 3: Pull Voice Message

Step 4: Send Voice Message

7. Awareness – Advertisement – 1 of 2

Digital Ad

Ageless
Sponsored

[FREE 4 MIN CLASS]

Having trouble getting what you want in life?
Sometimes all you need is your own
support group to belong to.

Ageless Group Ad Photo

Learn More

7. Awareness – Advertisement – 2 of 2

Content Headline

How to Win in Life in 3 Easy Steps

7. Awareness – Status – 3 of 3

Email Message Status

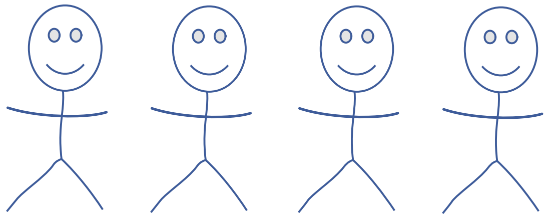
Email Message 1 – Not Read

Email Message 2 – Read

Email Message 3 – Not Read

8. Classification – Group – 1 of 4

Fitness Lovers



8. Classification – Type – 2 of 4

Fitness Workout Belt Type

- Small
- Medium
- Large
- X Large

8. Classification – Behavior – 3 of 4

Fitness Content Delivery Behavior

- Mobile – 50% of the time
- Web – 25% of the time
- Desktop – 25% of the time

8. Classification – Observation – 4 of 4

Fitness Workout Facts

- 50% of women from the Midwest Workout
- 25% of men from the Midwest Workout

Chapter Development

GOAL – To learn how to setup a **UNIQUE** solution (feature) for an idea

Step 5-A

HOW TO ADD WHAT MAKES AN IDEA "UNIQUE"

A **PREPARE** to learn where we add the PROPRIETARY solutions for our idea.

B **REMEMBER** to know that BaseDI takes in account **YOUR UNIQUE** way to solve problems

C **LEARN**

What is a CHAPTER?

BaseDI has 3 concepts that make up a chapter

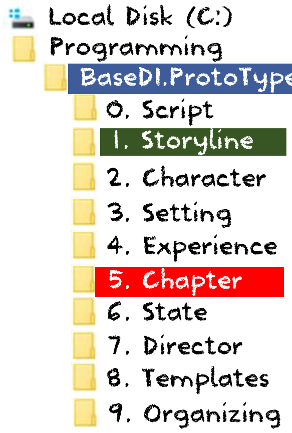
Concept 1 – The Problem
- The problem is a task of work that needs to be completed in order to reach an end goal.

Concept 2 – The Solution
- The solution is **OUR UNIQUE** way of solving a problem.

Concept 3 – The Steps
- BaseDI requires us to break up each solution into a series of 10 steps that make up a Page.

BaseDI enforces source code folder and file naming standards

A CHAPTER WILL NEVER SHARE ANOTHER CHAPTER



TASK ID: Page_1

The Problem: I need a way to collect email addresses

The Solution: To CREATE a DIRECTOR to solve the problem

Example: Director Of ListBuilding Chapter 3 1 Page 1 CreateListForRTG Handler X X.cs

- Action_1_Begin_Process
- Action_2_Validate_Process
- Action_3_Process_StoryAuthor
- Action_4_Process_StoryCharacters
- Action_5_Process_StorySettings
- Action_6_Process_StoryProps
- Action_7_Process_StoryResources
- Action_8_Process_CRUD
- Action_9_Verify_Process
- Action_10_End_Process

THESE ARE THE STANDARD BASEDI METHOD CALLS FOR "ALL" DIRECTORS.

Step 5-B

HOW TO IMPLEMENT AN IDEA IN PIECES

A **PREPARE** to learn how BaseDI wants us to implement **OUR UNIQUE** solution

B **REMEMBER** to know that BaseDI wants us to code in a **PREDICTABLE** way

C **LEARN**

How to design PAGES

Pages are TARGETED steps to a solution

Rule 1 – Find Abstract Class

- Search for "aClass_Programming_ScriptPage" under the 0. Script folder

Rule 2 – Implement Abstract Class

- Implement abstract class aClass_Programming_ScriptPage_G_N_V_V

Rule 3 – Save the new page

- We must follow the following naming rules for creating new pages.
  - This new page will be saved under a SUB FOLDER located in the 5. Chapter Folder
  - The name of the new page will follow the pattern of.

Page_P_1_Begin_Process_G_N_V_V
Page_P_2_Validate_Process_G_N_V_V
Page_P_3_Process_StoryAuthor_G_N_V_V
Page_P_4_Process_StoryCharacters_G_N_V_V
Page_P_5_Process_StorySettings_G_N_V_V
Page_P_6_Process_StoryExperiences_G_N_V_V
Page_P_7_Process_StoryResources_G_N_V_V
Page_P_8_Process_CRUD_G_N_V_V
Page_P_9_Verify_Process_G_N_V_V
Page_P_10_End_Process_G_N_V_V

P = (Project Number)

EXAMPLE FILE NAME BELOW

BaseDI enforces source code folder and file naming standards

```
Local Disk (C:)
Programming
  BaseDI.ProtoType.UnitTest.Backend
    0. Script
    1. Storyline
    2. Character
    3. Setting
    4. Experience
    5. Chapter
    6. State
    7. Director
    8. Templates
    9. Organizing
```

aClass..Page_G_N_V_V

```
IRepository_X_X Repository
JObject StorylineDetails
JObject StorylineDetails_Parameters
ExtraData_G_N_V_V ExtraData
Task<JObject> Action();
```

↓

10 Steps to solve each problem UNIQUELY

Page_P_1_Begin_Process_G_N_V_V
Page_P_2_Validate_Process_G_N_V_V
Page_P_3_Process_StoryAuthor_G_N_V_V
Page_P_4_Process_StoryCharacters_G_N_V_V
Page_P_5_Process_StorySettings_G_N_V_V
Page_P_6_Process_StoryExperiences_G_N_V_V
Page_P_7_Process_StoryResources_G_N_V_V
Page_P_8_Process_CRUD_G_N_V_V
Page_P_9_Verify_Process_G_N_V_V
Page_P_10_End_Process_G_N_V_V

NOTE: THE NEXT LESSON BUILDS ON THIS CONCEPT

State Assignment

GOAL – To setup DATA that describes and instructs what a feature of an idea is suppose to do

Step 6-A

HOW TO DESIGN DATA FOR AN IDEA

A **PREPARE** to learn how BaseDI wants us to design data

B **REMEMBER** that the founders of BaseDI has aligned themselves with Azure ARM Templates

C **LEARN**

What is STATE?

BaseDI has ONE FLEXIBLE data schema

- BaseDI follows an extended version of the Microsoft Azure ARM template.

THE FOUNDERS OF BASEDI WANTED A FLEXIBLE DATA DESIGN STANDARD

- BaseDI uses Microsoft Azure ARM Templates for ALL data schemas.
- BaseDI "EXTENDS" the schema, but follows the core ARM Template Standards

ALL "ARGUMENTS" of a method "WILL ALWAYS" have THESE TWO ARGUMENTS.

An EXAMPLE of a Method would look like the following below

-
  ```
  public void MyWebService(JObject storylineDetails,
                           JOject storylineDetails_Parameters)
  ```

storylineDetails_Parameters = Client Information

RESOURCES TO CHECK OUT

BaseDI ARM Templates – You can see how our templates look. (REQUEST ACCESS)

- https://github.com/NerdyGroupAffiliate/BaseDI.QuickStart.Templates

Microsoft Azure ARM Templates – You can compare our templates to their templates.

- https://github.com/Azure/azure-quickstart-templates

Step 6-B

HOW TO DESIGN DATA FOR AN IDEA I

SOMETHING TO THINK ABOUT

- Think about giving someone step by step instructions to perform an act.
  - When designing a BaseDI ARM Template Schema
  - It is actually having a conversation.

WHAT THE BASEDI ARM TEMPLATE SCHEMA IS REALLY SAYING

I want to define my data as the BaseDI ARM data standard

1

```
{
 "schema": "https://scheme..basedi...deploymentTemplate.json",
```

2

The client will fill in certain parts of the template for us

```
"parameters":
  "baseDI_Facebook_APIPage_MainProfile":    {
  "type": "object",
  "defaultValue": {"baseDIInstructions": { "business": [{"key":"Parameters":"values":
                   [{"value":[{"page_id":"NerdyGuy365"}]}]}]}}
{
```

3

The client must follow the rules in this documentation and predefined resource template

```
["resources":
 {
  "baseDIProfiles":[{
    "baseDI_Facebook_APIPage_MainProfile": "{BASEDIPARAMETERPLACEHOLDER}",
    "baseDI_Facebook_APIPage_DocumentationProfile": [...]
]}
```

CONPEPTS TO BE AWARE OF

- Every resource will consist of a main profile and a documentation profile.
  - The Main Profile allows you to create a predefined template with marked key values of {BASEDIPARAMETERPLACEHOLDER} that will be replaced by the same key in the parameters section above.
  - The Documentation Profile is used to tell us what parameter key/value pairs are possible.
    It also describes what everything means and what can be done.
    This explains the WHY.
  - Profiles follow the naming convention of
    "baseDI_ + WHO_ + WHAT_ + MainProfile "OR" DocumentationProfile"

Step 6-C

HOW TO DESIGN DATA FOR AN IDEA 2

A **PREPARE** to learn how to design client parameters for a BaseDI JSON schema.

B **REMEMBER** OUR DATA SCHEMA IS INSPRIED BY AZURE ARM TEMPLATES

C **LEARN**

How to design PARAMETERS

BaseDI parameters consist of 5 main VALUES sections.

Section 1 – Presentation
- Used for dynamic content creation.

Section 2 – Business
- Used to set TEST vs LIVE mode.
- Used to set the dynamic client input parameters.
- Used to set the name of the Director or Experience to HANDLE the business logic
- Used to set the dynamic conditional logic.

Section 3 – Service
- Used to set the BaseDI NicheMaster name.
- Used to set additional service information.

Section 4 – Security
- Used to set security related information such as Tokens, Username and Encrypted Passwords.

Section 5 – Data
- Used to set one of the "7" BaseDI repository types.
- Used to set one of the "7" BaseDI repository types to handle EXCEPTIONS (MISTAKES).

NOTE: Every parameter schema will have the same schema template

```
{
 "key":"The name of parameter",
 "values":["Array of key value pairs that represent inputs from the client"],
 "type":"object",
 "buzzWords":"Used to mark REQUIRED and also words to describe the purpose of
 the parameter's key",
 "extraKeyValuePairs":["Follows the same pattern as values.
 Usually used for creative purposes to define your own ideas."],
}
```

The 5 Sections to mentioned in POINT C goes inside the "values" array.

Step 6-D

HOW TO PROCESS DATA FOR OUR IDEA

A **PREPARE** to learn more about BaseDI's 10 steps to solve a problem

B **REMEMBER** that BaseDI has 10 methods that are used to solve a problem in a step by step way

C **LEARN**

How to PROCESS DATA?

Solutions are broken up into 10 steps.

- BaseDI uses <u>10 METHODS (STEPS)</u> to formulate a solution to a problem. These methods do not intend to change.

  The reason for this.

  The founders of the BaseDI wanted coders to repeat the same process over and over again.

  This will imprint things into muscle memory.

  Which will increase speed of implementing a solution.

Details about what each step of a solution <u>can be</u> used for

STEP 1. Page P_1_Begin_Process_G_N_V_V

Getting Started

USE CASES

- To convert the JSON BaseDI StorylineDetails object into POCO objects
- To make any service or database calls to APPEND data onto the existing JSON BaseDI StorylineDetails Object

EXAMPLES

<u>PseduoCode: Converting to POCO</u>

```
• JObject rss = JObject.Parse(storylineDetails);
• JArray clientData
              = (JArray)rss["parameters"];
• clientData.SerializeInto(SomePOCOObject)
```

<u>PseduoCode: Appending data to StorylineDetails</u>

```
• JObject rss = JObject.Parse(storylineDetails);
• rss.Write(storylineDetails, "SomeExtraData");
```

<u>PseduoCode: Making a service call</u>

```
• JObject twitter = api.CallTwitter(someJSON);

• JObject rss2 = JObject.Parse(storylineDetails);
• rss.Write(storylineDetails, twitter);
```

STEP 2. Page P_2_Validate_Process_G_N_V_V

Data Validation

USE CASES

- To validate data against business rules. IF FAILED…THROW EXCEPTION

EXAMPLES

<u>Handling Broken Business Rules</u>

```
• If FirstName = ""
    • Call MasterController to Report Mistake
```

STEP 3. Page P_3_Process_StoryAuthor_G_N_V_V

The Owners

USE CASES

- To store a project information used to track the something about an idea. This information can be used to communicate with stakeholders on how an idea is working.

EXAMPLES

Business Team Wants Report of Sales Each Week

- JObject rss = JObject.Parse(storylineDetails);
- JArray clientData
  = (JArray)rss["parameters"];
- string productID = (string)clientData["ID"];
- string orderAmt = (decimal)clientData["Amt"];
- var api = api.CreateOrder(productID,
  orderAmt, Date.Now());
- var api = api.SendSaleReport(productID);

STEP 4. Page P_4_Process_StoryCharacters_G_N_V_V

The People

USE CASES

- To store WHO is the story about?
  - Which (CHARACTER) from Lesson 2

EXAMPLES

Website Visitor Has Submitted Contact Information

- JObject rss = JObject.Parse(storylineDetails);
- JArray clientData
  = (JArray)rss["parameters"];
- string fName = (string)clientData["fName"];
- string email = (string)clientData["Email"];
- var api = api.CreateLead(fName, email);

STEP 5. Page P_5_Process_StorySettings_G_N_V_V

The Location

USE CASES

- To store the location of the story
- To store our RESEARCH goals.

EXAMPLES

- Blogging = <html>...</html>
  - JObject rss = JObject.Parse(storylineDetails);
  - JArray clientData
    = (JArray)rss["parameters"];
  - string productID = (string)clientData["ID"];
  - var api = api.CreateWebpage(productID);
  - return api.HTMLString;

STEP 6. Page P_6_Process_StoryExperiences_G_N_V_V

The Experiences

USE CASES

- To trigger the various aspects that make up the human experience.

EXAMPLES

Sight

- return Experiene.AppendStyleSheets(api.HTMLString);

STEP 7. Page P_7_Process_StoryResources_G_N_V_V

The Hardware & Software Services

USE CASES

- The scale up and down various hardware resources in the cloud.
- To setup calls to various software services.

EXAMPLES

- Virtual Machine = Scale Up Now
- REST Api = Call ScaleUpAzureVM(...);

STEP 8. Page P_8_Process_CRUD_G_N_V_V

The Information

USE CASES

- To Create, Read, Update and/or Delete

STEP 9. Page_P_9_Verify_Process_G_N_V_V

The Quality Assurance Check

USE CASES

- To check the state of our story to see if something is wrong.

EXAMPLES

We just created data in Page_P_8_Process_CRUD ....

- NO ID was returned
  - throw exception("Invalid ID Returned")

STEP 10. Page_P_10_End_Process_G_N_V_V

The One Last Time to Do Something

USE CASES

- To have one final chance to make an impact.

EXAMPLES

- The feature REQUIRED us to PROCESS_CRUD and READ data from a service BEFORE "STEP 8".

- So we added our CRUD logic to the end instead.

Step 6-E

HOW TO CONTROL THE DATA OF AN IDEA

A **PREPARE** to learn how BaseDI wants us to work with data

B **REMEMBER** to ask yourself **WHERE** is the data stored and **HOW** will it be controlled

C **LEARN**

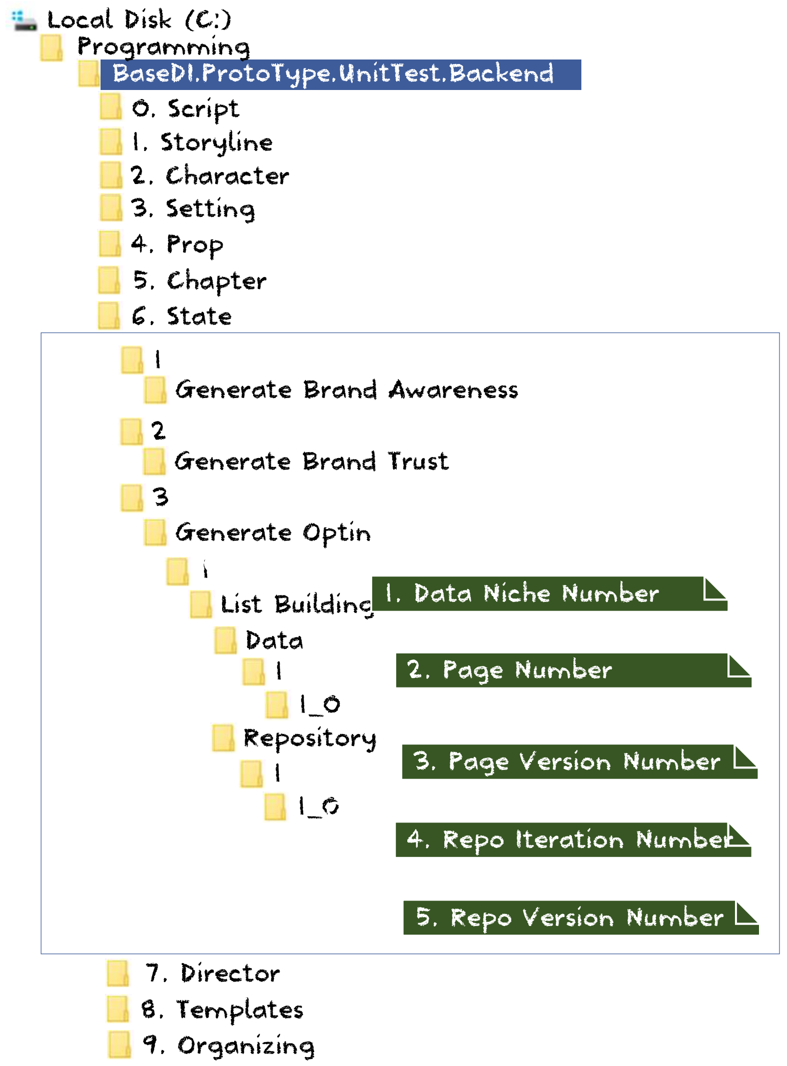
How to locate DATA?

BaseDI has 7 repository types
Repository 1 – Local File • This is perfect for testing features. BaseDI ARM Template data design standard.
Repository 2 – Local Database • This is perfect for testing features. On Premise SQL, Access Database and Excel Spreadsheets are examples.
Repository 3 – Local Service • This is perfect for testing features. This is usually used to see if an API call can be completed on a local computer.
Repository 4 – Remote File • This is perfect for testing and production ready features. An example would be hitting a remote JSON file that is hosted elsewhere.
Repository 5 – Remote Database • This is perfect for testing and production ready features. An example would be hitting a cloud SQL Server Database.
Repository 6 – Remote Service • This is perfect for testing and production ready features. An example would be hitting a Microservice.
Repository 7 – Remote Service Vendor • This is perfect for testing and production ready features. An example would be hitting a 3rd Party's API that you don't own.

CONTINUE TO NEXT PAGE

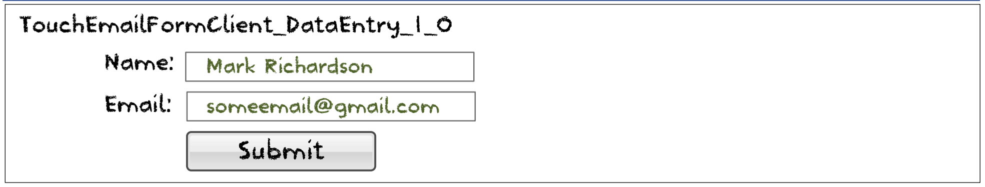
DATA LOCATION EXAMPLE: HOW TO STORE TEST DATA IN A LOCAL FILE

1 Pick Data File Location



Local Disk (C:)
- Programming
  - BaseDI.ProtoType.UnitTest.Backend
    - 0. Script
    - 1. Storyline
    - 2. Character
    - 3. Setting
    - 4. Prop
    - 5. Chapter
    - 6. State
      - 1
        - Generate Brand Awareness
      - 2
        - Generate Brand Trust
      - 3
        - Generate Optin
          - 1
            - List Building
              - Data
                - 1
                  - 1_0
              - Repository
                - 1
                  - 1_0
    - 7. Director
    - 8. Templates
    - 9. Organizing

2 Design Data Wireframe



TouchEmailFormClient_DataEntry_1_0

Name:	Mark Richardson
Email:	someemail@gmail.com

3 Create Test Data Using BaseDI Standards

4 Saved Local Test File Name

LocalFile_Director_Of_Chapter_3_1_Page_1_CreateEmailListForRTG_Handler_1_0_09022019_101730_1_0.json

LocalFile_Experience_The_Movement_ToFacebookPage_DataTransfer_2_3_1_0_09022019_101730.json

Repository DIRECTORS follow the pattern of.

- [REPOSITORY TYPE] +
- _ +
- Director_Of_Chapter_G_N_Page_P + (G_N = Goal Number and Niche Number, P = Project Number)
- _ +
- [CRUDValue] [Niche] + (Create or Read or Update or Delete)(Ezine.com Niche)
- [Preposition] [Noun] +
- {SMALL Description} + (Optional One Extra Word)
- _ +
- {CurrentTime} (10:17pm and 30 seconds is converted to 101730 +
- _ Handler _+
- V_V + (V_V = Version Number)
- .{JSON}

Example Final File Name

LocalFile_Director_Of_ListBuilding_Chapter_3_1_Page_1_CreateEmailListForRTG_Handler_1_0_09022019_101730_1_0.json

Repository EXPERIENCES follow the pattern of.

- [REPOSITORY TYPE] +
- _ +
- Experience_The_{Group} + (Group = See Lesson 4 for the 8 group classifications)
- _ +
- {SMALL Description}_ +
- {SubGroup}_ + (SubGroup = See Lesson 4 for the 8 group sub classifications)
- G_N_V_V_ + (G_N = Goal Number and Niche Number, P = Project Number) (V_V = Version Number)
- {CurrentTime} (10:17pm and 30 seconds is converted to 101730 +
- .{JSON}

Example Final File Name

LocalFile_Experience_The_Movement_ToFacebookPage_DataTransfer_2_3_1_0_09022019_101730.json

Director

GOAL – To setup our remote control that allows us to Play, Stop, Fast Forward and Rewind an Idea

Step 8-A

HOW TO START EXECUTION FOR AN IDEA

A **PREPARE** to learn how BaseDI wants us CREATE ENTRY POINTS into a solution

B **REMEMBER** to always know that BaseDI wants us to START LOGIC in a PREDICTABLE way

C **LEARN**

What is a DIRECTOR?

BaseDI requires us to say ACTION to start a process

After a script is created and the actors know their parts.

A director can say "ACTION" to tell the actors to act out a SCENE of a movie.

In our case...

The director's main job is to EXECUTE ACTION for a feature of an idea.

All Entry Points to logic are called DIRECTORS

A DIRECTOR WILL NEVER SHARE ANOTHER DIRECTOR

- Local Disk (C:)
  - Programming
    - BaseDI.ProtoType.UnitTest.Backend
      - 0. Script
      - 1. Storyline
      - 2. Character
      - 3. Setting
      - 4. Experience
      - 5. Chapter
      - 6. State
      - 7. Director
      - 8. Templates
      - 9. Organizing

PROJECT ID: Page_1

The Problem: I need a way to collect email addresses

The Solution: To CREATE a DIRECTOR to solve the problem

E.g: Director_Of_ListBuilding_Chapter_3_1_Page_1_CreateListForRTG_Handler_X_X.cs

```
internal async Task Action()
```

- Action_1_Begin_Process
- Action_2_Validate_Process
- Action_3_Process_StoryAuthor
- Action_4_Process_StoryCharacters
- Action_5_Process_StorySettings
- Action_6_Process_StoryProps
- Action_7_Process_StoryResources
- Action_8_Process_CRUD
- Action_9_Verify_Process
- Action_10_End_Process

THESE ARE THE STANDARD BASEDI METHOD CALLS FOR "ALL" DIRECTORS.

Step 8-B

HOW TO DESIGN A DIRECTOR OF AN IDEA

A **PREPARE** to learn how BaseDI uses DESIGN PATTERNS to create DIRECTORS

B **REMEMBER** to always know that BaseDI wants us to use DESIGN PATTERNS in our code

C **LEARN**

How to design DIRECTORS

Directors must follow a Gang of 4 Design Pattern

Rule 1 – Find Abstract Class
- Search for "aClass_Programming_ScriptDirector" under 0. Script folder

Rule 2 – Implement Abstract Class
- Implement abstract class aClass_Programming_ScriptDirector_{PatternName}Pattern_G_N_V_V

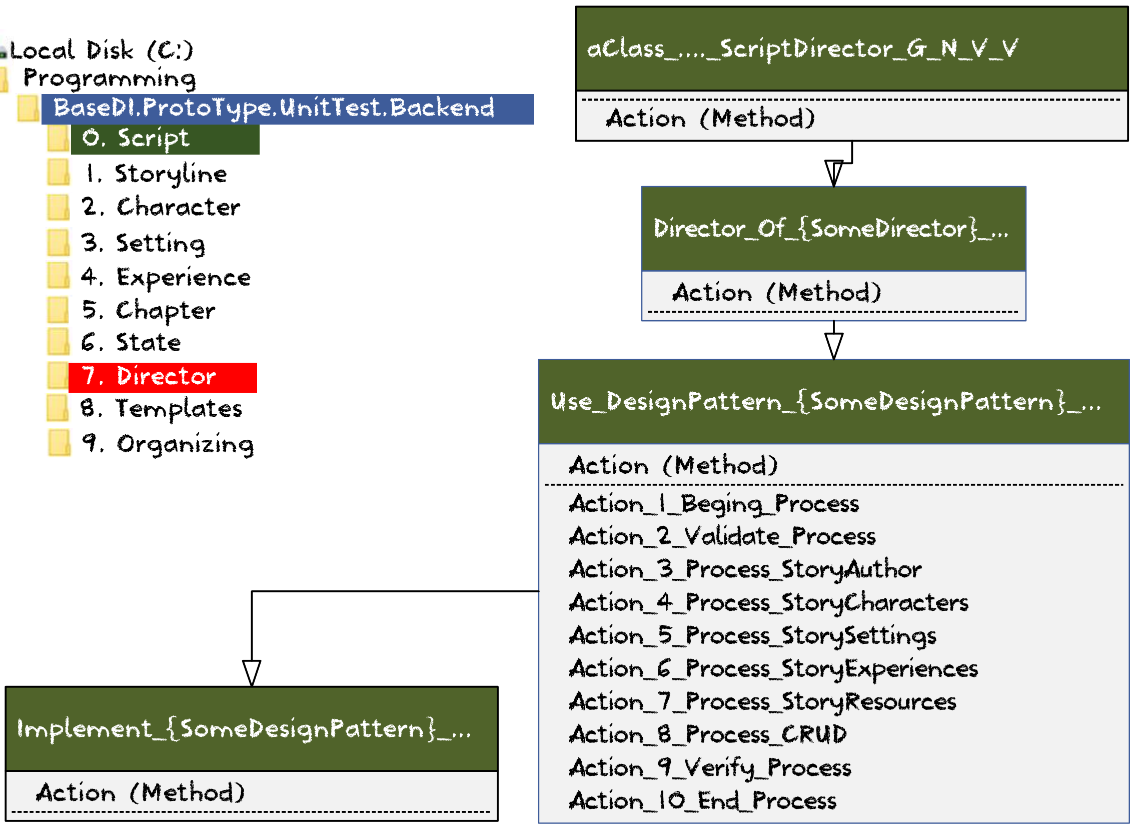
Rule 3 – Save the new DIRECTOR
- We must follow the following naming rules for creating new directors.
  - This new director will be saved under a SUB FOLDER located in the 7. Director Folder
  - The name of the new director will follow the pattern of.
    - Director_Of_{Ezines.com Niche}_Chapter_G_N -
      - G_N = Goal # + Niche #+
    - Page_X +
      - X = Page Number
    - {Create|Read}Update or Delete} +
    - {Small Description} +
    - {English Preposition} +
    - {WHO|WHAT (Example: RTGMastermind} +
    - _EntryPoint_
    - V_V = Version Number

EXAMPLE FILE NAME BELOW

Director_Of_Advertising_Chapter_1_1_Page_1_CreateAdvertisementForAll_Handler_1_0.cs

BaseDI enforces source code folder and file naming standards

- Local Disk (C:)
  - Programming
    - BaseDI.ProtoType.UnitTest.Backend
      - 0. Script
      - 1. Storyline
      - 2. Character
      - 3. Setting
      - 4. Experience
      - 5. Chapter
      - 6. State
      - 7. Director
      - 8. Templates
      - 9. Organizing



aClass_...._ScriptDirector_G_N_V_V
Action (Method)
Director_Of_{SomeDirector}_...
Action (Method)
Use_DesignPattern_{SomeDesignPattern}_...
Action (Method)
Action_1_Beging_Process
Action_2_Validate_Process
Action_3_Process_StoryAuthor
Action_4_Process_StoryCharacters
Action_5_Process_StorySettings
Action_6_Process_StoryExperiences
Action_7_Process_StoryResources
Action_8_Process_CRUD
Action_9_Verify_Process
Action_10_End_Process
Implement_{SomeDesignPattern}_...
Action (Method)

Lesson 9

Architecture

GOAL – To understand how BaseDI is designed from an N Tier Perspective

Step 9-A

HOW TO THE BASEDI ARCHITECTURE LOOKS

A **PREPARE** to learn more about how the inner workerings of BaseDI is designed

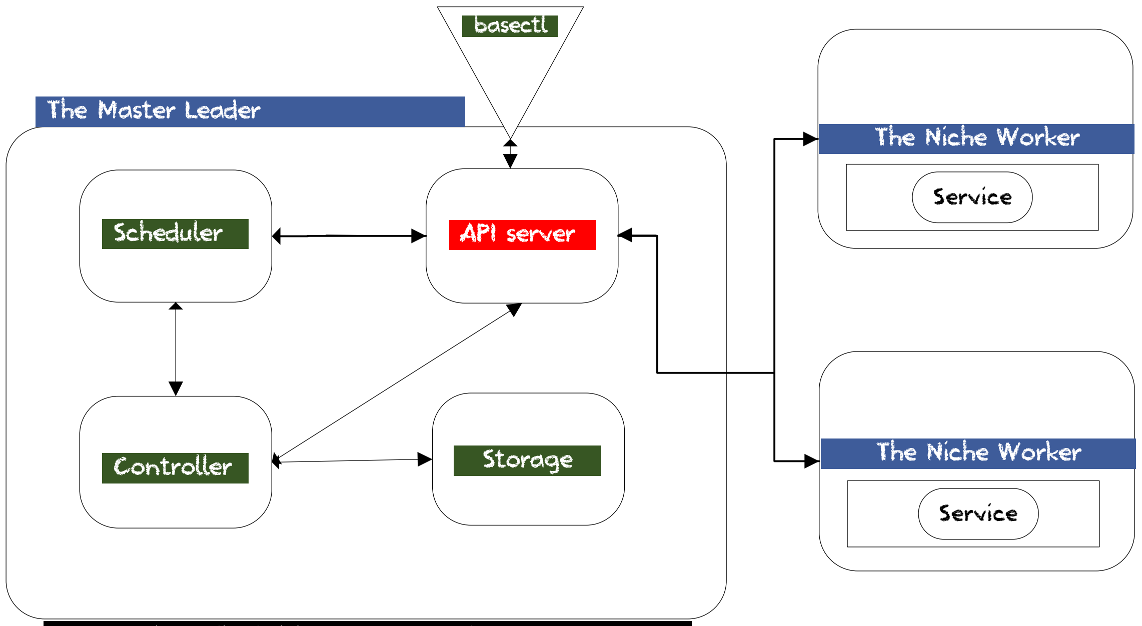
B **REMEMBER** to always know that BaseDI wants to link to successful concepts

C **LEARN**

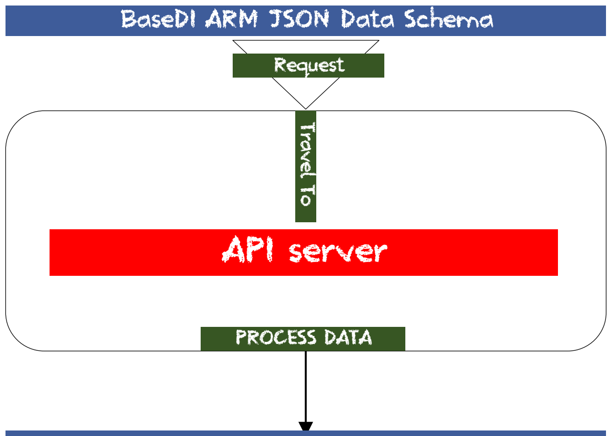
What is the ARCHITECTURE?

Google's Kubernetes Influenced the BaseDI design
• The founders of BaseDI designed to model it's service N Tier design to emulate Google's Kubernetes.
• The basectl clients Base Control Clients consist of various clients to interact and manage BaseDI
• The Master Leader (API Server, Controller & Scheduler) • Serves as a communication router between various microservices. • Serves as a microservice centralized settings and secrets repository manager. • Serves as a centralized exception & logging manager. • Serves as a centralized BACKUP manager.
• The Niche Workers The actual microservices themselves.

The architecture was inspired by Google's Kubernetes



ProgrammingStudioAdministrator_MasterLeader_12_2_1_0
ProgrammingFactoryImplementer_NicheMaster_12_2_1_0



The Master Leader

Scheduler

- How to transport a request to the correct worker resource.

Controller

- How to convert a request to be processed.
- How to sense exceptions and failures in the worker resources.
- How to sense the hardware cost impact of each worker resource.
- How to store backups and restores of data and hardware.

Storage

- How to store configuration and secret data in a centralized place for all worker resources.

Types of Questions Each Worker Resource can Answer (1 of 4)

1. Worker Resource (WR)

Generate Brand Awareness Service

WHO
- How to discover WHO responded to various ads.

  Example: { "IPAddress": "Chicago" }

WHY
- How to prove WHY a market might responded to an ad.

  Example: { "TestAd":"Motivation Ideas" }

WHAT
- How to know predict WHAT channel should we use for a certain topic.

  Example: { "Platform":"Facebook" }

HOW
- How to know HOW can we verify the best way to advertise to a market.

  Example: { "Device":"Mobile" }

2. Worker Resource (WR)

Generate Brand Trust Service

WHO
- How to discover WHO we are building trust with.

  Example: { "Gender":"Women" }

WHY
- How to prove WHY a market might listen to us.

  Example: { "TestAd":"TheyResponsed" }

WHAT
- How to predict WHAT we should talk about to boost our reputation.

  Example: { "Topic":"Motivation Habits" }

HOW
- How to know HOW can we verify a market will response positivity to a topic.

  Example: { "Channel":"Facebook" }

Types of Questions Each Worker Resource can Answer (2 of 4)

3. Worker Resource (WR)

Generate Brand Optin Service

WHO
- How to discover WHO has given us permission to follow up with them.

  Example: { "Joe": "joe@gmail.com" }

WHY
- How to prove WHY a person who want us to contact them again.

  Example: { "SubscribedFor":"PDF" }

WHAT
- How to predict WHAT should we follow up with a person about.

  Example: { "Topic":"Motivation" }

HOW
- How to know HOW can we verify the best method of communication in the future.

  Example: { "ResponseType":"Email" }

4,5,6 and 7. Worker Resource (WR)

Sell Low Ticket Offer (4) Service

Sell High Ticket Offer (5) Service

Sell Subscription Offer (6) Service

Sell Commission Offer (7) Service

WHO
- How to discover WHO is more likely to buy an offer.

  Example: { "Gender":"Women" }

WHY
- How to prove WHY they might buy from us.

  Example: { "SubscribedFor":"PDF" }

WHAT
- How to predict WHAT should we say to remove objections to a sell.

  Example: { "Research":"FAQ" }

HOW
- How to know HOW can we verify that someone really loves something?

  Example: { "Sales":"Checkout" }

8. Worker Resource (WR)

Account Loss or Gain

WHO
- How to discover WHO is making us the most money.

  Example: { "Gender":"Women" }

WHY
- How to prove WHY they like us?

  Example: { "Sales":"1,000,000" }

WHAT
- How to predict WHAT should we spend our budgets on.

  Example: { "Ads":"Motivation Habits" }

HOW
- How to know HOW can we verify that we should re-invested in a strategy.

  Example: { "Poll":"AreYouInterestedIn" }

Types of Questions Each Worker Resource can Answer (3 of 4)

9. Worker Resource (WR)

Improve Customer Service

WHO
- How to discover WHO rates our brand in a positive light.

  Example: { "Gender":"Women" }

WHY
- How to prove WHY someone like us?

  Example: { "Reviews":"Yelp" }

WHAT
- How to predict WHAT should we do to smooth the relationship with unhappy customers.

  Example: { "Survey":"HowToImprove" }

HOW
- How to know HOW we can verify customers like our brand?

  Example: { "Software":"Solutions" }

10. Worker Resource (WR)

Perform Manual Task Service

WHO
- How to discover WHO has work to do in order to help us hit our goals.

  Example: { "DesignAd":"Jane S" }

WHY
- How to prove WHY the work needs to be done.

  Example: { "GoalReason":"Revenue" }

WHAT
- How to predict WHAT the gain vs loss will be for task.

  Example: { "Project":"LaborCost" }

HOW
- How to know HOW we can verify the work will provide gains for us.

  Example: { "Sales":"Projections" }

11. Worker Resource (WR)

Automate Manual Task Service

WHO
- How to discover WHO is under pressure to do a lot of manual work.

  Example: { "DesignAd":"Jane S" }

WHY
- How to prove WHY they might be stressed and under pressure?

  Example: { "TaskLeft":"50" }

WHAT
- How to predict WHAT we should spend our time on to automate first.

  Example: { "TaskList":"10 Sales Task" }

HOW
- How to know HOW we can verify the work will provide gains for us.

  Example: { "Sales":"Projections" }

Types of Questions Each Worker Resource can Answer (4 of 4)

12A. Worker Resource (WR)

Other Service (1 of 2)

WHO
- How to <u>discover</u> WHO else do we need to help.

  Example: { "Tech":"CareerSeekers" }

WHY
- How to <u>prove</u> WHY someone would be interested?

  Example: { "TechSalaries":"100k" }

WHAT
- How to <u>predict</u> WHAT we should do to help someone.

  Example: { "Research":"TopSalaries" }

HOW
- How to know HOW we can <u>verify</u> someone is interested in a job?

  Example: { "Survey":"FAQ" }

12B. Worker Resource (WR)

Other Service (2 of 2)

WHO
- How to <u>discover</u> WHO else do we need to help.

  Example: { "TechService":"SharedCode" }

WHY
- How to <u>prove</u> WHY the work needs to be done.

  Example: { "Code":"DuplicateCode" }

WHAT
- How to <u>predict</u> WHAT code should be make shared or not?

  Example: { "Resource":"MoreThan1" }

HOW
- How to know HOW we can <u>verify</u> that code should be shared or not.

  Example: { "IsExternalAPI":"True" }

Step 0-A

XXXXX

A XXXX XXXX

- XXXX
- XXXX

C LEARN

XXXX

XXXX
• XXXX
• XXXX
• XXXX
• XXXX
• XXXX

B XXXX XXXX

Step 0-A

XXXXX

A XXXX XXXX

B XXXX XXXX

D LEARN

XXXX

XXXX
• XXXX
• XXXX
• XXXX
• XXXX
• XXXX

C XXXX XXXX

Step 0-A

XXXXX

A XXXX xxxx

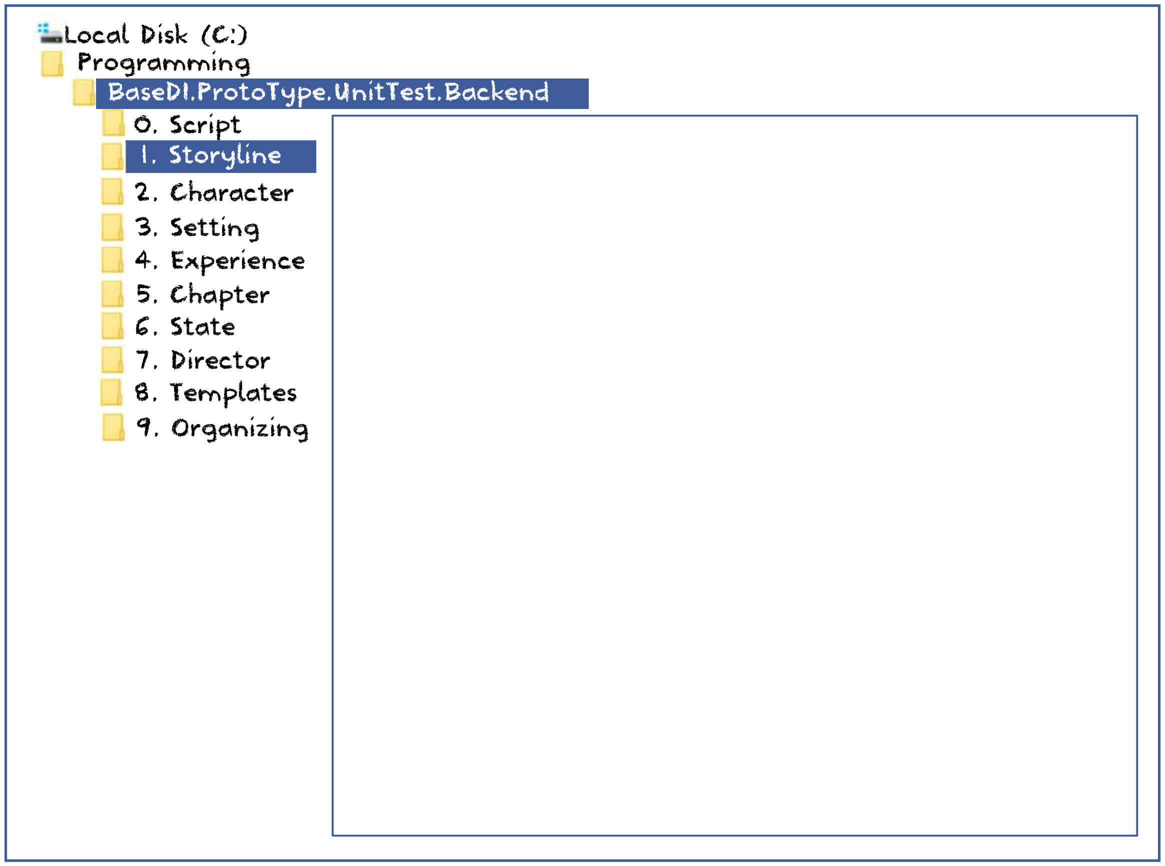
B XXXX xxxx

D LEARN

XXXX

xxxx
• xxxx
• xxxx
• xxxx
• xxxx
• xxxx

C XXXX XXXX



Local Disk (C:)
- Programming
  - BaseDI.ProtoType.UnitTest.Backend
    - 0. Script
    - 1. Storyline
    - 2. Character
    - 3. Setting
    - 4. Experience
    - 5. Chapter
    - 6. State
    - 7. Director
    - 8. Templates
    - 9. Organizing

Step 1-A

XXXX

A XXXX XXXX

B XXXX XXXX

C LEARN

How to stay ORGANIZED

BaseDI organizes everything in 9 concepts

xxxx
- xxxx

xxxx
- xxxx

xxxx
- xxxx

xxxx
- xxxx

xxxx
- xxxx

xxxx
- xxxx

xxxx
- xxxx

xxxx
- xxxx

xxxx
- xxxx

```
Local Disk (C:)
Programming
  BaseDI.ProtoType.UnitTest.Backend
    0. Script
    1. Storyline
    2. Character
    3. Setting
    4. Experience
    5. Chapter
    6. State
    7. Director
    8. Templates
    9. Organizing
```

XXXX

XXXXXX

- Action_1_Begin_Process
- Action_2_Validate_Process
- Action_3_Process_StoryAuthor
- Action_4_Process_StoryCharacters
- Action_5_Process_StorySettings
- Action_6_Process_StoryProps
- Action_7_Process_StoryResources
- Action_8_Process_CRUD
- Action_9_Verify_Process
- Action_10_End_Process

THESE ARE THE STANDARD BASEDI METHOD CALLS FOR "ALL" DIRECTORS.

Step 1-B

XXXX

A XXXX XXXX

B XXXX XXXX

C LEARN

What is a STORYLINE?

BaseDI solves problems like creating a movie

Concept 1 – The Big Picture (Vision)

- The vision is all about the "Thought" of what if.

  The vision would be a MOVIE SCRIPT

Concept 2 – The Problem

- When trying to actually implement the vision.

  A series of many problems will occur.

  The problems would be the DRAMA of the movie

Concept 3 – The Solution

- For each problem we encounter. We must implement a solution.

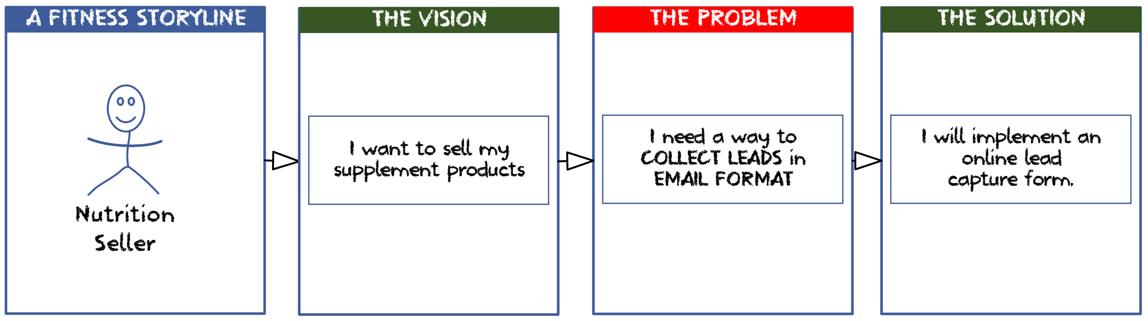
  Another word for solution is "feature".

  The solution would be the HAPPY ENDING of the movie

XXXX

- Goal 1: To Generate Brand Awareness (Advertising)
- Goal 2: To Generate Brand Trust (Friendship)
- Goal 3: To Generate Optin (List Building)
- Goal 4: To Sell Low Ticket Offer (Sales)
- Goal 5: To Sell High Ticket Offer (Sales)
- Goal 6: To Sell Subscription Offer (Sales)
- Goal 7: To Sell Commission Offer (Sales)
- Goal 8: To Account Gain or Loss (Accounting)
- Goal 9: To Improve Customer Experience (Customer Service)
- Goal 10: To Perform a Manual Task (Management)
- Goal 11: To Automate a Manual Task (Programming)
- Goal 12: Other

XXXX


A FITNESS STORYLINE
Nutrition Seller
THE VISION
I want to sell my supplement products
THE PROBLEM
I need a way to COLLECT LEADS in EMAIL FORMAT
THE SOLUTION
I will implement an online lead capture form.

Step 1-A

HOW TO UNDERSTAND THE REQUEST SYSTEM

A **PREPARE** to learn how BaseDI uses CENTRALIZED system to handle REQUEST.

B **REMEMBER** OUR REQUEST SYSTEM IS INSPRIED BY KUBERNETES MASTER NODE

C **LEARN**

How to design STORIES

BaseDI orchestrate out request to workers.

Rule 1 – Open the MasterLeader file

- Search for "MasterLeader" under 1. Storyline folder

Rule 2 – Find the Correct Region

- Open the 5. Action Script file of "MasterLeader" file

Rule 3 – IF NOT already added

- Follow the existing pattern to add a new REQUEST HANDLER "MASTER"

Rule 4 – IF ADDING new handler

- Implement base class aScriptStory_2_1_0

Rule 5 – Save the request handler

- We must follow the following naming rules for creating new files.
  - This new class will be saved under the 1. Storyline Folder
  - The name of the new class will follow the pattern of.
    - {Ezines.com Niche} +
    - FactoryImplementer _ +
    - Master _ +
    - Goal Category Number _ +
    - Goal Niche Number _ +
    - X_X = Version Number

EXAMPLE FILE NAME BELOW

AdvertisingFactoryImplementer_Master_1_1_1_0

Step 1-D

HOW TO UNDERSTAND THE CENTRALIZED SYSTEM