1. Why NIMBUS?

Nimbus Description from Technical docs

Project Vision :

Project Nimbus was formed to create the next generation care management platform for Anthem and its partners and customer, And to leverage Anthem’s current platforms and assets to create new ones based on current and emerging best practices in software engineering and to address current trends in the health care industry.

In addition, this next generation platform aims to create a flexible, expandable product that can grow and change as Anthem and its customer’s needs change while minimizing the need to rewrite large sections of the code and while minimizing the overall cost of change.

Note: Nimbus project is being open sourced and available in Git Hub under Anthem Open Source repository. This can be used by other communities to build their own projects / products faster by avoiding lot of boiler plate code.

Key features of the platform

• Open source - no licensing cost

• Customized Workflow

• Configurable UI

• Multi Tenancy

• Cloud based solution

Nimbus Platform reduces application development time by

• Providing the ability to build application through configuration.

• Providing boilerplate code for cross cutting concerns.

For more details please refer the Nimbus on boarding ppt (in Confluence).

**2. How is it different from regular web app development?**

**Controller:**

Unlike regular MVC, in nimbus HTTP request/response handling is abstracted out into a built-in controller.(WebActionController)

**ex:**

\* Ex 1: /anthem/icr/member/\_search - POST operation which contains criteria <br>

\* Ex 2: /anthem/icr/member/{id} - GET operation which has member id as path variable <br>

\* Ex 3: /anthem/acmp/member/{id}/\_info - GET operation which contains member id as path variable, but also contains an action 'info' to return less data <br>

\* Ex 4: /anthem/acmp/um/case/{cid}/member/address/{aid} - GET operation to return one address from the collection <br>

**please refer:**

<https://github.com/openanthem/nimbus-core/blob/1.2.x/nimbus-core/src/main/java/com/antheminc/oss/nimbus/channel/web/WebActionController.java>

or

**Exercise:**

1)Clone Nimbus Core like from above link, import to STS, 2)open WebActionController class file(short cut to open file,ctrl+shift+r/ctrl+shift+t),, to see http methods developed then open class hirechy( ctrl+shift+h)

**View:**

UI development is simplified through the use of annotations.

These are backed by abstractions of UI stack including HTML, CSS and JS, Bootstrap, and Angular

View comprises of UI components and the view domain which handles view related data.

UI layout in nimbus follows a predetermined nested structure

(Root->Page -> Tile -> Section -> Form/Card/Grid).

Eg. @TextBox, @Button etc

Source reference here

<https://github.com/openanthem/nimbus-core/blob/master/nimbus-core/src/main/java/com/antheminc/oss/nimbus/domain/defn/ViewConfig.java>

Source or application examples here:

Ex: **Button.java**

@Config(url="/pageAddEditGoal/tileEditGoal/sectionEditGoal/goalDetailsForm/\_nav?pageId

=pageCarePlanSummary")

@Button(type = Button.Type.DESTRUCTIVE)

private String delete;

For more information , refer View Config Annotations in the Reference Documentation.

**Model:**

Model classes are referred to as "Core". Data persistence is done through classes configured as ‘Core Domains’

Refer core domain entity in reference document

Source or application examples here.

**@Domain annotation**

Link:

<https://github.com/openanthem/nimbus-core/blob/master/nimbus-core/src/main/java/com/antheminc/oss/nimbus/domain/defn/Domain.java>

**@Model annotation**

Link:

<https://github.com/openanthem/nimbus-core/blob/master/nimbus-core/src/main/java/com/antheminc/oss/nimbus/domain/defn/Model.java>

**3. How do I get started?**

Structure of the application:

Refer the Framework Tech and Product Stack

1)<https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156893721/Training#Training-FrameworkTechStack>

---

**Exercises:**

**Prerequisite:**

1. Please refer Sample\_Training Doc /install setup .

2. Add reference doc links here (pending)

3. Add Git repo instructions(pending)

4. UseFul Links:

1)Create a database named “training” in mongo.

2)open sts create new application as application standard, define port no, db details, basic packages in application.yml file

3)Define an Employee class(with basic variables FirstName, LastName, DateOfBirth,Qualification,..) in any package as application standard.

4)Define Employee Class as a “Core Domain”, using required Config annotations. And mention mongo db as persistence.

5)Define Landing Root(View Root Domain) For that Employee class., using required View Config Annotations.

6)Define that landing Root Domain alias in login controller, so that when we will login reach to the respective landing page.

7)Define the Form in the Default page of Landing Root, with Respect to the fields provided in core domain and design respective view components in the form along with submit button, and give proper mapping to render the data.

8)Using core config annotations , define add functionality when you click on submit button.

9)In the same page Define another section for Grid to Display inserted records.(use view config annotations)

10)Define Base packages in the yml file as per to coding standard in specific location i.e domain.base.package.

11) build ,run application.

12)Test the functionality in console. By logging in with respective login as u mentioned in login controller.

13) read the console /note down if any exceptions.

14) Reverse kt

--