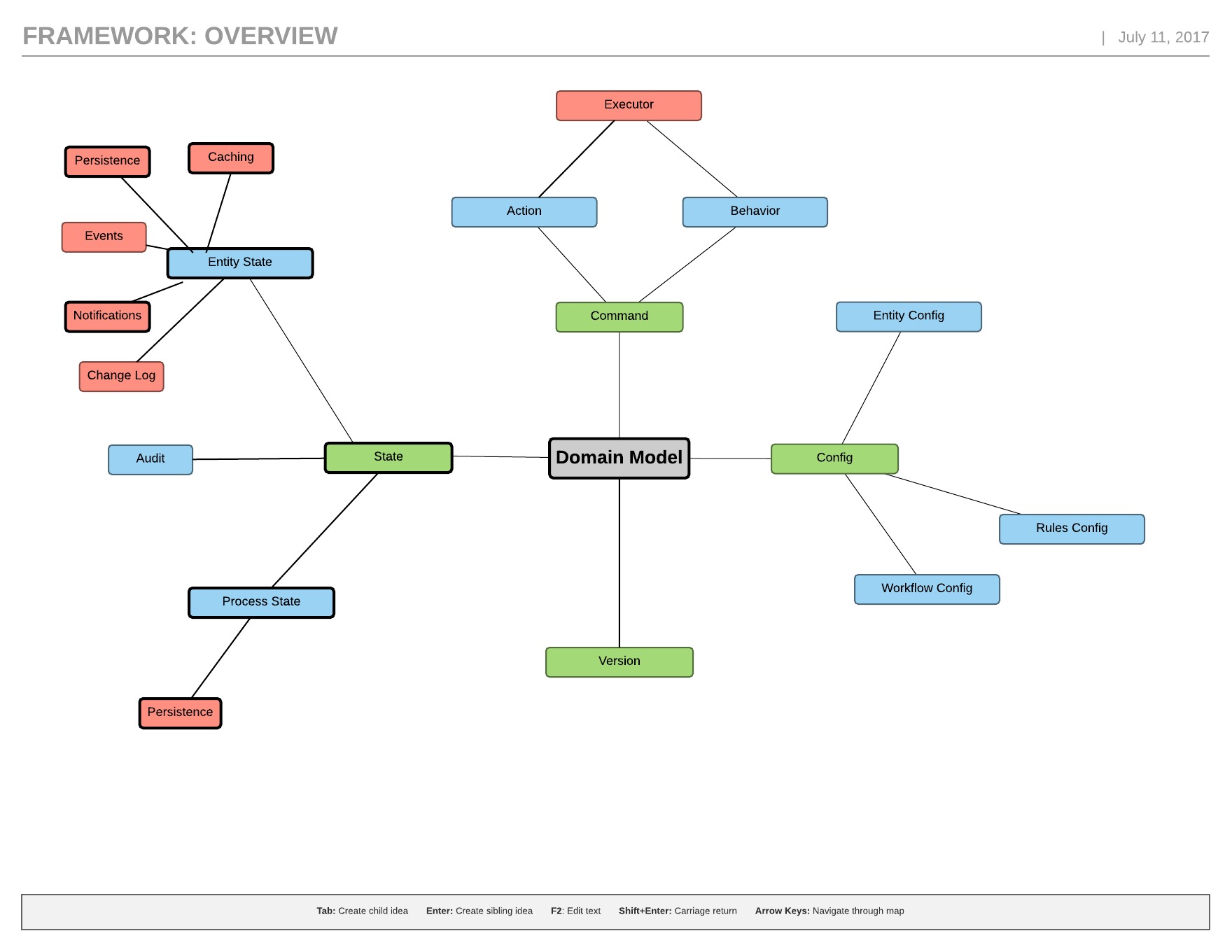
Nimbus

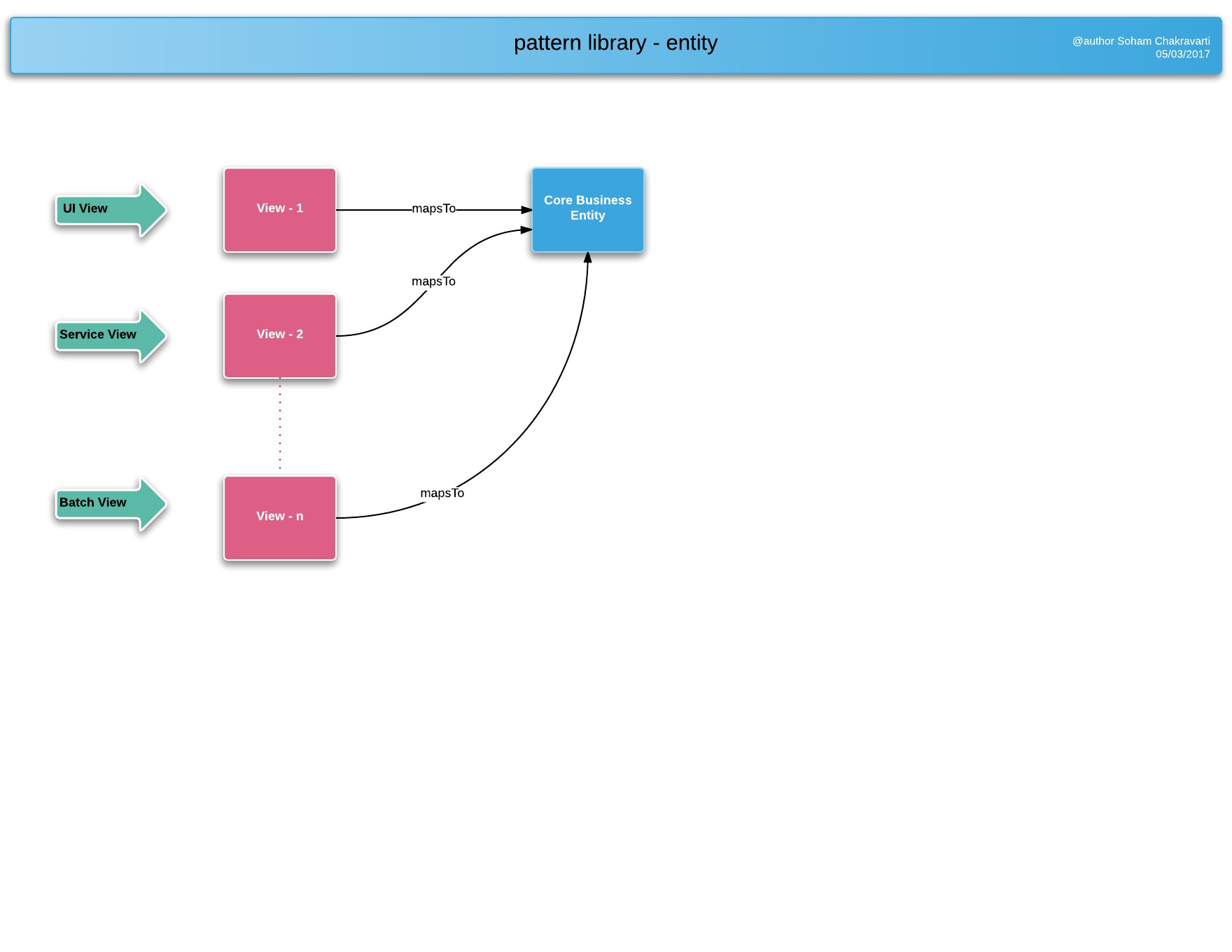
**Domain Model** - For any application we first have to define the business entity/entities.

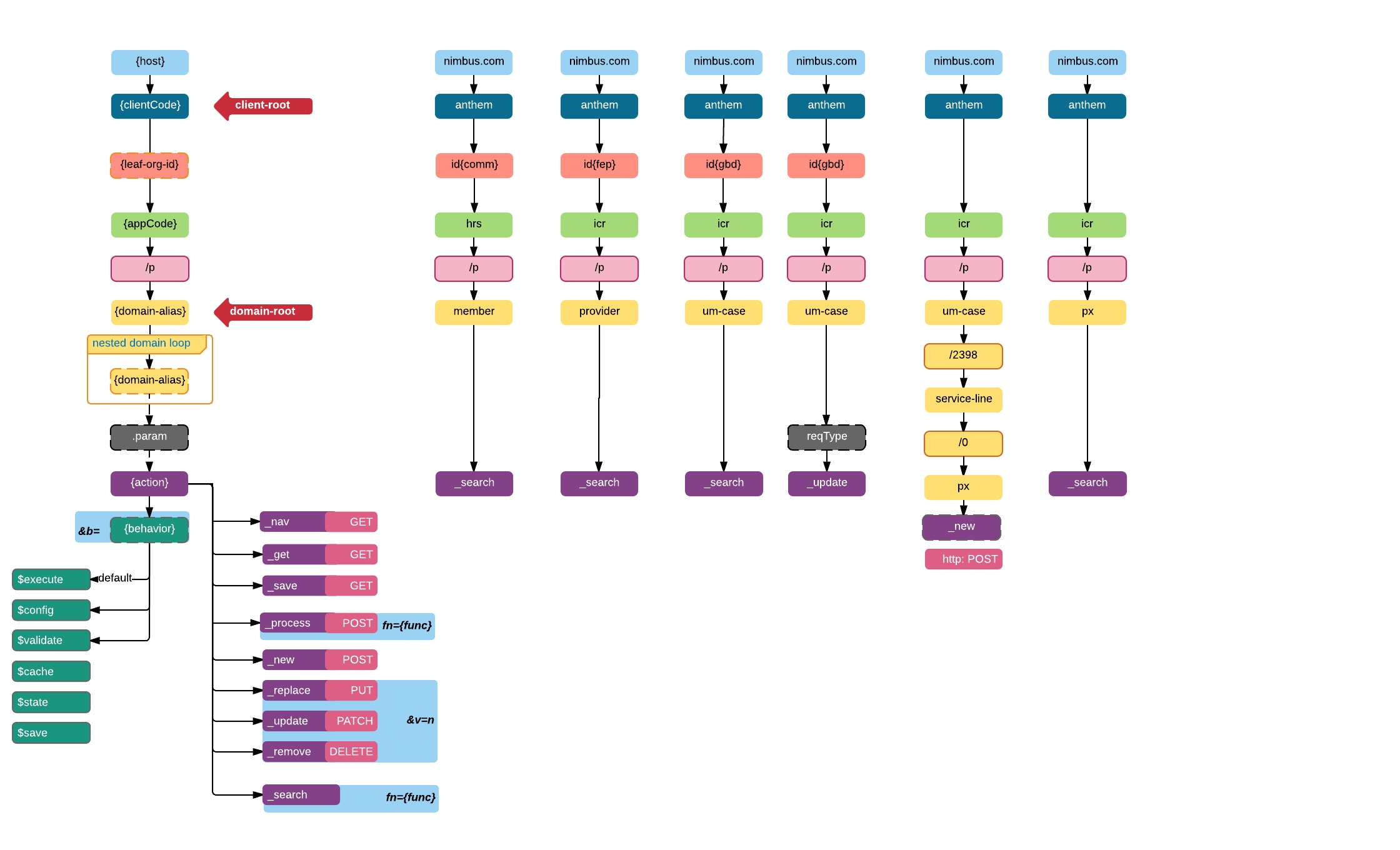
**Config** - Once we have the domain model, we can define the configuration for the view, workflow and the rules. The view definition configs, the mapping to the domain model, the workflow(if any) and the corresponding view and core domain rules can be written.

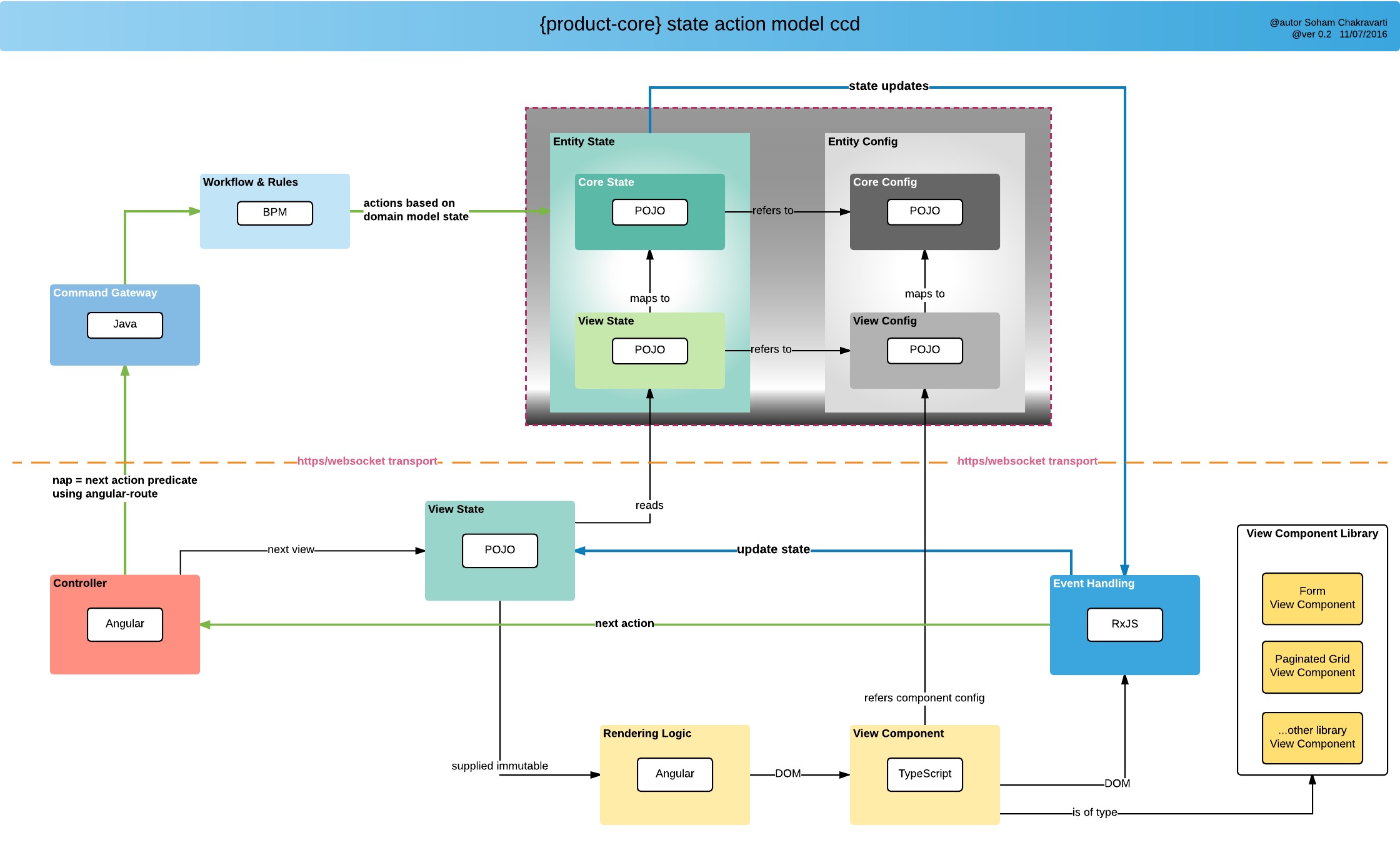
**Command** - The command is the instruction that the framework understands to execute and come back with an output.

**State** - The value of every entity and its corresponding attributes is referred to as state by the framework. There could various events for example generated based on the state and certain other things associated with the state. To get the history of changes that happened on an entity attribute, we would need audit to be enabled.





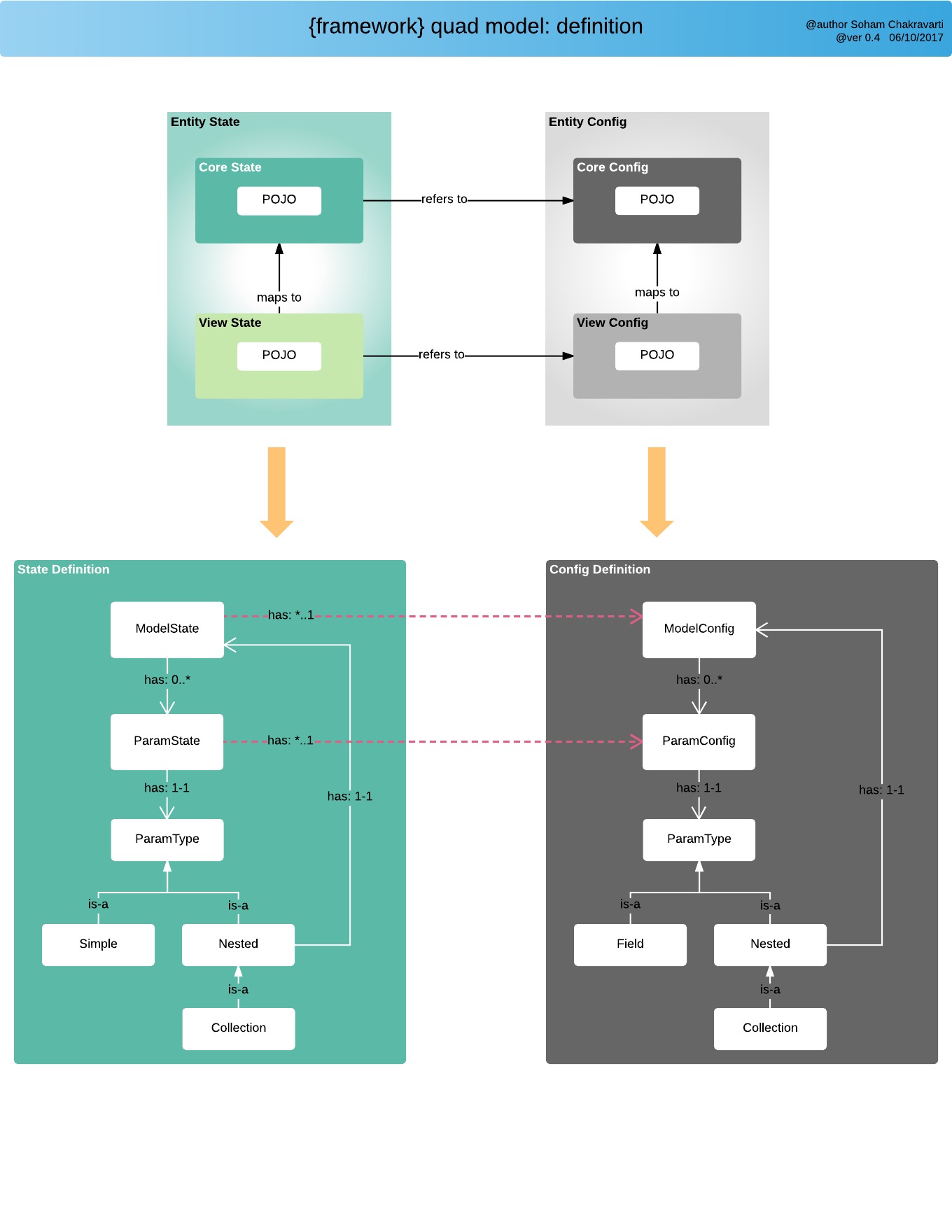




Command

* Target application identifier: Anything prior to /p identifies the application associated with the request
* Domain identifier: Anything post /p identifies the domain for which the request is to be processed.
  + Action
* \_new: Creates a new instance for the model
* \_get: Fetches the instance of the model referenced by the Id
* \_save: Saves the model into the database
* \_replace: Replaces the model state
* \_update: Updates the model state
* \_remove: Removes the model from the database
* \_search: Searches the model based on a search criteria
* \_process: Executes asigned workflow process or custom handlers

Quad Model



Using a url-based Source implementation to define a set of values

|  |
| --- |
| @Values(url="CLIENT\_ID/ORG/p/staticCodeValue/\_search?fn=lookup |

#### State Event Handlers

The role of a State Event Handler is to instruct the framework on how to process a conditional annotation. Therefore, it is responsible for maintaining the logic to be applied when a conditional annotation is present on a param. State Event Handlers are registered with the framework through the use of @EventHandler on startup as Spring beans. The logic is then executed during specific event hooks defined within the framework.

--

### Conditional Config Annotations

This section covers the conditional behavior that can be applied to params. In this context, conditional behavior refers to performing a set of actions based on a given condition. The condition is evaluated by using the powerful capabilities of SpEL (Spring Expression Language).

Typically this conditional behavior is evaluated from the context of the conditionally decorated param (the param that is annotated with a conditional annotation), meaning that the condition will inspect the decorated param object to infer if the condition should be true or false.

there is a .d that signifies the mapping to the core domain model i.e. it will point to the root which is CarEntity.

then switch from view config to core config using .m to assign a new car to attr\_list\_2\_CarNestedEntity list.

### Rule Configuration

Framework comes with support for Drools Rules. The rules can be defined at multiple levels.

**Where can rules be configured?**

* **Entity Level**:
* **Function Handler Level**:
* **Attribute Level**:
* **Configuration Level:**

### Process Configuration

#### Business Process Configuration

All standard BPM functions are available for creating business processes. Business process can be defined to manage business entity lifecycle and for creation of stateless processes that executes complex business functions.

#### Entity Lifecyle Management using BPM

The framework provides the ability to back a business or view entity using a workflow. There might be scenarios where an entity needs to traverse through a series of business steps. Having a BPM defined through standard BPMN construct can help with configuring the business steps and will also provide a visual depiction of the business configuration.

#### Associating a workflow with an entity

Steps for configuring a BPMN process as a function

* Create a BPMN process with a unique process id
* Define the config url with action as **\_process**,**fn** as **\_bpm** and **processId** as the name of the process id. Example **@Config(url="/p/patient:<!/.m/id!>/\_process?fn=\_bpm&processId=createcaseforpatient**

1)private String unmapped\_String; // unmapped

@Path

2)private Integer audit\_Integer; // mapped view to a core

3) @Path(value="/a/b/c/action", linked=false)

private Integer audit\_Integer;  // mapped detached and determine if the path has no links

**Function Handlers** are an abstraction within the framework to execute/handle a common set of f/w instructions for a given **Action**.

Given an action, a particular function handler can be executed by specifying a value for the query parameter **fn**. Ex: fn=\_set

**Where , set** is the Function Handler we have configured to execute as a part of this **@Config**configuration.

default.\_new$execute?fn=\_initEntity

default.\_get$execute?fn=param

default.\_nav$execute?fn=default

default.\_process$execute?fn=\_set

default.\_process$execute?fn=\_update

default.\_process$execute?fn=\_setByRule

default.\_process$execute?fn=\_add

default.\_process$execute?fn=\_bpm

default.\_search$execute?fn=lookup

default.\_search$execute?fn=example

default.\_search$execute?fn=query

default.\_process$execute?fn=\_eval

**public** **final** **class** CommandEvent {

@Event **@interface** OnRootExecute {

@Event **@interface** OnSelfExecute

**public** **final** **class** StateEvent {

@Event **@interface** OnStateLoad

@Event **@interface** OnStateChange

@Event **@interface** OnTxnExecute

**public** **final** **class** ExecutionRuntimeEvent {

@Event **@interface** OnRuntimeStart

@Event **@interface** OnRuntimeStop

**public** **final** **class** ConfigEvent {

@Event **@interface** OnParamCreate

@Config(url="/cmcase\_audit\_history/\_new?fn=initEntity",

kv={

@KeyValue(k="/entityId", v="<!/.d/id!>"),// <! ..evaluate in context of audit\_string.. !>

// @KeyValue(k="/property", v="assignment"), // string literal v="LocalDateTime.now()"

// @KeyValue(k="/oldValue", v="<!param(/).getTransientOldState()!>"), // <! .p1. !> == param(audit\_string).findStateByPath(.p1.)

// @KeyValue(k="/newValue", v="<!/!>"),

// @KeyValue(k="/newValue", v="/oldValue") // <! ..evaluate in context of cmcase\_audit\_history.. !>

// })

// })

@Value("${process.key.regex}")

**private** String processBeanRegex;

**public** **static** **final** Pattern SPECIAL\_REGEX\_CHARS = Pattern.compile("[{}()\\[\\].+\*?^$\\\\|]");

// To set multiple values values in target since the new command does not have handle of the old object

// Ex : @Config(url="/p/queue/\_new?fn=\_initEntity&target=/entityId&json=\"<!/.m/id!>\"&target=/name&json=\"<!/.m/name!>\"")

Param code value provided by:

**private** **static** **final** String DEFAULT\_KEY\_ATTRIBUTE = "id";

**private** **static** **final** String KEY\_VALUE\_SEPERATOR = "&";

DefaultActionExecutorSearch searchExecutor;

/\*\*

\* Search will be in the order:

\* 1. static code values (in below order):

\* 1.1 config server, if not found

\* 1.2 DB

\* 2. Model as code values (in below order)

\* 2.1 config server, if not found

\* 2.2 DB

\*

\*/

Param state:

\* Local is always kept, but follows behind cache if configured.

\*

\* 1. If cache=true, then retrieve state from cache AND set to local before returning if local state is different

\* 2. If cache=false, then throws error

"Follows the order: a)Transient b)Collection c)Nested d)Leaf

use the Action output from the setState to check if the action performed is \_update to return true

Stateholder:

It holds all details of state , by using this nimbus fw will take care which actions are allowed at that particular state and vise versa.

Rules.

Events.

Statelife cycle

Notifications.

MARKER\_URI\_PLATFORM("p"),

MARKER\_URI\_BEHAVIOR("b"),

MARKER\_COLLECTION\_ELEM\_INDEX("{index}"),

MARKER\_PLATFROM\_EXPR\_PREFIX("<!"),

MARKER\_PLATFROM\_EXPR\_SUFFIX("!>"),

MARKER\_SESSION\_SELF("#self"),

MARKER\_COMMAND\_PARAM\_CURRENT\_SELF("#this"),

MARKER\_REF\_ID("#refId"),

MARKER\_ELEM\_ID("#elemId"),

MARKER\_COL\_PARAM("col"),

MARKER\_COL\_PARAM\_EXPR("<!col!>"),

MARKER\_URI\_PAGE\_EXPR("page=y"),

SEPARATOR\_URI("/"),

SEPARATOR\_URI\_PLATFORM(SEPARATOR\_URI.code + MARKER\_URI\_PLATFORM.code), /\* /p \*/

SEGMENT\_PLATFORM\_MARKER(SEPARATOR\_URI\_PLATFORM.code + SEPARATOR\_URI.code), /\* /p/ \*/

SEPARATOR\_URI\_VALUE(":"),

SEPARATOR\_URI\_PARENT(".."),

SEPARATOR\_URI\_ROOT\_DOMAIN(".d"),

SEPARATOR\_URI\_ROOT\_EXEC(".e"),

SEPARATOR\_CONFIG\_ATTRIB("#"),

SEPARATOR\_UNIQUE\_KEYGEN("^"),

SEPARATOR\_BEHAVIOR\_START("$"),

SEPARATOR\_AND("And"),

SEPARATOR\_MAPSTO(".m"),

PREFIX\_FLOW("flow\_"),

PREFIX\_DEFAULT("default."),

PREFIX\_EVENT("e"),

PREFIX\_EVENT\_URI("e"+"\_"),

SUFFIX\_PROPERTY\_STATE("State"),

CODE\_VALUE\_CONFIG\_DELIMITER("-"),

PARAM\_VALUES\_URI\_PREFIX("\*/\*/\*/p/"),

PARAM\_VALUES\_URI\_SUFFIX("/\_lookup"),

KEY\_FUNCTION("fn"),

KEY\_FUNCTION\_NAME("name"),

KEY\_NAV\_ARG\_PAGE\_ID("pageId"),

KEY\_FN\_INITSTATE\_ARG\_TARGET\_PATH("target"),

KEY\_FN\_INITSTATE\_ARG\_JSON("json"),

KEY\_FN\_PARAM\_ARG\_EXPR("expr"),

KEY\_EXECUTE\_PROCESS\_CTX("processContext"),

KEY\_EXECUTE\_EVAL\_ARG("eval"),

KEY\_EXECUTE\_PROCESS\_ID("processId"),

REQUEST\_PARAMETER\_MARKER("?"),

CLIENT\_USER\_KEY("client-user-key"),

REQUEST\_PARAMETER\_URL\_MARKER("url"),

REQUEST\_PARAMETER\_DELIMITER("&"),

PARAM\_ASSIGNMENT\_MARKER("="),

/\* search request param constants \*/

SEARCH\_REQ\_PROJECT\_ALIAS\_MARKER("projection.alias"),

SEARCH\_REQ\_PROJECT\_MAPING\_MARKER("projection.mapsTo"),

SEARCH\_REQ\_AGGREGATE\_MARKER("aggregate"),

SEARCH\_REQ\_AGGREGATE\_COUNT("count"),

SEARCH\_REQ\_FETCH\_MARKER("fetch"),

SEARCH\_REQ\_ORDERBY\_MARKER("orderby"),

SEARCH\_REQ\_WHERE\_MARKER("where"),

SEARCH\_REQ\_PAGINATION\_SIZE("pageSize"),

SEARCH\_REQ\_PAGINATION\_PAGE\_NUM("page"),

SEARCH\_REQ\_PAGINATION\_SORT\_PROPERTY("sortBy"),

SEARCH\_NAMED\_QUERY\_DELIMTER("~~"),

SEARCH\_NAMED\_QUERY\_RESULT("result");

/\*\*

\* View is a perspective of Core. It can be used in presentation layer or can be part of web service integration.<br>

\* Relationship of View to Core is many-to-one. View is not mandated to have a core backing.<br>

\* <br>

\* Within the platform, View is associated to an user, while Core is the same across users.<br>

\* Authorization cross-cutting component ensures that access to Core by an user is valid.<br>

\* <br>

\* An user could potentially be logged into the platform with different sessions. Relationship of Session to View is one-to-one.<br>

\* In the scenario, with 2 users (UserA, UserB) with valid access to a domain-entity (id: D100) will have following relation:<br>

\* <br>

\* UserA\_\_\_\_\_\_\_\_SessionA1\_\_\_\_\_\_\_\_\_QuadA1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_D100 <br>

\* |\\_\_\_\_\_\_\_\_SessionA2\_\_\_\_\_\_\_\_\_QuadA2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/| <br>

\* \\_\_\_\_\_\_\_\_SessionA3\_\_\_\_\_\_\_\_\_QuadA3\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/| <br>

\* | <br>

\* UserB\_\_\_\_\_\_\_\_SessionB1\_\_\_\_\_\_\_\_\_QuadB1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/| <br>

\* \\_\_\_\_\_\_\_\_SessionB2\_\_\_\_\_\_\_\_\_QuadB2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ <br>

\* <br>

\* <br>

\* Domain state is always retrieved directly from the Repository API.<br>

\* Quad State is always obtained from User Session and persisted back to DB on explicit \_save or $save <br>

\* <br>

\* <br>

\* Within a Session, each Flow for a given domain-alias can point to multiple Quad models.<br>

\* <br>

\* Example:<br>

\* flow\_car has two pages. First page is for searching existing cars and second is to display details of any one unique car.<br>

\* <br>

\* flow\_car/\_new - creates new entity and assigns unique persistence id, if configured with {@linkplain Repo}<br>

\* flow\_car:100/\_get - checks if entity exists in session, others retrieves & puts entity in session, if configured as such in {@linkplain Repo}<br>

\* flow\_car/\_search - creates {@linkplain QuadModel} in transient mode and doesn't interact with session<br>

\* <br>

\* Possible URLs:<br>

\* /flow\_car/\_new \_\_\_\_\_Quad1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_new Car()

\* /flow\_car/search \_\_\_\_\_Quad2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Car[instance level of Quad1]

\* /flow\_car/details:100\_\_\_\_\_Quad3\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Car[100]

\* /flow\_car/details:200\_\_\_\_\_Quad4\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Car[200]

\*

\* @author Soham Chakravarti