--

enum Action {

/\* CRUD \*/

\_get, //HTTP GET - defaults to \_detail

\_save, //HTTP GET

\_new, //HTTP POST

\_replace, //HTTP PUT - full update

\_update, //HTTP PATCH- partial update

\_delete, //HTTP DELETE

/\* transient state \*/

\_search,

\_config,

/\* process \*/

\_process, //Allows for custom process/work-flow definitions

/\* navigation \*/

\_nav

;

public static final Action DEFAULT = \_get;

--

public interface CommandExecutorGateway {

default MultiOutput execute(Command cmd, String payload) {

return execute(new CommandMessage(cmd, payload));

}

MultiOutput execute(CommandMessage cmdMsg);

List<MultiOutput> executeConfig(ExecutionContext eCtx, Param<?> cmdParam, List<Execution.Config> execConfigs);

}

--

\*/

public interface ExecutionContextLoader {

public ExecutionContext load(Command rootDomainCmd);

public void unload(ExecutionContext eCtx);

public void clear();

}

--

public interface FunctionHandler<T, R> extends HierarchyMatch {

public R execute(ExecutionContext eCtx, Param<T> actionParameter);

}

--

@Getter @Setter

public class ExecutionContext implements Serializable {

private static final long serialVersionUID = 1L;

private final CommandMessage commandMessage;

private QuadModel<?, ?> quadModel;

public ExecutionContext(Command command) {

this(new CommandMessage(command, null));

}

public ExecutionContext(CommandMessage commandMessage) {

this.commandMessage = commandMessage;

}

public ExecutionContext(CommandMessage commandMessage, QuadModel<?, ?> quadModel) {

this(commandMessage);

setQuadModel(quadModel);

}

public String getId() {

return getCommandMessage().getCommand().getRootDomainUri();

}

public boolean equalsId(Command cmd) {

return StringUtils.equals(getId(), cmd.getRootDomainUri());

}

public ExecutionModel<?> getRootModel(){

return quadModel.getRoot();

}

public <P> Param<P> findParamByPath(String path) {

return getRootModel().findParamByPath(path);

}

public <P> P findStateByPath(String path) {

return getRootModel().findStateByPath(path);

}

@Override

public boolean equals(Object other) {

if(other==null)

return false;

if(!(other instanceof ExecutionContext))

return false;

ExecutionContext otherCtx = (ExecutionContext)other;

String thisDomainRootUri = getId();

String otherDomainRootUri = otherCtx.getId();

return thisDomainRootUri.equals(otherDomainRootUri);

}

@Override

public String toString() {

return getId();

}

}

--

public class ParamPathExpressionParser {

public static final String KEY\_PREFIX = "<!";

public static final String KEY\_SUFFIX = "!>";

public static Map<Integer, String> parse(String in) {

Map<Integer, String> entries = new LinkedHashMap<>();

parse(in, 0, entries);

return entries;

}

Still there

--

Events:

1)public final class StateEvent

T @interface OnStateLoad

T @interface OnStateChange

T @interface OnTxnExecute

2) public final class CommandEvent {

T @interface OnRootExecute

T @interface OnSelfExecute

3) public final class ConfigEvent {

T @interface OnParamCreate

4) public final class ExecutionRuntimeEvent {

T @interface OnRuntimeStart

T @interface OnRuntimeStop

--

@Domain("staticCodeValue")

@Repo(value=Database.rep\_mongodb, cache=Cache.rep\_device)

@Getter @Setter @RequiredArgsConstructor

public class StaticCodeValue extends IdString {

private static final long serialVersionUID = 1L;

private final String paramCode;

private final List<ParamValue> paramValues;

--

--public class ViewConfig {

type @interface DomainMeta {

T @interface ViewParamBehavior {

T @interface ViewStyle {

F @interface ConceptId {

--

@ViewParamBehavior

F @interface Hints {

F @interface Initialize {

F @interface Mode {

ReadOnly,

Hidden,

Inherit

}

Options value() default Options.Inherit;

}

Nimbus cheet sheet:

public class ViewConfig {

@Retention(RetentionPolicy.RUNTIME)

@Target(value={ElementType.ANNOTATION\_TYPE})

@Inherited

public @interface DomainMeta {

}

@Retention(RetentionPolicy.RUNTIME)

@Target(value={ElementType.ANNOTATION\_TYPE})

@Inherited

public @interface ViewParamBehavior {

}

@Retention(RetentionPolicy.RUNTIME)

@Target(value={ElementType.ANNOTATION\_TYPE})

@Inherited

public @interface ViewStyle {

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@DomainMeta

public @interface ConceptId {

String value() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewParamBehavior

public @interface Hints {

public enum AlignOptions {

Left,

Right,

Center,

Inherit;

}

AlignOptions value() default AlignOptions.Inherit;

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewParamBehavior

public @interface Initialize {

String alias() default "initialize";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewParamBehavior

public @interface Mode {

public enum Options {

ReadOnly,

Hidden,

Inherit

}

Options value() default Options.Inherit;

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.TYPE})

@ViewStyle

public @interface ViewRoot {

String alias() default "root";

String layout() default "";

}

/\*

\* Page Definition

\*/

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Page {

String alias() default "Page";

String cssClass() default "";

boolean defaultPage() default false;

String imgSrc() default "";

String route() default ""; // remove

}

/\*

\* Tile Definition

\*/

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Tile {

public enum Size {

XSmall, //25%

Small, //33%

Medium, //50%

Large, //100%

Colorbox //100%

}

String alias() default "Tile";

String imgSrc() default "";

Size size() default Size.Large;

}

/\*

\* Section Definition

\*/

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

//TODO: Will have to remove LEFTBAR and HEADER once the leftbarn& header navigation for case is in place

public @interface Section {

public enum Type {

HEADER,

LEFTBAR,

DEFAULT;

}

Type value() default Type.DEFAULT; // HEADER and LEFTBAR should be removed in future

String alias() default "Section";

String cssClass() default "";

}

/\*

\* Global Footer Definition

\*/

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface GlobalFooter {

String alias() default "Footer";

String cssClass() default "";

}

/\*

\* Global Header Definition

\*/

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface GlobalHeader {

String alias() default "Global-Header";

String cssClass() default "";

}

/\*

\* Global Navigation Menu Definition

\*/

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface GlobalNavMenu {

String alias() default "Global-Nav-Menu";

String cssClass() default "";

}

/\*\*

\* <p>Framework View Style Component: <b>Modal</b></p>

\*

\* <p>Renders a popup window with content defined by the nested fields within the field

\* that is decorated with <tt>&#64;Modal</tt>.</p>

\*

\* <p>

\* <b>Notes:</b>

\* <ul>

\* <li>Default contextual properties are set by <tt>ModalStateEventHandler</tt> during

\* the <tt>OnStateLoad</tt> event.</li>

\* </ul>

\* </p>

\*

\* @author Tony Lopez (AF42192)

\* @see com.anthem.oss.nimbus.core.domain.model.state.extension.ModalStateEventHandler

\*

\*/

@Retention(RetentionPolicy.RUNTIME)

@Target({ ElementType.FIELD })

@ViewStyle

@OnStateLoad

public @interface Modal {

String alias() default "Modal";

String cssClass() default ""; // new

Type type() default Type.dialog;

boolean closable() default false;

String width() default "medium";

public enum Type {

dialog, slider

}

ParamContext context() default @ParamContext(enabled = true, visible = false);

boolean resizable() default true;

}

/\*

\* Container for Form Elements {textbox, combobox.. etc}

\* Form can have Accordions

\* TODO Add Section, Collapsible Sections/Panels instead of Accordions

\*

\*/

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Form {

String alias() default "Form";

String cssClass() default "";

boolean submitButton() default true; // remove

String submitUrl() default ""; // remove

String b() default ""; // remove

String navLink() default ""; // remove

}

/\*

\* Container for Buttons

\*/

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface ButtonGroup {

String alias() default "ButtonGroup";

String cssClass() default "text-sm-center";

}

/\*

\* Grid Container

\* TODO expandable row Grid

\*/

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Grid {

String alias() default "Grid";

String cssClass() default ""; // new

boolean expandableRows() default false;

boolean onLoad() default false;

boolean isTransient() default false;

String url() default "";

boolean rowSelection() default false;

String pageSize() default "25";

boolean showHeader() default true;

boolean pagination() default true;

boolean postButton() default false;

String postButtonUrl() default "";

String postButtonTargetPath() default "";

String postButtonAlias() default "";

String postButtonLabel() default "";

boolean postEventOnChange() default false;

boolean clearAllFilters() default false;

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface GridRowBody {

String alias() default "GridRowBody";

String cssClass() default "";

boolean asynchronous() default false;

}

/\*

\* Card Grid

\*/

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface CardDetailsGrid { // new Introduce Sections within Card Grid for grouping.

String alias() default "CardDetailsGrid";

String editUrl() default "";

boolean draggable() default false;

boolean onLoad() default false;

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Header {

public enum Size {

H1,

H2,

H3,

H4,

H5

}

String alias() default "Header";

Size size() default Size.H3;

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface CardDetail {

String alias() default "CardDetail";

String cssClass() default "";

String imgSrc() default "";

boolean editable() default false;

String modelPath() default "";

boolean draggable() default false;

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Tag {

String alias() default "CardDetailsTag";

String cssClass() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Header {

String alias() default "CardDetailsHeader";

String cssClass() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Body {

String alias() default "CardDetailsBody";

String cssClass() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Footer {

String alias() default "CardDetailsFooter";

String cssClass() default "";

}

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface FieldValue {

public enum Type {

Field,

Divider

}

String alias() default "FieldValue";

Type type() default Type.Field;

String imgSrc() default "";

String cssClass() default "";

boolean showName() default true;

String cols() default "4";

String iconField() default "";

boolean inplaceEdit() default false;

String inplaceEditType() default "";

String datePattern() default "";

String placeholder() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Assessment {

String alias() default "Assessment";

String cssClass() default "text-sm-right";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Button {

public enum Style {

PRIMARY,

SECONDARY,

PLAIN,

DESTRUCTIVE;

}

public enum Type {

submit,

reset,

button

}

Type type() default Type.button;

String alias() default "Button";

String url() default "";

String b() default "$execute";

String method() default "GET";

String imgSrc() default "";

Style style() default Style.PLAIN;

String payload() default "";

String cssClass() default "btn btn-plain";

boolean browserBack() default false;

boolean formReset() default true;

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface FilterButton {

String alias() default "FilterButton";

String url() default "";

String b() default "$execute";

String method() default "GET";

String imgSrc() default "";

String cssClass() default "btn btn-primary";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Menu {

public enum Type {

CONTEXT;

}

Type value() default Type.CONTEXT;

String alias() default "Menu";

String cssClass() default "";

String text() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface LinkMenu {

String cssClass() default "";

String alias() default "LinkMenu";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Link {

public enum Type {

MENU,

EXTERNAL,

DEFAULT;

}

Type value() default Type.DEFAULT;

String alias() default "Link";

String url() default "";

String method() default "GET";

String b() default "$executeAnd$nav";

String imgSrc() default "";

String cssClass() default "";

String altText() default "";

String target() default "";

String rel() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface InPlaceEdit {

String alias() default "InPlaceEdit";

String type() default "text";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewParamBehavior

public @interface PageHeader {

public enum Property {

LOGO,

APPTITLE,

SUBTITLE,

USERNAME,

USERROLE,

SUBHEADER,

MENU,

DEFAULT,

SETTINGS,

NOTIFICATIONS,

NUMBEROFNOTIFICATIONS,

HELP,

LOGOUT;

}

Property value() default Property.DEFAULT;

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewParamBehavior

public @interface FooterProperty {

public enum Property {

DISCLAIMER,

LINK,

SSLCERT

}

Property value() default Property.DISCLAIMER;

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Paragraph {

String alias() default "Paragraph";

String cssClass() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Image {

String alias() default "Image";

String imgSrc() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface TextBox {

String alias() default "TextBox";

boolean hidden() default false;

boolean readOnly() default false;

String help() default "";

String labelClass() default "anthem-label";

String type() default "text";

boolean postEventOnChange() default false;

String controlId() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface TextArea {

String alias() default "TextArea";

boolean hidden() default false;

boolean readOnly() default false;

String help() default "";

String labelClass() default "anthem-label";

String type() default "textarea";

String rows() default "5";

boolean postEventOnChange() default false;

String controlId() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface ComboBox {

String alias() default "ComboBox";

boolean readOnly() default false;

String labelClass() default "anthem-label";

boolean postEventOnChange() default false;

String postButtonUrl() default "";

String controlId() default "";

String help() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface StaticText {

String alias() default "StaticText";

String contentId() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface MultiSelect {

String alias() default "MultiSelect";

String labelClass() default "anthem-label";

boolean postEventOnChange() default false;

String help() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface InputDate {

String alias() default "InputDate";

boolean readOnly() default false;

String labelClass() default "anthem-label";

String type() default "date";

boolean postEventOnChange() default false;

String controlId() default "";

String help() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Calendar {

String alias() default "Calendar";

boolean readOnly() default false;

String labelClass() default "anthem-label";

String type() default "calendar";

boolean showTime() default false;

String hourFormat() default "12";

boolean timeOnly() default false;

boolean postEventOnChange() default false;

String controlId() default "";

String help() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface GridContainer {

String alias() default "GridContainer";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Radio {

String alias() default "Radio";

String labelClass() default "anthem-label";

String level() default "0";

String cssClass() default "";

boolean postEventOnChange() default false;

String controlId() default "";

String help() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface CheckBox {

String alias() default "CheckBox";

String cssClass() default "";

String labelClass() default "anthem-label";

boolean postEventOnChange() default false;

String controlId() default "";

String help() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface CheckBoxGroup {

String alias() default "CheckBoxGroup";

String level() default "0";

String cssClass() default "";

String labelClass() default "anthem-label";

boolean postEventOnChange() default false;

String controlId() default "";

String help() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ ElementType.FIELD })

@ViewStyle

public @interface AccordionMain {

String alias() default "AccordionMain";

String cssClass() default "panel-default";

boolean multiple() default false;

String activeIndex() default "0";

boolean showExpandAll() default false;

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ ElementType.FIELD })

@ViewStyle

public @interface AccordionTab {

String alias() default "AccordionTab";

String cssClass() default "panel-default";

boolean selected() default false;

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ ElementType.FIELD })

@ViewStyle

public @interface Accordion {

String alias() default "Accordion";

String cssClass() default "panel-default";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ ElementType.FIELD })

@ViewStyle

public @interface AccordionGroup {

String alias() default "AccordionGroup";

String cssClass() default "panel-default";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface MultiSelectCard {

String alias() default "MultiSelectCard";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface MultiGrid {

String alias() default "MultiGrid";

String level() default "0";

String header() default "test";

String cssClass() default "question-header";

boolean postEventOnChange() default false;

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface GlobalSection {

String alias() default "globalSection";

String imgSrc() default "";

String cssClass() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface Summary {

String alias() default "summary";

String imgSrc() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface BreadCrumb {

String label();

int order();

String alias() default "breadCrumb";

boolean postEventOnChange() default false;

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface LinearGauge {

String alias() default "LinearGauge";

String labelClass() default "anthem-label";

boolean postEventOnChange() default false;

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface ContentContainer {

String alias() default "ContentContainer";

String content() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface SubHeader {

String alias() default "SubHeader";

String cssClass() default "col-sm-6 pb-0 align-top"; //pb-0 is added for the demo. It is temp fix

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface PickList {

String alias() default "PickList";

String labelClass() default "anthem-label";

boolean readOnly() default false;

String cssClass() default "";

boolean postEventOnChange() default false;

String sourceHeader() default "SourceList";

String targetHeader() default "TargetList";

String help() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target({ ElementType.FIELD })

@ViewStyle

public @interface FileUpload {

public enum ControlType {

FORMCONTROL

}

boolean multiple() default true;

String url() default "";

String alias() default "FileUpload";

String type() default ".pdf,.png";

ControlType controlType() default ControlType.FORMCONTROL;

}

/\*\*

\* Following GridColumn attributes can be provided on individual gridcolumns within a grid.<br>

\* <ul><li>hidden - Setting this to true will hide the column in the grid</li>

\* <li>sortable - Enables sorting on the column</li>

\* <li>filter - Setting this to true will enable a explicit filter component(Input TextBox) on the UI</li>

\* <li>filterValue - Default filterValue for a particular column.

\* Can be used without setting filter attribute to restrict user from providing any other filter value.

\* <li>filterMode - Mode to filter data within the column. It is defaulted to 'equals'

\* Current filter modes supported by the Grid component are listed as enum values</li></ul>

\*

\*/

@Retention(RetentionPolicy.RUNTIME)

@Target({ElementType.FIELD})

@ViewStyle

public @interface GridColumn {

String alias() default "GridColumn";

boolean hidden() default false;

boolean sortable() default true;

boolean filter() default false;

String filterValue() default "";

boolean expandable() default true;

SortAs sortAs() default SortAs.DEFAULT; // number, text

String placeholder() default "";

public enum FilterMode {

equals,

contains,

endsWith,

in

}

public enum SortAs {

DEFAULT,

NUMBER,

TEXT;

}

FilterMode filterMode() default FilterMode.equals;

String datePattern() default "";

}

}

--

More:

/\*\*

\* <p>ParamContext is used to set the contextual properties of a field during the OnStateLoad event.

\* Contextual properties include those such as: visible, enabled, etc. It is likely that as the framework

\* evolves, additional values will be defined within <tt>ParamContext</tt>.

\*

\* <p>The intent of <tt>&#64;ParamContext</tt> is to be able to decorate fields with default contextual

\* behavior. For example:</p>

\*

\* <pre>

\* public static class SampleView {

\*

\* &#64;Modal(context = &#64;ParamContext(enabled=true, visible=true))

\* private MyModal myModal;

\*

\* public static class MyModal { ... }

\* }

\* </pre>

\*

\* <p>In this scenario we have configured the contextual values for enabled and visible to be

\* <b>true</b>. These values will be set during the <tt>OnStateLoad</tt> event.</p>

\*

\* @author Soham Chakravarti, Tony Lopez (AF42192)

\* @see com.antheminc.oss.nimbus.domain.model.state.extension.ParamContextStateEventHandler

\*

\*/

@Documented

@Retention(RUNTIME)

@Target({FIELD, ANNOTATION\_TYPE})

@OnStateLoad

public @interface ParamContext {

/\*\*

\* Whether or not the decorated target should be visible.

\*/

boolean visible();

/\*\*

\* Whether or not the decorated target should be enabled.

\*/

boolean enabled();

}