Command:

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

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

https://github.com/openanthem/nimbus-core/blob/master/nimbus-core/src/main/java/com/antheminc/oss/nimbus/domain/cmd/exec/internal/DefaultActionExecutorConfig.java

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

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

<https://github.com/openanthem/nimbus-core/blob/master/nimbus-core/src/main/java/com/antheminc/oss/nimbus/domain/cmd/exec/internal/nav/DefaultActionNewInitEntityFunctionHandler.java>

<https://github.com/openanthem/nimbus-core/blob/master/nimbus-core/src/main/java/com/antheminc/oss/nimbus/domain/cmd/exec/internal/nav/DefaultActionNewInitEntityFunctionHandler.java>

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

<https://github.com/openanthem/nimbus-core/blob/master/nimbus-core/src/main/java/com/antheminc/oss/nimbus/channel/web/WebCommandBuilder.java> god

// input doesn't have /p/ : prefix client/org/app/p/{domain-root} from incoming command

--

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

@Getter @RequiredArgsConstructor

public enum ProcessExecutorEvents {

pre(Constants.PREFIX\_EVENT\_URI.code + "pre"),

post(Constants.PREFIX\_EVENT\_URI.code + "post"),

error(Constants.PREFIX\_EVENT\_URI.code + "error"),

;

final public String code;

}

public enum Constants {

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");

public final String code;

private Constants(String code) {

this.code = code;

}

@Override

public String toString() {

return "["+name() + " : " +code+"]";

}

}

|  |
| --- |
| // .../p/flow\_um-case/\_process?b=$save |
|  |  |
|  | // .../p/flow\_um-case/\_findPatient/\_process?b=$execute |
|  |  |
|  | // .../p/flow\_um-case/\_findPatient/\_process?b=$executeAndSave |
|  |  |
|  | // .../p/flow\_um-case/requestType/\_update?b=$executeAndSave |
|  |  |
|  | // url-> .../icr/\*\*/flow\_e\_ AND b=null ==> b=$execute |
|  | // url-> .../icr/p/flow\_s\_um-case b==null ==> b=$execute&b=$save |
|  |  |
|  | // url-> .../acmp/\*\*/ AND b=null ==> b=$executeAndSave OR b=$execute&b=$save |
|  |  |
|  | // .../icr/p/flow\_abc/patient/\_save |
|  | } |

|  |
| --- |
| public 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 static Action getByName(String name) { |
|  | return Stream.of(Action.values()) |
|  | .filter((action) -> StringUtils.equals(action.name(), name)) |
|  | .findFirst() |
|  | .orElse(null); |
|  | } |
|  | } |

public final class CommandExecution {

@RequiredArgsConstructor @ToString

private static class ActionBehavior {

@Getter private final String inputCommandUri;

@JsonIgnore

@Getter private final ExecutionContext context;

@Getter private final Action action;

private final List<Behavior> behaviors;

protected List<Behavior> getBehaviors() {

return Optional.ofNullable(behaviors).map(Collections::unmodifiableList).orElse(Collections.emptyList());

}

}

@Getter @ToString(callSuper=true)

public static class Input extends ActionBehavior {

public Input(String inputCommandUri, ExecutionContext context, Action action, Behavior b) {

super(inputCommandUri, context, action, Arrays.asList(b));

}

public Behavior getBehavior() {

return getBehaviors().get(0);

}

}

@Getter @Setter @ToString(callSuper=true)

public static class Output<T> extends ActionBehavior {

private T value;

private String rootDomainId;

private ValidationResult validation;

private ExecuteError error;

@JsonIgnore

@Getter @Setter

private Set<ParamEvent> aggregatedEvents = new HashSet<>();

public Output(String inputCommandUri, ExecutionContext context, Action action, Behavior b) {

this(inputCommandUri, context, action, Arrays.asList(b));

}

public Output(String inputCommandUri, ExecutionContext context, Action action, List<Behavior> behaviors) {

this(inputCommandUri, context, action, behaviors, null);

}

public Output(String inputCommandUri, ExecutionContext context, Action action, List<Behavior> behaviors, T value) {

super(inputCommandUri, context, action, behaviors);

setValue(value);

setRootDomainId(context.getCommandMessage().getCommand().getRootDomainElement().getRefId());

}

public static <T> Output<T> instantiate(Input input, ExecutionContext eCtx) {

return new Output<>(input.getInputCommandUri(), eCtx, input.getAction(), input.getBehavior());

}

public static <T> Output<T> instantiate(Input input, ExecutionContext eCtx, T value) {

Output<T> output = instantiate(input, input.getAction(), eCtx, value);

return output;

}

public static <T> Output<T> instantiate(Input input, Action a, ExecutionContext eCtx, T value) {

Output<T> output = new Output<>(input.getInputCommandUri(), eCtx, a, input.getBehavior());

output.setValue(value);

output.setRootDomainId(eCtx.getCommandMessage().getCommand().getRootDomainElement().getRefId());

return output;

}

@Override

public List<Behavior> getBehaviors() {

return super.getBehaviors();

}

}

public static class EventOutput<T> extends Output<T> {

public EventOutput(Action action, Behavior b) {

super(null, null, action, b);

}

public EventOutput(T value, Action action, Behavior b) {

this(action, b);

setValue(value);

}

}

@ToString(callSuper=true)

public static class MultiOutput extends Output<Object> {

@Getter @Setter

private List<Output<?>> outputs;

public MultiOutput(String inputCommandUri, ExecutionContext context, Action action, Behavior b) {

super(inputCommandUri, context, action, b);

}

public MultiOutput(String inputCommandUri, ExecutionContext context, Action action, List<Behavior> behaviors) {

super(inputCommandUri, context, action, behaviors);

}

@JsonIgnore

private final CollectionsTemplate<List<Output<?>>, Output<?>> template = CollectionsTemplate.linked(this::getOutputs, this::setOutputs);

public CollectionsTemplate<List<Output<?>>, Output<?>> template() {

return template;

}

@JsonIgnore

public Object getSingleResult() {

if(CollectionUtils.isEmpty(getOutputs())) return null;

// if(getOutputs().size() > 1)

// throw new IllegalStateException("Multi output contains more than one output elements: "+getOutputs());

return getOutputs().get(0).getValue();

}

}

}

Nimbus r n d:

org.springframework.boot.autoconfigure.EnableAutoConfiguration=com.antheminc.oss.nimbus.app.extension.config.BPMEngineConfig,\

com.antheminc.oss.nimbus.app.extension.config.DefaultCoreConfiguration,\

com.antheminc.oss.nimbus.app.extension.config.DefaultMongoConfig,\

com.antheminc.oss.nimbus.app.extension.config.WebConfig,\

com.antheminc.oss.nimbus.app.extension.config.SwaggerConfig,\

com.antheminc.oss.nimbus.app.extension.config.DefaultProcessConfig,\

com.antheminc.oss.nimbus.app.extension.config.DefaultCoreExecutorConfig,\

com.antheminc.oss.nimbus.app.extension.config.DefaultCoreBuilderConfig,\

com.antheminc.oss.nimbus.app.extension.config.ActivitiProcessAsBeanRegistrar,\

com.antheminc.oss.nimbus.app.extension.config.DefaultFrameworkExtensionsConfig

package com.antheminc.oss.nimbus.domain.model.state;StateHolder bvn

getPath()

getBeanPath()

--

|  |
| --- |
| /\*\* |
|  | \* Copyright 2016-2018 the original author or authors. |
|  | \* |
|  | \* Licensed under the Apache License, Version 2.0 (the "License"); |
|  | \* you may not use this file except in compliance with the License. |
|  | \* You may obtain a copy of the License at |
|  | \* |
|  | \* http://www.apache.org/licenses/LICENSE-2.0 |
|  | \* |
|  | \* Unless required by applicable law or agreed to in writing, software |
|  | \* distributed under the License is distributed on an "AS IS" BASIS, |
|  | \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. |
|  | \* See the License for the specific language governing permissions and |
|  | \* limitations under the License. |
|  | \*/ |
|  | package com.antheminc.oss.nimbus.domain.model.state; |
|  |  |
|  | import java.beans.PropertyDescriptor; |
|  | import java.util.List; |
|  |  |
|  | import com.antheminc.oss.nimbus.InvalidConfigException; |
|  | import com.antheminc.oss.nimbus.InvalidOperationAttemptedException; |
|  | import com.antheminc.oss.nimbus.domain.cmd.Action; |
|  | import com.antheminc.oss.nimbus.domain.defn.extension.ValidateConditional.ValidationGroup; |
|  | import com.antheminc.oss.nimbus.domain.model.config.EntityConfig; |
|  | import com.antheminc.oss.nimbus.domain.model.config.ParamConfig; |
|  | import com.antheminc.oss.nimbus.domain.model.config.ParamValue; |
|  | import com.antheminc.oss.nimbus.domain.model.state.EntityState.Param; |
|  | import com.antheminc.oss.nimbus.support.pojo.LockTemplate; |
|  |  |
|  | import lombok.Getter; |
|  |  |
|  | public class StateHolder { |
|  |  |
|  | public static class EntityStateHolder<T, SE extends EntityState<T>> implements EntityState<T> { |
|  | protected final SE ref; |
|  |  |
|  | public EntityStateHolder(SE stateRef) { |
|  | this.ref = stateRef; |
|  | } |
|  |  |
|  | @Override |
|  | public String getPath() { |
|  | return this.ref.getPath(); |
|  | } |
|  |  |
|  | @Override |
|  | public String getBeanPath() { |
|  | return this.ref.getBeanPath(); |
|  | } |
|  |  |
|  | @Override |
|  | public EntityConfig<T> getConfig() { |
|  | throw throwEx(); //return this.ref.getConfig(); |
|  | } |
|  |  |
|  | @Override |
|  | public String getConfigId() { |
|  | return this.ref.getConfigId(); |
|  | } |
|  |  |
|  | @Override |
|  | public <S> S findStateByPath(String path) { |
|  | return this.ref.findStateByPath(path); |
|  | } |
|  |  |
|  | @Override |
|  | public <S> Model<S> findModelByPath(String path) { |
|  | throw throwEx(); //return this.ref.findModelByPath(path); |
|  | } |
|  |  |
|  | @Override |
|  | public <S> Model<S> findModelByPath(String[] pathArr) { |
|  | throw throwEx(); //return this.ref.findModelByPath(pathArr); |
|  | } |
|  |  |
|  | @Override |
|  | public <P> Param<P> findParamByPath(String path) { |
|  | Param<P> p = this.ref.findParamByPath(path); |
|  | return findParamByPath(p); |
|  | } |
|  |  |
|  | @Override |
|  | public <P> Param<P> findParamByPath(String[] pathArr) { |
|  | Param<P> p = this.ref.findParamByPath(pathArr); |
|  | return findParamByPath(p); |
|  | } |
|  |  |
|  | @SuppressWarnings("unchecked") |
|  | protected <P> Param<P> findParamByPath(Param<P> p) { |
|  | if(p==null) |
|  | return null; |
|  |  |
|  | if((p==this.ref || p==this) |
|  | && ParamStateHolder.class.isInstance(this)) |
|  | return ParamStateHolder.class.cast(this); |
|  |  |
|  | return new ParamStateHolder.Writeable<>(p); |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isMapped() { |
|  | return this.ref.isMapped(); |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isRoot() { |
|  | return this.ref.isRoot(); |
|  | } |
|  |  |
|  | @Override |
|  | public Model<?> getRootDomain() { |
|  | throw throwEx(); //return this.ref.getRootDomain(); |
|  | } |
|  |  |
|  | @Override |
|  | public ExecutionModel<?> getRootExecution() { |
|  | throw throwEx(); //return this.ref.getRootExecution(); |
|  | } |
|  |  |
|  | @Override |
|  | public void fireRules() { |
|  | this.ref.fireRules(); |
|  | } |
|  |  |
|  | /\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Operations Not Allowed \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/ |
|  | @Override |
|  | public void initSetup() { |
|  | throw throwEx(); |
|  | } |
|  |  |
|  | @Override |
|  | public void initState() { |
|  | throw throwEx(); |
|  | } |
|  |  |
|  | public boolean onLoad() { |
|  | return !isStateInitialized(); |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isStateInitialized() { |
|  | return this.ref.isStateInitialized(); |
|  | } |
|  |  |
|  | @Override |
|  | public void setStateInitialized(boolean initialized) { |
|  | throw throwEx(); |
|  | } |
|  |  |
|  | protected InvalidOperationAttemptedException throwEx() { |
|  | return throwEx("Attempted operation not allowed in the config used on param: "+this.ref |
|  | +", if this is needed - pls submit a feature request to bring it up for evaluation."); |
|  | } |
|  |  |
|  | protected InvalidOperationAttemptedException throwEx(String errMsg) { |
|  | return new InvalidOperationAttemptedException(errMsg); |
|  | } |
|  |  |
|  | @Override |
|  | public EntityStateAspectHandlers getAspectHandlers() { |
|  | throw throwEx(); |
|  | } |
|  |  |
|  | @Override |
|  | public LockTemplate getLockTemplate() { |
|  | throw throwEx(); |
|  | } |
|  |  |
|  | @Override |
|  | public Mapped<T, ?> findIfMapped() { |
|  | throw throwEx(); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | public static class ParamStateHolder<T> extends EntityStateHolder<T, Param<T>> implements Param<T> { |
|  |  |
|  | @Getter |
|  | private final T state; |
|  |  |
|  | public ParamStateHolder(Param<T> param) { |
|  | super(param); |
|  | this.state = param.getLeafState(); |
|  | } |
|  |  |
|  | public static class Writeable<S> extends ParamStateHolder<S> { |
|  | public Writeable(Param<S> from) { |
|  | super(from); |
|  | } |
|  |  |
|  | @Override |
|  | public S getState() { |
|  | return this.ref.getState(); |
|  | } |
|  |  |
|  | @Override |
|  | public Action setState(S state) { |
|  | return this.ref.setState(state); |
|  | } |
|  | } |
|  |  |
|  | public static class Mapped<S, M> extends ParamStateHolder<S> implements MappedParam<S, M> { |
|  |  |
|  | private final ParamStateHolder<M> mapsTo; |
|  |  |
|  | public Mapped(MappedParam<S, M> mapped, ParamStateHolder<M> mapsTo) { |
|  | super(mapped); |
|  | this.mapsTo = mapsTo; |
|  | } |
|  |  |
|  | @Override |
|  | public Param<M> getMapsTo() { |
|  | return mapsTo; |
|  | } |
|  |  |
|  | @Override |
|  | public MappedParam<S, M> findIfMapped() { |
|  | return this; |
|  | } |
|  |  |
|  | @Override |
|  | public void handleNotification(Notification<M> event) { |
|  | throw throwEx(); |
|  | } |
|  |  |
|  | @Override |
|  | public boolean requiresConversion() { |
|  | throw throwEx(); |
|  | } |
|  | } |
|  |  |
|  | @Override |
|  | public ParamConfig<T> getConfig() { |
|  | throw throwEx(); //return ref.getConfig(); |
|  | } |
|  |  |
|  | @Override |
|  | public T getLeafState() { |
|  | return this.ref.getLeafState(); |
|  | } |
|  |  |
|  | @Override |
|  | public Action setState(T state) { |
|  | if(this.ref.isStateInitialized()) |
|  | throw throwEx("'setState' not allowed once parameter is initialized to avoid circular event trigger on param: "+this.ref |
|  | +", if this is needed - pls submit a feature request to bring it up for evaluation."); |
|  |  |
|  | return this.ref.setState(state); |
|  | } |
|  |  |
|  |  |
|  | @Override |
|  | public Model<?> getParentModel() { |
|  | throw throwEx(); //return this.ref.getParentModel(); |
|  | } |
|  |  |
|  |  |
|  | @Override |
|  | public StateType getType() { |
|  | throw throwEx(); //return this.ref.getType(); |
|  | } |
|  |  |
|  | @Override |
|  | public ListParam findIfCollection() { |
|  | throw throwEx(); //return this.ref.findIfCollection(); |
|  | } |
|  |  |
|  | @Override |
|  | public ListElemParam<T> findIfCollectionElem() { |
|  | throw throwEx(); //return this.ref.findIfCollectionElem(); |
|  | } |
|  |  |
|  | @Override |
|  | public LeafParam<T> findIfLeaf() { |
|  | throw throwEx(); //return this.ref.findIfLeaf(); |
|  | } |
|  |  |
|  | @Override |
|  | public Param<?> findIfLinked() { |
|  | throw throwEx(); //return this.ref.findIfLinked(); |
|  | } |
|  |  |
|  | @SuppressWarnings({ "unchecked", "rawtypes" }) |
|  | @Override |
|  | public MappedParam<T, ?> findIfMapped() { |
|  | MappedParam<T, ?> mp = this.ref.findIfMapped(); |
|  | if(mp==null) |
|  | return null; |
|  |  |
|  |  |
|  | ParamStateHolder<?> mapsTo = new ParamStateHolder<>(mp.getMapsTo()); |
|  | return new ParamStateHolder.Mapped(mp, mapsTo); |
|  | } |
|  |  |
|  | @Override |
|  | public Model<T> findIfNested() { |
|  | throw throwEx(); //return this.ref.findIfNested(); |
|  | } |
|  |  |
|  | @Override |
|  | public MappedTransientParam<T, ?> findIfTransient() { |
|  | return this.ref.findIfTransient(); |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isCollection() { |
|  | return this.ref.isCollection(); |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isCollectionElem() { |
|  | return this.ref.isCollectionElem(); |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isLeaf() { |
|  | return this.ref.isLeaf(); |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isLeafOrCollectionWithLeafElems() { |
|  | return this.ref.isLeafOrCollectionWithLeafElems(); |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isLinked() { |
|  | return this.ref.isLinked(); |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isNested() { |
|  | return this.ref.isNested(); |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isTransient() { |
|  | return this.ref.isTransient(); |
|  | } |
|  |  |
|  | public boolean isAssigned() { |
|  | if(!isTransient()) |
|  | throw new InvalidConfigException("Attempted method on non-transient parameter: "+this.ref); |
|  |  |
|  | return findIfTransient().isAssinged(); |
|  | } |
|  |  |
|  | /\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Parameter Context State Attributes \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/ |
|  | @Override |
|  | public void activate() { |
|  | this.ref.activate(); |
|  | } |
|  |  |
|  | @Override |
|  | public void deactivate() { |
|  | this.ref.deactivate(); |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isActive() { |
|  | return this.ref.isActive(); |
|  | } |
|  |  |
|  | @Override |
|  | public Class<? extends ValidationGroup>[] getActiveValidationGroups() { |
|  | return this.ref.getActiveValidationGroups(); |
|  | } |
|  | @Override |
|  | public void setActiveValidationGroups(Class<? extends ValidationGroup>[] activeValidationGroups) { |
|  | this.ref.setActiveValidationGroups(activeValidationGroups); |
|  | } |
|  |  |
|  | @Override |
|  | public Message getMessage() { |
|  | return this.ref.getMessage(); |
|  | } |
|  |  |
|  | @Override |
|  | public void setMessage(Message msg) { |
|  | this.ref.setMessage(msg); |
|  | } |
|  |  |
|  | @Override |
|  | public List<ParamValue> getValues() { |
|  | return this.ref.getValues(); |
|  | } |
|  | @Override |
|  | public void setValues(List<ParamValue> values) { |
|  | this.ref.setValues(values); |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isEnabled() { |
|  | return this.ref.isEnabled(); |
|  | } |
|  | @Override |
|  | public void setEnabled(boolean enabled) { |
|  | this.setEnabled(enabled); |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isVisible() { |
|  | return this.ref.isVisible(); |
|  | } |
|  | @Override |
|  | public void setVisible(boolean visible) { |
|  | this.ref.setVisible(visible); |
|  | } |
|  |  |
|  | /\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Operations Not Allowed \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/ |
|  | @Override |
|  | public boolean deregisterConsumer(MappedParam<?, T> consumer) { |
|  | throw throwEx(); |
|  | } |
|  |  |
|  | @Override |
|  | public void emitNotification(Notification<T> event) { |
|  | throw throwEx(); |
|  | } |
|  |  |
|  |  |
|  | @Override |
|  | public List<MappedParam<?, T>> getEventSubscribers() { |
|  | throw throwEx(); |
|  | } |
|  |  |
|  |  |
|  | @Override |
|  | public PropertyDescriptor getPropertyDescriptor() { |
|  | throw throwEx(); |
|  | } |
|  |  |
|  |  |
|  | @Override |
|  | public void onStateLoadEvent() { |
|  | throw throwEx(); |
|  | } |
|  |  |
|  | @Override |
|  | public void onStateChangeEvent(ExecutionTxnContext txnCtx, Action a) { |
|  | throw throwEx(); |
|  | } |
|  |  |
|  | @Override |
|  | public void registerConsumer(MappedParam<?, T> consumer) { |
|  | throw throwEx(); |
|  | } |
|  |  |
|  | } |
|  | } |

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

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

<https://github.com/openanthem/nimbus-core/tree/master/nimbus-core/src/main/java/com/antheminc/oss/nimbus/domain/model/state/internal> bvn

package com.antheminc.oss.nimbus.domain.defn; god

|  |
| --- |
| public @interface Execution { |
|  |  |
|  | /\*\* |
|  | \* Only the first execution config would have access to supplied payload. |
|  | \* |
|  | \* @author Soham Chakravarti |
|  | \*/ |
|  | @Retention(RetentionPolicy.RUNTIME) |
|  | @Target({ElementType.FIELD}) |
|  | @Repeatable(Configs.class) |
|  | @Execution |
|  | public @interface Config { |
|  |  |
|  | String url(); |
|  |  |
|  | String col() default ""; |
|  |  |
|  | KeyValue[] kv() default {}; |
|  | } |
|  |  |
|  | public @interface KeyValue { |
|  |  |
|  | String k(); |
|  | String v(); |
|  | } |
|  |  |
|  | } |