|  |  |  |  |
| --- | --- | --- | --- |
|  | @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 @interface Config {    String url();    String col() default "";    KeyValue[] kv() default {};  }    public @interface KeyValue {    String k();  String v();  }  'use strict'; | | |
|  |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | |  | |
|  | | export class ServiceConstants { | |
|  | | private static hostName: string; | |
|  | | private static port: string; | |
|  | | private static protocol: string; | |
|  | | private static locale : string; | |
|  | |  | |
|  | | public static get PAGE\_INDEX() : number { return 0;} | |
|  | | public static get PAGE\_SIZE() : number { return 20;} | |
|  | | public static get URL\_SEPERATOR(): string { return '/'; } | |
|  | | public static get WS\_SUBSCRIBE\_Q(): string { return '/user/queue/updates'; } | |
|  | | public static get LOCALE\_LANGUAGE() : string { return this.locale;} | |
|  | | public static set LOCALE\_LANGUAGE(locale : string ) {this.locale = locale;} | |
|  | |  | |
|  | | /\* Enable for stopgap server \*/ | |
|  | | public static get BASE\_URL(): string { return this.STOPGAP\_APP\_HOST\_URL+this.WEB\_CONTENT\_PORT; } | |
|  | | public static get LOGIN\_URL(): string { return this.STOPGAP\_APP\_HOST\_URL+this.STOPGAP\_APP\_PORT+'/login'; } | |
|  | | public static get PLATFORM\_BASE\_URL(): string { return this.STOPGAP\_APP\_HOST\_URL+this.STOPGAP\_APP\_PORT+'/Anthem/fep/p'; } | |
|  | | public static get CLIENT\_BASE\_URL(): string { return this.STOPGAP\_APP\_HOST\_URL+this.STOPGAP\_APP\_PORT+'/'; } | |
|  | | public static get WS\_BASE\_URL(): string { return 'ws://'+this.STOPGAP\_APP\_HOST+':'+this.WS\_PORT+'/updates'; } | |
|  | |  | |
|  | | public static get WEB\_CONTENT\_URL(): string { return this.BASE\_URL+'/web\_content\_server/'; } | |
|  | | public static get CONTENT\_MANAGEMENT(): string { return this.STOPGAP\_APP\_HOST\_URL+this.WEB\_CONTENT\_PORT+'/contentManagement'; } | |
|  | |  | |
|  | | public static set STOPGAP\_APP\_HOST(name:string) { this.hostName = name;} | |
|  | | public static set STOPGAP\_APP\_PORT(port:string) { this.port = port;} | |
|  | | public static set STOPGAP\_APP\_PROTOCOL(protocol:string) {this.protocol = protocol;} | |
|  | | public static get STOPGAP\_APP\_HOST() : string { return this.hostName; } | |
|  | | public static get STOPGAP\_APP\_PORT() : string { return this.port; } | |
|  | | public static get STOPGAP\_APP\_PROTOCOL() : string { return this.protocol; } | |
|  | |  | |
|  | | public static get STOPGAP\_APP\_HOST\_URL() : string { | |
|  | | return this.STOPGAP\_APP\_PROTOCOL+'//'+this.STOPGAP\_APP\_HOST+':'; | |
|  | | } | |
|  | |  | |
|  | | public static get WS\_PORT() : string { return '8080';} | |
|  | | public static get WEB\_CONTENT\_PORT() : string { return '3004';} | |
|  | | public static get STATIC\_CONTENT\_PORT() : string { return '4000';} | |
|  | |  | |
|  | | //static content server constants | |
|  | | public static get BLUE\_THEME\_URL(): string { return 'http://'+this.hostName+':'+this.STATIC\_CONTENT\_PORT+'/styles/anthem/anthem.blue.theme.css'; } | |
|  | | public static get BLACK\_THEME\_URL(): string { return 'http://'+this.hostName+':'+this.STATIC\_CONTENT\_PORT+'/styles/anthem/anthem.black.theme.css'; } | |
|  | | public static get IMAGES\_URL(): string { return 'http://'+this.hostName+':'+this.STATIC\_CONTENT\_PORT+'/resources/icons/'; }  -- | |

|  |
| --- |
| 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"); |

--

|  |
| --- |
| /\*\* |
|  | \* STEPS to follow with examples. <br> |
|  | \* |
|  | \* <p>EXAMPLES:</p> |
|  | \* 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> |
|  | \* Ex 5: /anthem/acmp/um/case/{cid}/member/address - POST operation to create new address <br> |
|  | \* Ex 6: /anthem/icr/um/case/{cid}/member/address/{aid}?version=n - PUT operation to update a given address. This would be a complete model update <br> |
|  | \* Ex 7: /anthem/icr/um/case/{cid}/member/address/{aid}?version=n - PATCH operation to update a given address. This would be a partial model update <br> |
|  | \* Ex 8: /anthem/icr/member/\_search$validate - POST operation which contains criteria which will be validated and response provided per field <br> |
|  | \* Ex 9: /anthem/hrs/member/\_search$config - GET operation that returns static meta data created for 'member/\_search' process <br> |
|  | \* Ex 10: /anthem/hrs/um/case/{cid}/member/address$validate - POST operation that contains address field populated which will be validated at each field level <br> |
|  | \* Ex 11: /anthem/hrs/address$validate - POST operation |
|  | \* <br> |
|  | \* <br> |
|  | \* <p>STEPS:</p> |
|  | \* 1. Inject a list of available domain-aliases to consider |
|  | \* 2. Inject a list of available actions to consider |
|  | \* 3. Inject any rule to explicitly allow or prohibit a certain combination of: client-code + app-code + /p + domain-alias + action + behavior + RequestMethod.Type |
|  | \* 4. Inject patterns to follow<ol> |
|  | \* <li>/{app}/{domain-alias}/{id}<ul> |
|  | \* <li>a. domain-alias would be identified from injected list</li> |
|  | \* <li>b. {id} would be unrecognized and considered as refId for identified domain-alias</li> |
|  | \* <li>c. If the HTTP method is GET, then action would be "\_get"</li> |
|  | \* <li>d. If the HTTP method is POST, then action would be "\_create" and any RequestBody JSON object would be used</li> |
|  | \* <li>e. For DELETE - '\_delete', PUT - '\_replace', PATCH - '\_update'. All these operations require mandatory 'version' parameter</li></ul></li> |
|  | \* |
|  | \* <li>/{app}/{domain-alias-A}/{domain-alias-B}/{B\_id}/.../{domain-alias-N}/{N\_id}<ul> |
|  | \* <li>a. Detect all domain-aliases from the url and corresponding refId, if available</li> |
|  | \* <li>b. Follow same steps as above in '/{domain-alias}/{id}'</li></ul></li> |
|  | \* |
|  | \* <li>/{app}/{domain-alias}/{\_action} -OR- /{app}/{domain=alias}/{refId}/{\_action}<ul> |
|  | \* <li>a. domain-alias would be identified from injected list</li> |
|  | \* <li>b. action would be identified from injected list</li> |
|  | \* <li>c. Follow same steps as above for recurring domain-alias identification</li></ul></li> |
|  | \* |
|  | \* <li>/{app}/{domain-alias}/{\_action}.{$behavior} OR /{app}/{domain-alias}{$behavior}<ul> |
|  | \* <li></li></ul></li></ol> |
|  | \* |
|  | \* 5. Action can appear only once and at the end of the url but prior to Behavior, if any |
|  | \* 6. Action if explicitly provided, overrides inferred action. Ex: /{app}/{domain-alias}/{id}/\_delete GET : would be treated as '\_delete' action and not '\_get' inferred from GET methods |
|  | \* 7. GET/\_get and DELETE/\_delete only accepts {refId} as argument and not any RequestBody |
|  | \* 8. Behavior can appear only once at the end or url prefixed by '.' Ex: .../../../\_search$config OR ../../\_search$validate |
|  |  |
|  |  |
|  |  |
|  |  |
|  | @RestController |
|  | //@EnableResourceServer |
|  | public class WebActionController { |
|  |  |
|  | public static final String URI\_PATTERN\_P = "/{clientCode}/\*\*/p"; |
|  | public static final String URI\_PATTERN\_P\_OPEN = URI\_PATTERN\_P + "/\*\*"; |
|  |  |
|  | @Autowired WebCommandDispatcher dispatcher; |
|  |  |
|  | @Autowired ExecutionContextLoader ctxLoader; |
|  |  |
|  | @RequestMapping(value=URI\_PATTERN\_P+"/clear", produces="application/json", method=RequestMethod.GET) |
|  | public void clear() { |
|  | ctxLoader.clear(); |
|  | } |
|  |  |
|  | @RequestMapping(value=URI\_PATTERN\_P+"/gc", produces="application/json", method=RequestMethod.GET) |
|  | public void gc() { |
|  | System.gc(); |
|  | System.runFinalization(); |
|  | } |
|  |  |
|  |  |
|  | @RequestMapping(value=URI\_PATTERN\_P\_OPEN, produces="application/json", method=RequestMethod.GET) |
|  | public Object handleGet(HttpServletRequest req, @RequestParam(required=false) String a) { |
|  | return handleInternal(req, RequestMethod.GET, null, a); |
|  | } |
|  |  |
|  | @RequestMapping(value=URI\_PATTERN\_P\_OPEN, produces="application/json", method=RequestMethod.DELETE) |
|  | public Object handleDelete(HttpServletRequest req, @RequestParam String v) { |
|  | return handleInternal(req, RequestMethod.DELETE, v, null); |
|  | } |
|  |  |
|  | @RequestMapping(value=URI\_PATTERN\_P\_OPEN, produces="application/json", method=RequestMethod.POST) |
|  | public Object handlePost(HttpServletRequest req, @RequestBody String json) { |
|  | return handleInternal(req, RequestMethod.POST, null, json); |
|  | } |
|  |  |
|  | @RequestMapping(value=URI\_PATTERN\_P\_OPEN, produces="application/json", method=RequestMethod.PUT) |
|  | public Object handlePut(HttpServletRequest req, @RequestParam String v, @RequestBody String json) { |
|  | return handleInternal(req, RequestMethod.PUT, v, json); |
|  | } |
|  |  |
|  | @RequestMapping(value=URI\_PATTERN\_P\_OPEN, produces="application/json", method=RequestMethod.PATCH) |
|  | public Object handlePatch(HttpServletRequest req, @RequestParam String v, @RequestBody String json) { |
|  | return handleInternal(req, RequestMethod.PATCH, v, json); |
|  | } |
|  |  |
|  | @RequestMapping(value=URI\_PATTERN\_P+"/event/notify", produces="application/json", method=RequestMethod.POST) |
|  | public Object handleEventNotify(HttpServletRequest req, @RequestBody ModelEvent<String> event) { |
|  | Object obj = dispatcher.handle(req, RequestMethod.POST, event); |
|  |  |
|  | Holder<Object> output = new Holder<>(obj); |
|  | return output; |
|  | } |
|  |  |
|  | @RequestMapping({"/login/\*"}) |
|  | public ResponseEntity<?> login(){ |
|  | return ResponseEntity.status(HttpStatus.OK).body(null); |
|  | } |
|  |  |
|  |  |
|  | protected Object handleInternal(HttpServletRequest req, RequestMethod httpMethod, String v, String json) { |
|  | Object obj = dispatcher.handle(req, httpMethod, v, json); |
|  | Holder<Object> output = new Holder<>(obj); |
|  | return output; |
|  | } |

|  |  |
| --- | --- |
|  | public cass WebCommandBuilder { |
|  |  |
|  | private JustLogit logit = new JustLogit(this.getClass()); |
|  |  |
|  | public Command build(HttpServletRequest request) { |
|  | logit.trace(()->"Received http request. "+request.getMethod()+" URI: "+request.getRequestURI()); |
|  |  |
|  | return handleInternal(request.getRequestURI(), request.getParameterMap()); |
|  | } |
|  |  |
|  | public Command build(HttpServletRequest request, ModelEvent<String> event) { |
|  | String uri = request.getRequestURI(); |
|  | logit.info(()->"Received http request. "+request.getMethod()+" URI: "+uri+" with event: "+event); |
|  |  |
|  | final String constructedUri; |
|  | if(event==null) { |
|  | constructedUri = request.getRequestURI(); |
|  | } else { |
|  | String clientUri = StringUtils.substringBefore(uri, Constants.SEPARATOR\_URI\_PLATFORM.code+Constants.SEPARATOR\_URI.code); //separator = /p/ |
|  | constructedUri = clientUri + Constants.SEPARATOR\_URI\_PLATFORM.code + event.getPath() + Constants.SEPARATOR\_URI.code +event.getType(); |
|  |  |
|  | logit.info(()->"Constructed URI: "+constructedUri); |
|  | } |
|  |  |
|  |  |
|  | Command cmd = handleInternal(constructedUri, request.getParameterMap()); |
|  | return cmd;} |
|  |  |
|  |  |
|  | public Command handleInternal(String uri, Map<String, String[]> rParams) { |
|  | Command cmd = CommandBuilder.withUri(uri).addParams(rParams).getCommand(); |
|  | return cmd; |
|  | } |
|  | public class MapsTo { |
|  |  |
|  | public enum Mode { |
|  | UnMapped, |
|  | MappedAttached, |
|  | MappedDetached; |
|  | } |
|  |  |
|  | public enum Nature { |
|  | Default, |
|  | TransientColElem, |
|  | TransientModel; |
|  |  |
|  | public boolean isTransient() { |
|  | return !Default.name().equals(name()); |
|  | } |
|  | } |
|  |  |
|  | @Retention(RetentionPolicy.RUNTIME) |
|  | @Target({ElementType.TYPE}) |
|  | @Model |
|  | @Inherited |
|  | public @interface Mapped { |
|  |  |
|  | } |
|  |  |
|  | @Retention(RetentionPolicy.RUNTIME) |
|  | @Target({ElementType.TYPE}) |
|  | @Mapped |
|  | public @interface Type { |
|  |  |
|  | Class<?> value(); |
|  | } |
|  |  |
|  | public enum LoadState { |
|  | PROVIDED, // Manual loaded such as UI onload from grid or BPM or exec.config |
|  | AUTO; // Framework loaded: from within mapsTo entity or by making cmdGateway call |
|  | } |
|  |  |
|  | @Retention(RetentionPolicy.RUNTIME) |
|  | @Target({ElementType.ANNOTATION\_TYPE}) |
|  | public @interface DetachedState { |
|  | // load state provided (via Grid call, BPM or exec-config) OR via F/w making cmdGateway call |
|  | LoadState loadState() default LoadState.PROVIDED; |
|  |  |
|  | // once state is loaded, should it be cached or fetched each time |
|  | Cache cacheState() default Cache.rep\_none; |
|  |  |
|  | // once state is loaded, should it be managed (edits reflected) by this Quad or treated as ReadOnly |
|  | boolean manageState() default false; |
|  | } |
|  |  |
|  | @Retention(RetentionPolicy.RUNTIME) |
|  | @Target({ElementType.FIELD}) |
|  | public @interface Path { |
|  |  |
|  | String value() default ""; |
|  |  |
|  | boolean linked() default true; |
|  |  |
|  | String colElemPath() default DEFAULT\_COL\_ELEM\_PATH; |
|  |  |
|  | Nature nature() default Nature.Default; |
|  |  |
|  | DetachedState detachedState() default @DetachedState; |
|  | } |
|  |  |
|  | public static Mode getMode(Path path) { |
|  | if(path==null) return Mode.UnMapped; |
|  |  |
|  | return path.linked() ? Mode.MappedAttached : Mode.MappedDetached; |
|  | } |
|  |  |
|  | public static boolean hasCollectionPath(Path path) { |
|  | return StringUtils.trimToNull(path.colElemPath())!=null; |
|  | } |
|  |  |
|  | public static final String DETACHED\_SIMULATED\_FIELD\_NAME = "detachedParam"; |
|  | public static final String DEFAULT\_COL\_ELEM\_PATH = ""; |