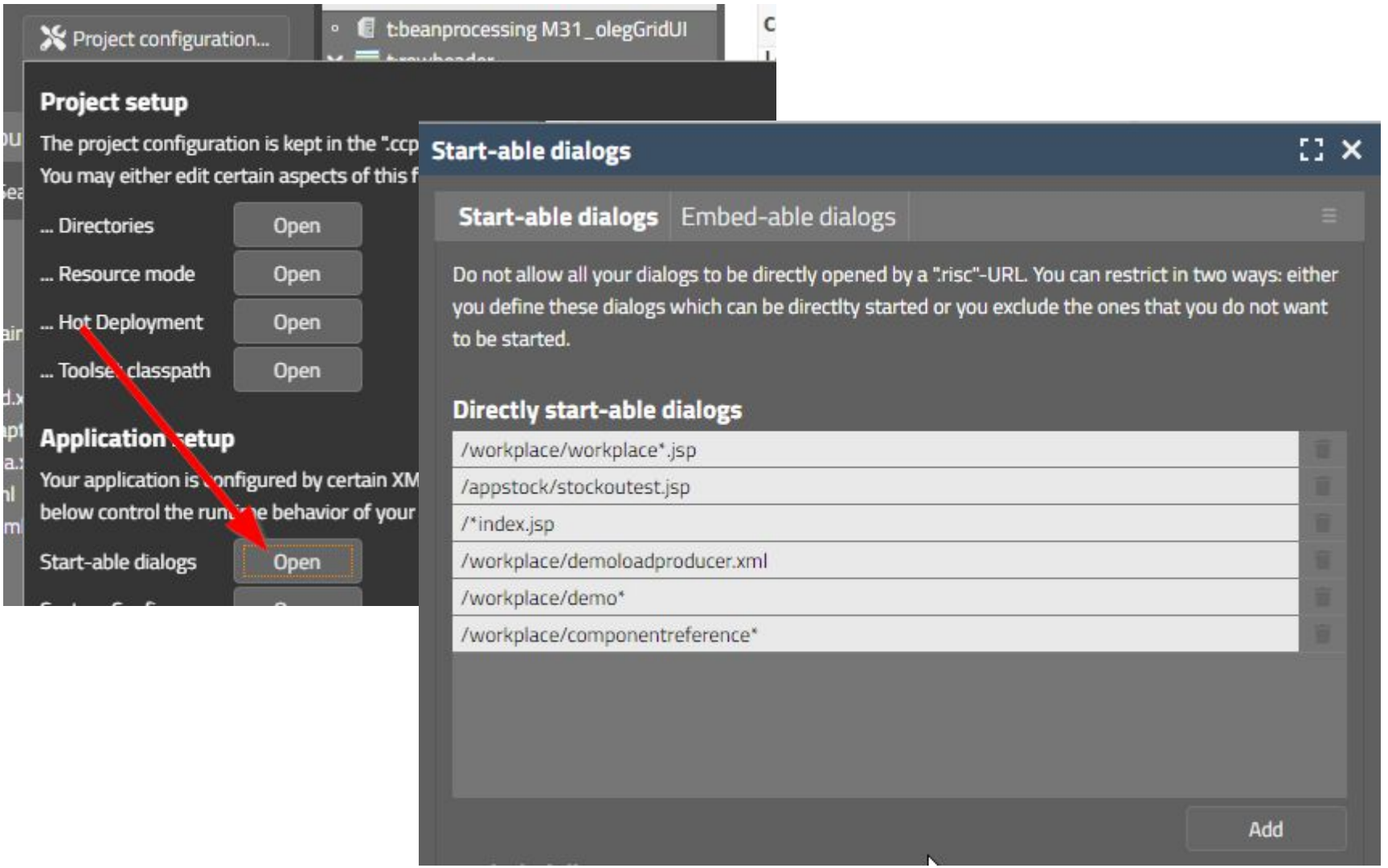
XVII. Community Meeting

CaptainCasa

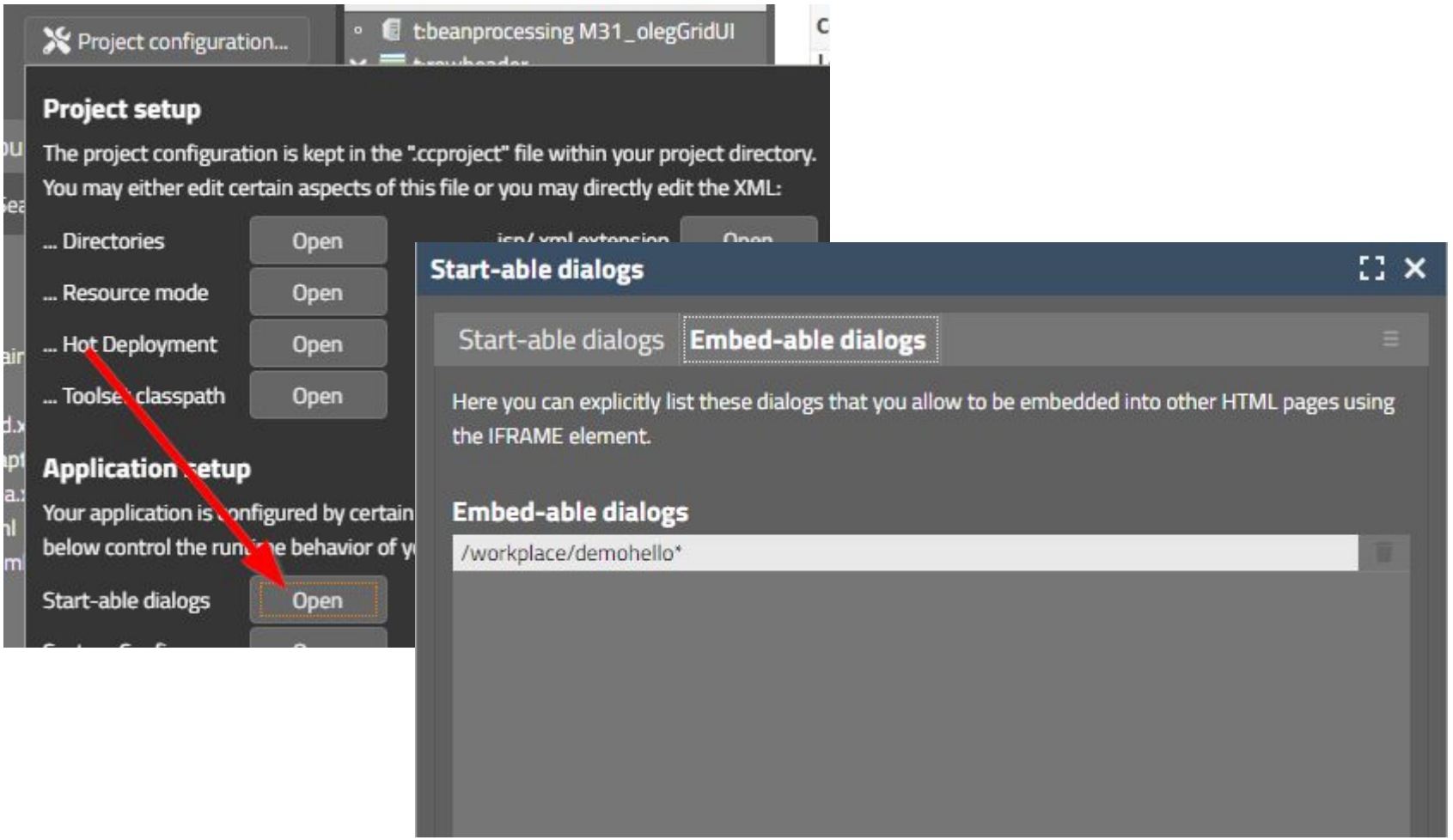
Security

Dedicated starting pages



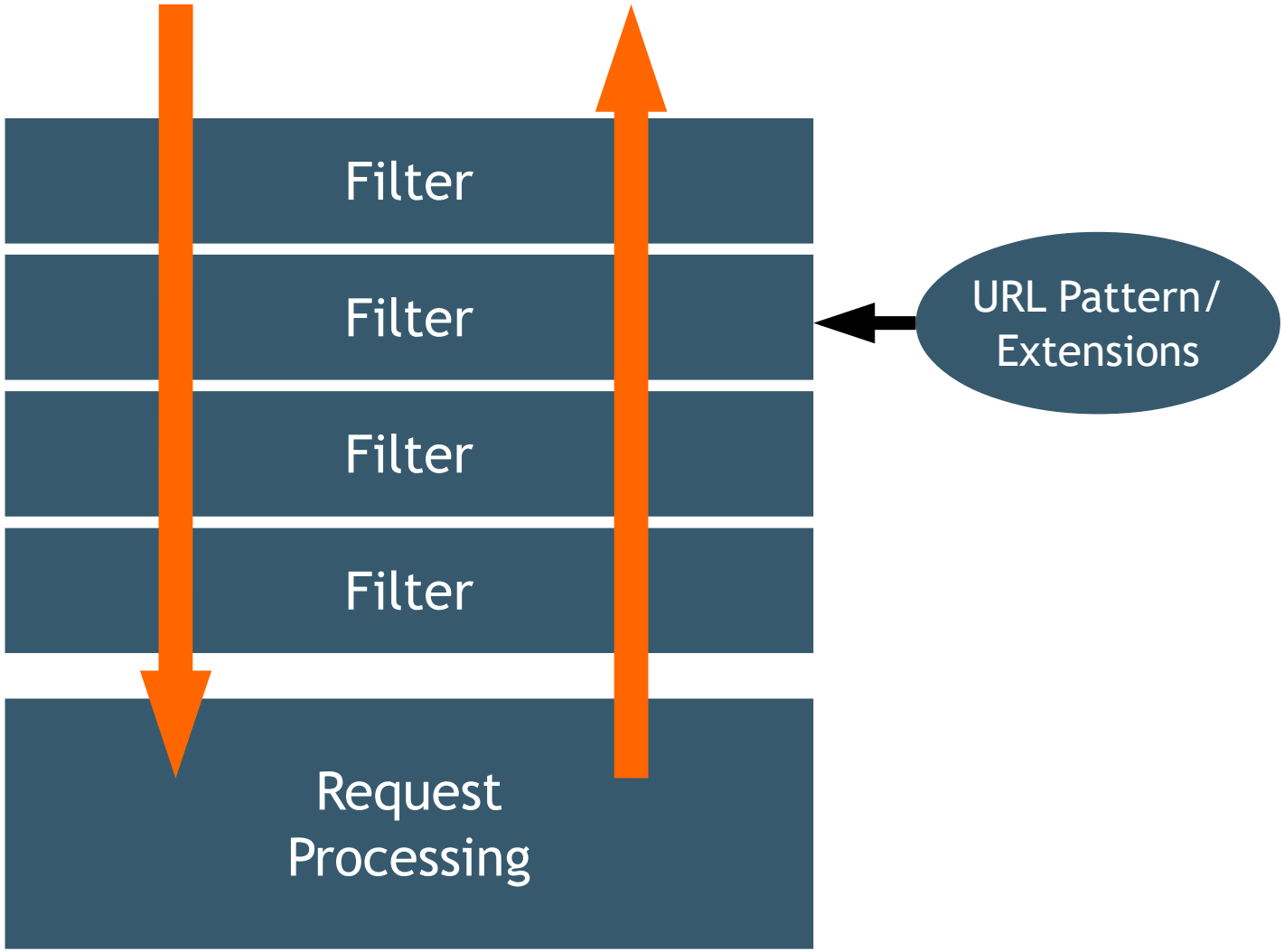
CaptainCasa

Dedicated embed-able pages



CaptainCasa

Filter Management



Filter Management

- Former times, before 2020
  - web.xml

```
<filter>
  <filter-name>org.eclnt.jsfserver.util.CompressionFilter</filter-name>
  <filter-class>org.eclnt.jsfserver.util.CompressionFilter</filter-class>
</filter>
<filter>
  <filter-name>org.eclnt.jsfserver.util.ResourceAccessFilter</filter-name>
  <filter-class>org.eclnt.jsfserver.util.ResourceAccessFilter</filter-class>
</filter>
<filter>
  <filter-name>org.eclnt.jsfserver.util.ThreadingFilter</filter-name>
  <filter-class>org.eclnt.jsfserver.util.ThreadingFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>org.eclnt.jsfserver.util.ThreadContextInitializerFilter</filter-name>
  <url-pattern>*</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>org.eclnt.jsfserver.util.CompressionFilter</filter-name>
  <url-pattern>*.jsp</url-pattern>
</filter-mapping>
<filter-mapping>
```

Filter Management

- Now, >= 2020
  - Configuration by API
  - Central class **CCInitializeServlets**

```
protected void initializeFilters(ServletContext servletContext)
{
    initializeFilter(servletContext, ErrorAnonymizerFilter.class,"/*");
    initializeFilter(servletContext,ThreadContextInitializerFilter.class,"/*");
    initializeFilter(servletContext,SecurityFilterPreflights.class,"/*");
    initializeFilter(servletContext, SecurityFilterURLCheck.class,
            "/faces/*",
            "*.ccbuffer",
            "/ccbuffer/*",
            "*.ccupload",
            "/cctempfileaccess/*"
            );
    initializeFilter(servletContext, NoCacheNoStoreFilter.class,
            "*.jsp",
            "*.risc",
            "*.ccbuffer",
            "/ccbuffer/*",
            "/cctempfileaccess/*"
            );
    initializeFilter(servletContext, HttpHeaderAttributesForPagesFilter.class, "*.risc","*.html");
    initializeFilter(servletContext, SameOriginFilterForHtml.class, "/eclnt/risc/*");
    initializeFilter(servletContext, CacheFilter.class, "*.js", "*.css");
    initializeFilter(servletContext, CompressionFilter.class, "*.jsp", "*.xml", "*.css", "*.js", "*.ttf", "*.i18n", "*.json");
    initializeFilter(servletContext, ResponseLoggerFilter.class, "/*");
    initializeFilter(servletContext, SecurityFilterRemoteAddress.class, "/*");
    initializeFilter(servletContext, SecurityFilter.class, "*.jsp", "*.ccupload", "*.ccinvalidatesession");
    initializeFilter(servletContext, SecurityFilterGeneral.class
```

Filter Management

- CaptainCasa by default activates new filters with reasonable default values
  - system.xml allows to switch off/re-configure filters

```
<filterconfiguration
    active="false"
    classname="...classNameOfFilter..."
/>
```

```
<filterconfiguration
    active="true"
    classname="...classNameOfFilter..."
    additionalmappings="...mapping...;...mapping...;...mapping..."
/>
```

Filter „ErrorAnonymizerFilter“

- Catches any error in the request processing
  - Hides the details of the error
    - Stack trace of cause
  - Makes sure the details are written to the log
  - Throws an anonymized error
- Reason
  - Attackers must not see details of the system layering
    - e.g. which persistence management is used

Filter „SecurityFilterPreflights“

```
String v = httpRequest.getHeader("Access-Control-Request-Private-Network");
if ("true".equals(v))
{
    httpResponse.setHeader("Access-Control-Request-Private-Network","true");
}
```

Introduction

Chrome is deprecating direct access to private network endpoints from public websites as part of the Private Network Access (PNA) specification.

Chrome will start sending a CORS preflight request ahead of any private network request for a subresource, which asks for explicit permission from the target server. This preflight request will carry a new header, `Access-Control-Request-Private-Network: true`, and the response to it must carry a corresponding header, `Access-Control-Allow-Private-Network: true`.

The aim is to protect users from cross-site request forgery (CSRF) attacks targeting routers and other devices on private networks. These attacks have affected hundreds of thousands of users, allowing attackers to redirect them to malicious servers.

Rollout plan

Filter „SecurityFilterURLCheck“

- Checks the URL for occurance of
  - „../“
  - „..\“
- Reason
  - Attacker must not be allowed to access „side areas“

Filter „NoCacheNoStoreFilter“

- Disables any type of client side caching
- Reason
  - Sensitive data can be contained e.g. in cached PDF documents
  - Clients might be run in an environment where attackers can access the cache
    - Internet cafe
    - Workplace...!

Filter „CacheFilter“

- Sets caching to „eternal“
  - Must only be used for non-application content
    - JavaScript
    - CSS
  - Must only be used if there is some mechanism to force refreshing of cache

Filter „HttpHeaderAttributesForPagesFilter“

- Sets certain http header attributes
  - x-xss-protection 1;mode=block
  - x-content-type-options nosniff
  - content-security-policy ...URL sandbox...
  - referrer-policy same-origin

- Configuration by system.xml

```
<httpheaderattributesforpages
    x-xss-protection="1; mode=block"
    x-content-type-options="nosniff"
    content-security-policy="default-src 'self' data: 'unsafe
    referrer-policy="no-referrer"
    />
```

Filter „HttpHeaderAttributesForPagesFilter“

- content-security-policy
  - ...URL sandbox...

```
default-src 'self' data: 'unsafe-inline' 'unsafe-eval'; img-src * data:
```

```
default-src 'self' www.captaincasa.com www.youtube.com
data: 'unsafe-inline' 'unsafe-eval'; img-src * data:
```

Filter „SecurityFilterRemoteAddress“

- Checks if the client changes during a session
  - Session started at Chrome browser „on the left“ but suddenly continued at Firefox browser „on the right“
- Three ways
  - httpRequest.getRemoteAddr();
  - http header: x-forwarded-for
  - http header: user-agent
- Reason
  - Detect session-hijacking

Filter „SecurityFilterGeneral“

- Adds an internally generated id per http-session as COOKIE
  - Requests are only processed if the COOKIE-id fits to the session
- Reason
  - Prevent session-hijacking
  - Only relevant for session management by URL
  - Hijacking the „jsessionid“ is not enough to get into the session – also the (encrypted!) COOKIE-id must be correct

Filter „SameOriginFilterForHtml“

- Sets http-header for „.html“
  - X-Frame-Options sameorigin
- Reason
  - Ensures that IFRAME-included pages from same server are processed
    - PDF document shown as IFRAME
  - In particular: integration of sub-frameworks
    - chart.js
    - osm
    - ...

Resource access by class loader

- Only for dedicated extensions
  - switch on/off by system.xml
  - interface „IResourceSecurityChecker“

```
public final static Set<String> s_extensionsWhiteList = new
public final static Set<IResourceSecurityChecker> s_security

static
{
    s_extensionsWhiteList.add("bmp");
    s_extensionsWhiteList.add("css");
    s_extensionsWhiteList.add("eot");  // font info
    s_extensionsWhiteList.add("gif");
    s_extensionsWhiteList.add("giff");
    s_extensionsWhiteList.add("htm");
    s_extensionsWhiteList.add("html");
    s_extensionsWhiteList.add("jar");  // client libs for Sw
    s_extensionsWhiteList.add("jpg");
    s_extensionsWhiteList.add("jpeg");
    s_extensionsWhiteList.add("js");
    s_extensionsWhiteList.add("json");
    s_extensionsWhiteList.add("map"); // js.map ⇒ *.js.map
    s_extensionsWhiteList.add("pdf");
    s_extensionsWhiteList.add("png");
    s_extensionsWhiteList.add("svg");
    s_extensionsWhiteList.add("tif");
    s_extensionsWhiteList.add("tiff");
    s_extensionsWhiteList.add("ttf");   // font info
    s_extensionsWhiteList.add("txt");
    s_extensionsWhiteList.add("woff");  // font info
    s_extensionsWhiteList.add("woff2"); // font info
}
```

FIXGRID exports

```
public interface IFIXGRIDExportSanitizer
{
    public void sanitizeStringDataBeforeExport
        (ENUMExportType exportType, List<List<String>> data);
    public void sanitizeValueDataBeforeExport
        (ENUMExportType exportType, List<List<FormattedValue>> data);
}
```

```
<fixgrid
        ...
        exportsanitizer="...className of implementation..."
        ...
/>
```