Better fit your users interests

Exploiting the Audience Targeting API

Luca Comin, Software Architect SMC

Di cosa discuteremo

Scoprire Audience Targeting e sfruttarne le potenzialità:

- Introduzione ad Audience Targeting
- Focus su Score Points
- Una nuova funzione
- L’implementazione

“Liferay’s Audience Targeting app allows you to divide your audience into user segments and target specific content to different segments”

Audience Targeting

L’utente naviga il sito

Le preferenze vengono registrate

Nuovi contenuti vengono proposti

Vantaggi

- Arrichire l’esperienza dell’utente rendendola più personale
- Aumentare interesse e coinvolgimento verso i contenuti
- Aumentare il numero e la qualità dei Lead

Core Features

- Segmentazione utenti
- Associazione Asset ai Segmenti
- Le Campagne
- Tracking delle azioni
- Reports

1

Definire degli insiemi di interesse

I Segmenti

2

Associare Contenuti ai Segmenti

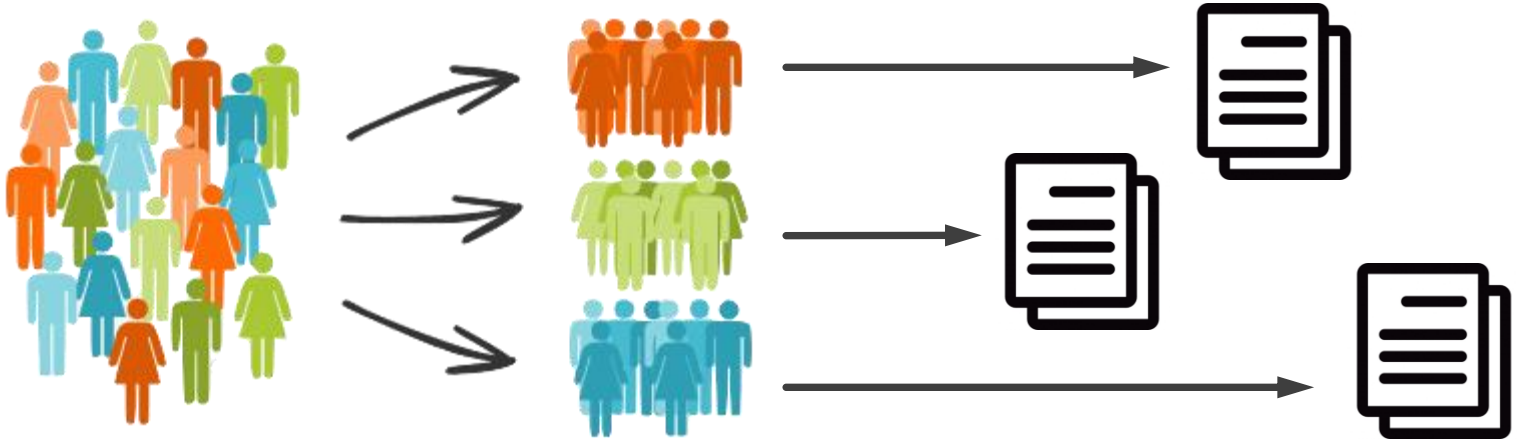
3

Veicolare i contenuti agli utenti in base all’interesse

Come funziona la segmentazione?

- Ogni Segmento è un AssetCategory
- I segmenti appartengono all'AssetVocabulary Segmento Utente
- I contenuti vengono categorizzati
- Vengono quindi sfruttati i framework base Liferay di Asset e Categorie

Come funziona la segmentazione?



Ad ogni Segmento è associata una categoria, ad ogni categoria diversi Asset

Quali contenuti?

- Tutti i Model che sono anche Asset
- Documenti, Contenuti Web, Blog etc…
- Contenuti personalizzati: nuovi Tipi

Meccanismo di segmentazione

- Ogni utente può essere associato ad uno o più Segmenti (in ogni momento)
- Vale anche per gli utenti Anonimi / Guest
- L’associazione avviene soddisfacendo le “Rules”
- Ogni Segmento può possedere più Rule di accesso

Le Rule

- Sono le regole di accesso ad un Segmento
- Sono @Component OSGi
- Implementano l’interfaccia Rule
- Ogni Rule deve implementare il metodo evaluate
- E’ possibile estendere BaseJSPRule e BaseFreemarkerRule

Le Rule out of the box

Comportamento

- ContentVisitedRule
- PageVisitedRule
- PreviousVisitedSiteRule

Sessione

- BrowserRule
- LanguageRule
- IPGeocodeRule
- TimeRule

Le Rule out of the box

Attributi Utente

- AgeRule
- CustomFieldRule
- RegularRoleRule
- UserLoggedRule

Social

- FacebookAgeRule
- FacebookCityRule
- FacebookFriendsRule
- FacebookLikeRule

Gli utenti Anonimi

- Audience Targeting associa un cookie ad ogni Guest
- Le sessioni utente sono riconosciute attraverso il cookie ANONYMOUS_USER_ID
- Alla cancellazione del cookie il targeting è resettato
- Non tutte le Rule hanno effetto su Guest

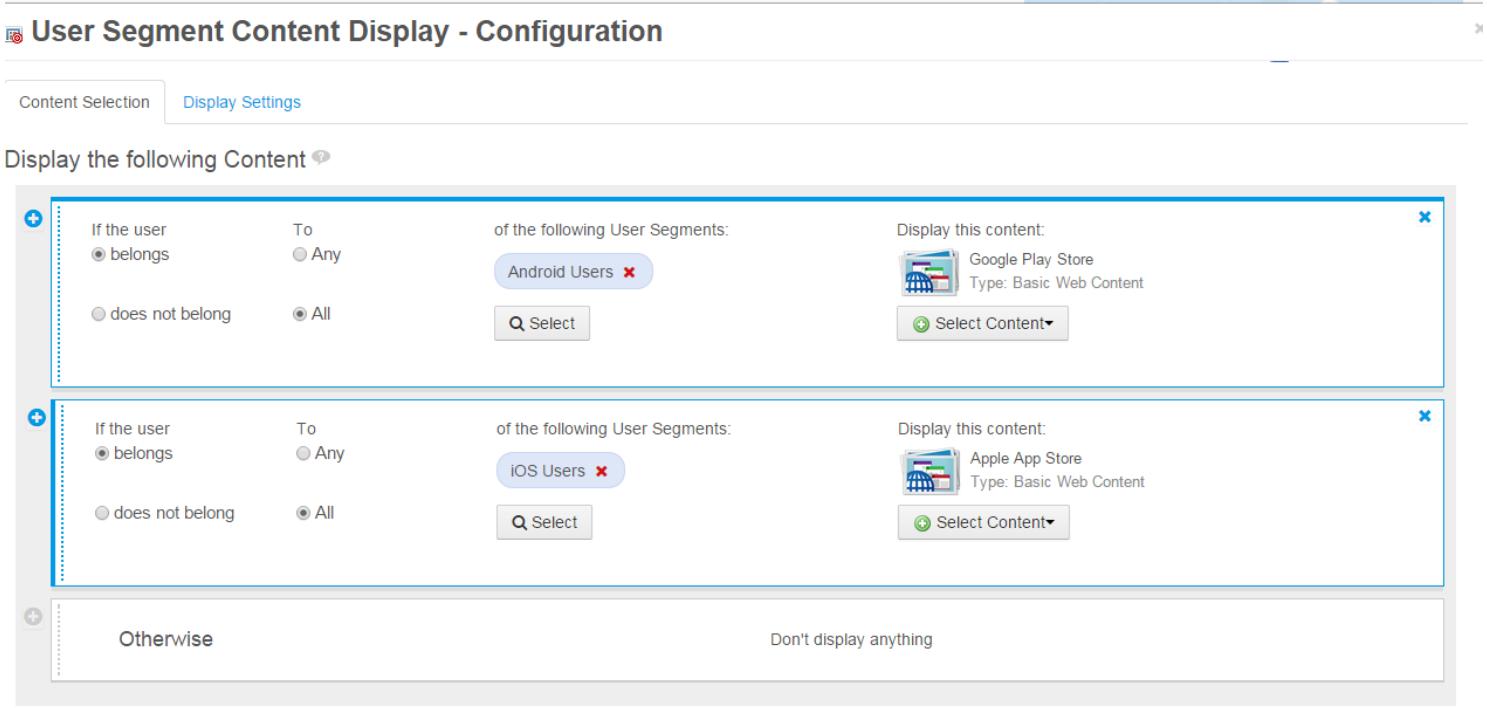
Gli Score Points

- Funzione base di Audience Targeting
- Permette di accumulare punti alla visita di un contenuto segmentato
- E’ disponibile una Rule che valuta il punteggio totale
- Se il totale supera la soglia l’utente entra nel segmento

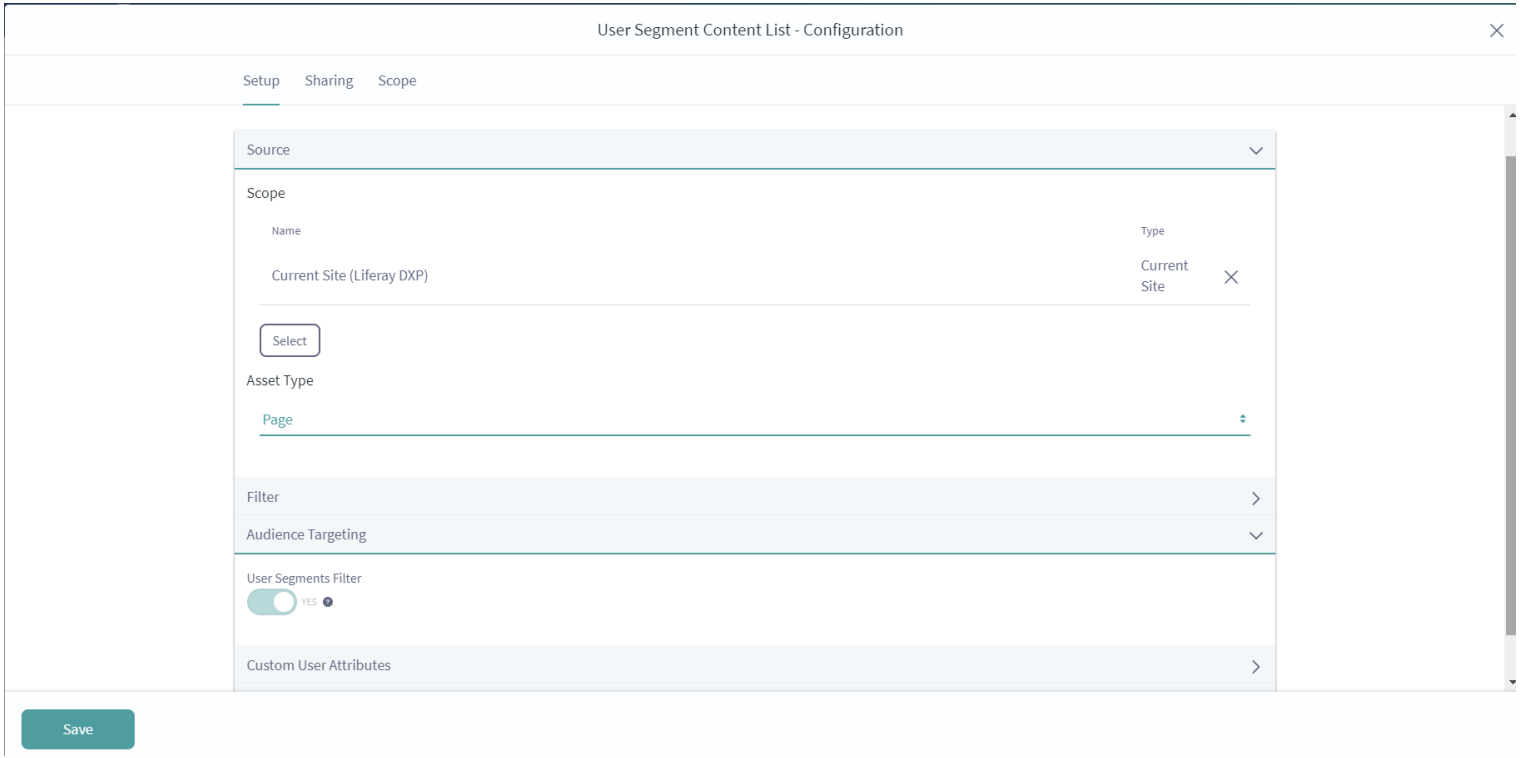
Applicativi aggiuntivi

- User Segment Content Display
- User Segment Content List
- Campaign Content Display

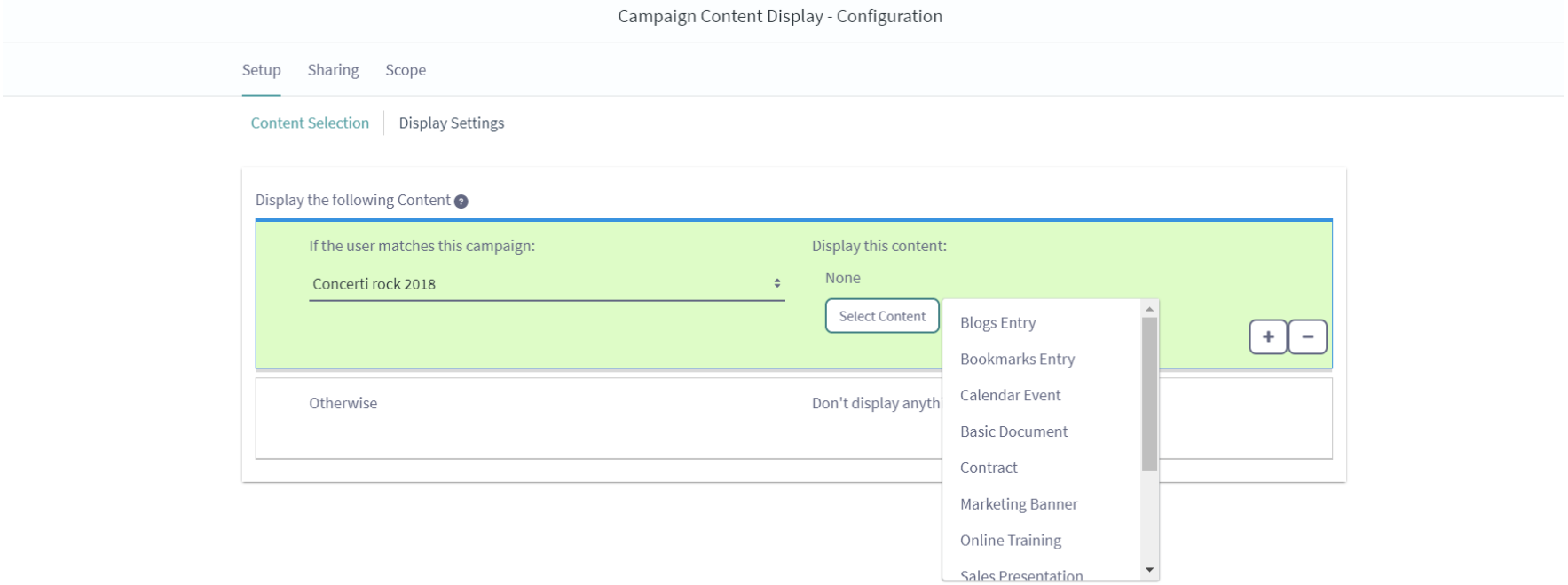
User Segment Content Display



User Segment Content List



Campaign Content Display



Una nuova funzione

Estensione del core

1. Selezione automatica dei contenuti
2. Ordinamento automatico per Segmento
3. Ordinamento automatico agli Asset
4. Distinzione tra i contenuti dei vari Segmenti

Calcolo totali sulle visite

Segmentazione utenti

Ordinamento automatico dei contenuti

In dettaglio

- Ad ogni visita un utente acquisisce un punto
- Un Asset Publisher ordina automaticamente i Segmenti in base al totale
- A parità di segmento i contenuti sono poi ordinati per priorità

L’implementazione

1

Le pagine diventano Asset

2

Nuova ricerca ordinata per Score Points

3

Nuovo ADT per Asset Publisher

Trasformare i Layout in Asset

- Nuova classe CustomLayoutAssetRendererFactory
- Nuovo @Component su service AssetRendererFactory
- Necessaria per istanziare un nuovo LayoutAssetRenderer

Custom vs Base

```
@Component(
    immediate = true,
    property = {
        "javax.portlet.name=" + LayoutAdminPortletKeys.LAYOUT_ADMIN,
        "service.ranking:Integer=200"
    },
    service = AssetRendererFactory.class
)
public class CustomLayoutAssetRendererFactory
    extends BaseAssetRendererFactory<Layout> {

    public static final String TYPE = "layout";

    public CustomLayoutAssetRendererFactory() {
        setClassName(Layout.class.getName());
        setSelectable(true);
        setSearchable(true);
        setLinkable(true);
        setPortletId(LayoutAdminPortletKeys.LAYOUT_ADMIN);
    }
```

```
@Component(
    immediate = true,
    property = {"javax.portlet.name=" + LayoutAdminPortletKeys.LAYOUT_ADMIN},
    service = AssetRendererFactory.class
)
public class LayoutAssetRendererFactory
    extends BaseAssetRendererFactory<Layout> {

    public static final String TYPE = "layout";

    public LayoutAssetRendererFactory() {
        setClassName(Layout.class.getName());
        setSelectable(false);
        setPortletId(LayoutAdminPortletKeys.LAYOUT_ADMIN);
    }
```

Vantaggi della nuova Factory

- Le Pagine diventano un Asset selezionabile
- Le Pagine diventano un Asset relazionabile
- Le Pagine diventano un Asset ricercabile

Nuovo LayoutAssetRenderer

- Permette di modificare
  - Summary delle Pagine
  - Jsp full_content
  - Jsp abstract

Nuovo LayoutIndexer

- Permette di indicizzare le Pagine
- Permette di effettuare query di ricerca sulle Pagine
- Attraverso il metodo postProcessSearchQuery
  - Ordina le Pagine per Segmento (Score Points)
  - Ordina le Pagine per priorità

LayoutIndexer

```java
@Component(immediate = true, service = Indexer.class)
public class LayoutIndexer extends BaseIndexer<Layout> {

    public static final String CLASS_NAME = Layout.class.getName();

    public LayoutIndexer() {
        setDefaultSelectedFieldNames(
            Field.ASSET_TAG_NAMES, Field.COMPANY_ID, Field.ENTRY_CLASS_NAME,
            Field.ENTRY_CLASS_PK, Field.GROUP_ID, Field.TITLE,
            Field.MODIFIED_DATE, Field.SCOPE_GROUP_ID, Field.UID,
            Field.USER_ID);
        setDefaultSelectedLocalizedFieldNames(Field.NAME);
        setFilterSearch(true);
        setPermissionAware(true);
        setSelectAllLocales(true);
    }
```

```java
@Override
public void postProcessSearchQuery(
        BooleanQuery searchQuery, BooleanFilter fullQueryBooleanFilter,
        SearchContext searchContext)
    throws Exception {

    PostProcessUtil.setOrderBySegmentScoreAndPriority(
        searchQuery, searchContext);

    super.postProcessSearchQuery(
        searchQuery, fullQueryBooleanFilter, searchContext);
}
```

Per ogni tipo di Asset indicizzato deve essere personalizzato il metodo postProcessSearchQuery

Logica di ordinamento

- Recupero di AnonymousUser con *AnonymousUserLocalServiceUtil*
- Recupero dei Segmenti Utente con AnonymousUserUserSegmentLocalServiceUtil
- Recupero dei punteggi attraverso

  ScorePointLocalServiceUtil.getPoints(anonymousUserId, userSegmentId)

Logica di ordinamento

```
if (segmentScore.size() > 0) {
    Set<Long> sortedUserSegments = segmentScore.keySet();

    Object[] assetCategoryIds = sortedUserSegments.toArray();

    if (orderType.equalsIgnoreCase("DESC")) {
        ArrayUtil.reverse(assetCategoryIds);
    }

    for (int i = 0 ; i < assetCategoryIds.length ; i++) {
        long assetCategoryId = (long) assetCategoryIds[i];

        TermQueryImpl termQuery = new TermQueryImpl(
            "assetCategoryIds", String.valueOf(assetCategoryId));

        float boost = assetCategoryIds.length - i;

        termQuery.setBoost(boost);

        searchQuery.add(termQuery, BooleanClauseOccur.SHOULD);
    }

    Sort scoreSort = SortFactoryUtil.create("_score", false);
    Sort prioritySort = SortFactoryUtil.create(
        "priority_sortable", true);

    searchContext.setSorts(scoreSort, prioritySort);
}
```

- Segmenti ordinati per Score Points in senso decrescente
- Per ogni Segmento viene composta una TermQuery con boost decrescente
- Le query risultanti sono aggregate in modalità should
- Viene impostato il sorting dei risultati secondo _score e priority_sortable

Presentazione

- ADT per Asset Publisher
- Modifica a jsp di Asset Publisher

Presentazione - ADT

- Scorrimento dei risultati ottenuti da classe Indexer
- Visualizzazione personalizzata dei risultati

Presentazione - ADT

```
<#assign LayoutLocalService = serviceLocator.findService("com.liferay.portal.kernel.service.LayoutLocalService")>
<#assign layout_custom_field_key_navigation_description = themeDisplay.getThemeSetting("layout-custom-field-key-navigation-description")!"" />

<#if entries?has_content>
    <div class="custom-wrapper recommended-pages">
        <div class="recommended-pages-container row">
            <#assign element_class = "col-md-4">
            <#if entries?size == 1>
                <#assign element_class = "col-md-12">
            </#if>
            <#if entries?size == 2>
                <#assign element_class = "col-md-6">
            </#if>
            <#list entries as curEntry>
                <#if curEntry?index == 3>
                    <#break>
                </#if>

                <#assign layout = LayoutLocalService.getLayout(curEntry.getClassPK())>

                <#if curEntry?index == 3>
                    <#break>
                </#if>
                <div class=" recommended-pages-item ${element_class}">
                    <div class="recommended-pages-item-cover" style='background-image:url("/image/layout_icon?img_id=${layout.getIconImageId()}")'>
                    </div>
                    <div class="recommended-pages-item-content custom-wrapper custom-wrapper-info">
                        <h5>${curEntry.getTitle(locale)}</h5>
                        <p>
                            <#if layout_custom_field_key_navigation_description?? && layout_custom_field_key_navigation_description != "" &&
                            layout.getExpandoBridge().getAttribute(layout_custom_field_key_navigation_description)?? && layout.getExpandoBridge().getAttribute(
                            layout_custom_field_key_navigation_description) != "">
                                ${layout.getExpandoBridge().getAttribute(layout_custom_field_key_navigation_description)}
                            </#if>
                        </p>
                        <a href="${layout.getFriendlyURL()}" class="pull-right">Discover &gt;</a>
                        <div style="clear:both"></div>
                    </div>
                </div>
            </#list>
        </div>
    </div>
</#if>
```

Presentazione - JSP

- Override di configuration_dynamic.jsp
- Aggiunta opzione di ordinamento user-segment-score
- Override di view_dynamic_list.jspf
- Aggiunti attributi per la classe Indexer

Presentazione - JSP

configuration_dynamic

```
<aui:select inlineField="<%= true %>" inlineLabel="left" label="order-by" name="pre
    <c:if test="<%= assetPublisherDisplayContext.isOrderingByTitleEnabled() %>">
        <aui:option label="title" />
    </c:if>

    <aui:option label="create-date" value="createDate" />
    <aui:option label="modified-date" value="modifiedDate" />
    <aui:option label="publish-date" value="publishDate" />
    <aui:option label="expiration-date" value="expirationDate" />
    <aui:option label="priority" value="priority" />
    <aui:option label="user-segment-score" value="userSegmentScore" />

    <c:if test="<%= !AssetPublisherWebConfigurationValues.SEARCH_WITH_INDEX %>">
        <aui:option label="view-count" value="viewCount" />
        <aui:option label="ratings" value="ratings" />
    </c:if>
</aui:select>
```

view_dynamic_list

```
if (assetPublisherDisplayContext.getOrderByColumn1().equals("userSegmentScore")) {
    String anonymousUserId = CookieKeys.getCookie(request, "ANONYMOUS_USER_ID");

    attributes.put("orderByUserSegmentScore", true);
    attributes.put("orderTypeByUserSegmentScore", assetPublisherDisplayContext.getOrderByType1());
    attributes.put("anonymousUserId", anonymousUserId);
}
```

Demo

Contatti


luca.comin@smc.it


@comLuke


ComLuke

Rate My Session!

Download the Liferay
Events Mobile App Today

Download on the App Store

ANDROID APP ON Google play