Week 6: Additional Topics

Unit 2: Multitenant Data Isolation

Run Simple

openSAP
open.sap.com

Multitenant Data Isolation

Demo: make your application tenant-aware

Multitenant Data Isolation

Make your application tenant-aware – Part 1

How to fetch tenant GUID from the JWT token ("zid")

```
String tenantId =  SecurityContext.getUserInfo().getIdentityZone();
```

Initialize log context with tenant

```
import static  com.sap.hcp.cf.logging.common.customfields.CustomField.customField;
...
logger.info("Log tenant info with custom field {}", customField("tenant", tenantId"));
```

Apply tenant to the downstream service routes

```
String uriTemplate = "https://{tenant}-approuter.mydomain/ads/api/v1/ads/";
URI url = new UriTemplate(uriTemplate).expand(tenant);
HttpEntity<String> request = new HttpEntity<>("My Content");
ResponseEntity<String> response = restTemplate.exchange(url,  HttpMethod.GET, request, String.class);
```

Multitenant Data Isolation

Make your application tenant-aware – Part 2

Example TenantContext

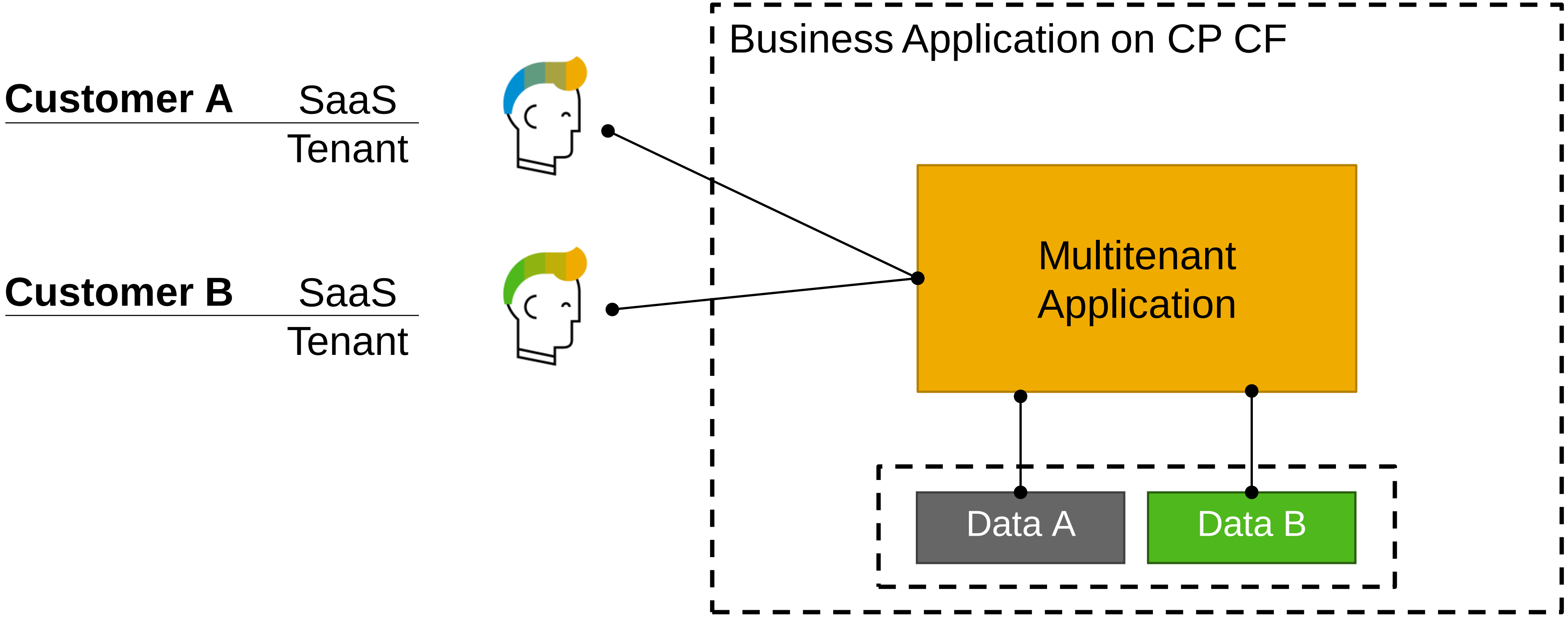
```
public class TenantContext {
    final static String DEFAULT_TENANT = "public";

    // gives each thread its own local variable
    private static ThreadLocal<String> currentTenant = ThreadLocal.withInitial(() -> DEFAULT_TENANT);

    public static String getCurrentTenant() {
        return currentTenant.get(); // provides a dedicated instance
    }
    // should only be accessed once per request (or within test)
    static void setCurrentTenant(String tenantId) {
        currentTenant.set(tenantId);
    }
    static void clear() {
        currentTenant.remove(); // avoid memory leaks: release it for garbage collector
    }
}
```

For each request, initialize TenantContext as part of a HandlerInterceptor or ServletFilter

Multitenant Data Isolation

Data isolation

One common question raised when dealing with SaaS architecture is whether you need an isolated database for every customer or whether there should be a shared database.



Multitenant Data Isolation

Data isolation – Aspects and degrees

Degrees of (data) isolation

- **Shared schema:** Tables contains **tenant discriminator column** (e.g. MANDT)
- **Separate schemas:** Each tenant has its own **database schema (or HDI container)** but shares a DB (recommended approach)
- **Separate databases (premium approach):** Each tenant has its own separate database (by process / physical) to provide “bad neighborhood protection” in terms of data and failure isolation, resource consumption, performance (cache, I/O,…) etc.
  In SAP HANA, this would be a separate **Multitenant Database Container** (MDC, logical database) per customer.

Aspects to be considered: Customer requirements for data security/isolation, performance guarantees, extensibility, reliability, disaster recovery (backup and restore),…

Multitenant Data Isolation

Demo: tenant discriminator column (EclipseLink) vs. tenant discriminator schema (Hibernate)

Multitenant Data Isolation

Tenant discriminator column

Example using EclipseLink

```
import static org.eclipse.persistence.annotations.MultitenantType.*;

@Entity
@Table(name = "advertisements")
@Multitenant(SINGLE_TABLE)
@TenantDiscriminatorColumns({
    @TenantDiscriminatorColumn(name = "TENANT_ID“, length = 36),
    @TenantDiscriminatorColumn(name = "TENANT_CODE")
})
public class Advertisement {
}
```

➢ Provide the tenant ID to the entity manager when accessing data for a multitenant entity.

Multitenant Data Isolation

Tenant discriminator schema

Hibernate natively supports schema-based multitenancy e.g. in conjunction with PostgreSQL. It requires three main components:

- **Configuration**
  Wiring up Hibernate correctly
- **CurrentTenantIdentifierResolver**
  Class responsible for resolving the correct tenant, map tenant to db schema
- **MultiTenantConnectionProvider**
  Class responsible for providing and closing tenant connections

Reference

- Tutorial: Spring-boot Schema based multi tenancy

Multitenant Data Isolation

Further reading

i Additional Material

- EclipseLink User Guide: Multitenancy Annotations in JPA
- Hibernate Documentation: Multi-tenancy in Hibernate
- Tutorial: Spring-boot Schema based multi tenancy

Multitenant Data Isolation

What you’ve learned in this unit

- How to handle multitenant data isolation
  - Shared schema (and tenant discriminator column)
  - Separate schemas
  - Separate databases

Thank you.

Contact information:

open@sap.com

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.