Proposed Cosys Multi-Tenant Architecture

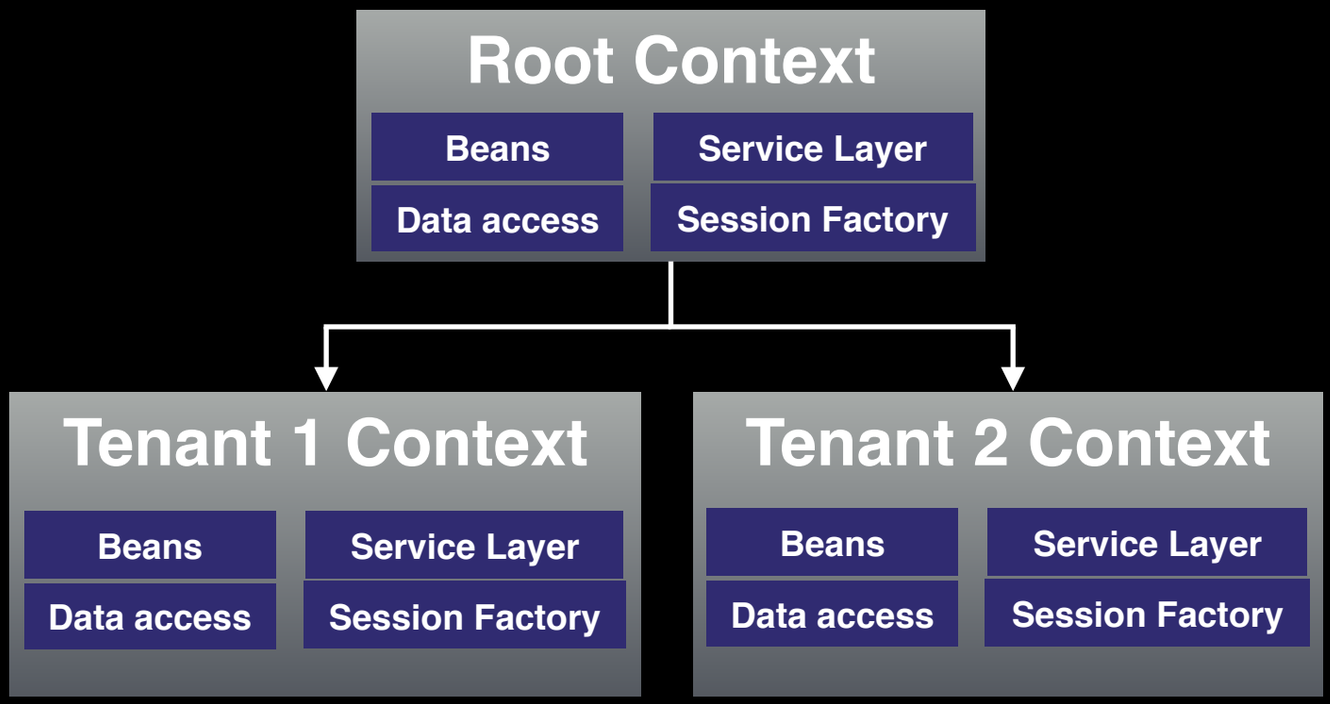
Solution

- UI customisations through CSS and/or add on pages
- Leverage Spring Framework to allow tenants to:
  - Override default system behaviour
  - Extend the default system behaviour
  - Isolate data through tenant specific SessionFactory

Strategy

- Each tenant will be have an isolated schema
- Tenant identification will be through subdomain or request header

Application Contexts

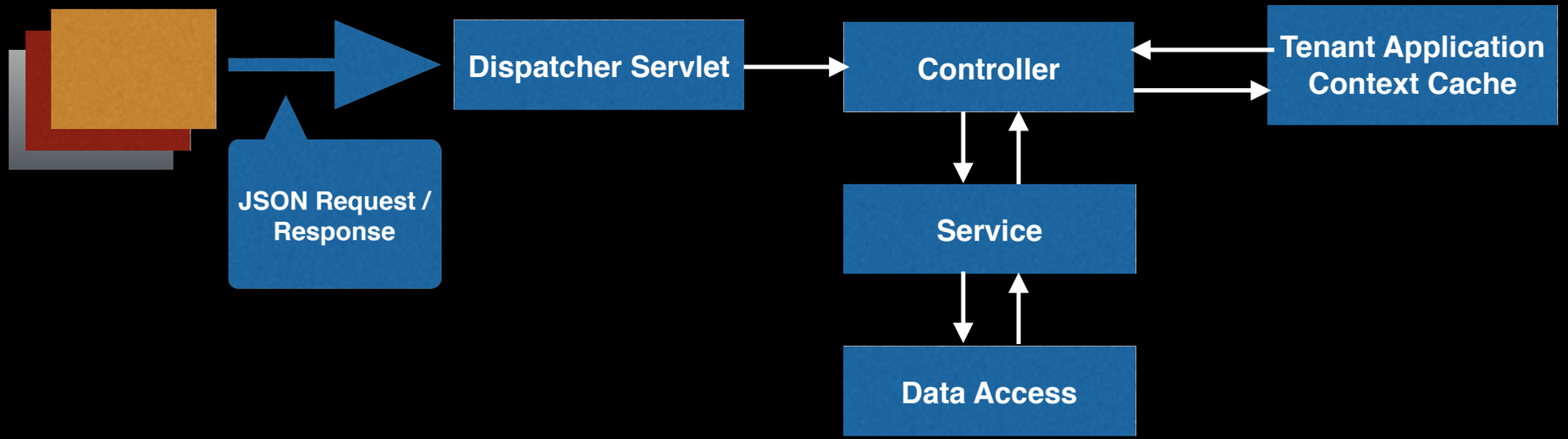



```
public class TenantContextBuilder {

    public void buildTenantContext(AnnotationConfigEmbeddedWebApplicationContext parentApplicationContext) {
        AnnotationConfigApplicationContext hkAppContext = new AnnotationConfigApplicationContext();
        // allow tenant context access to parent base context
        hkAppContext.setParent(parentApplicationContext);
        hkAppContext.register(HkTenant.class);
        hkAppContext.register(HkMysqlJpaConfiguration.class);
        hkAppContext.refresh();
        TenantContextHolder.addTenantContext(HkTenant.class.getSimpleName(), hkAppContext);

        AnnotationConfigApplicationContext sgAppContext = new AnnotationConfigApplicationContext();
        sgAppContext.setParent(parentApplicationContext);
        sgAppContext.register(SgTenant.class);
        sgAppContext.register(SgMysqlJpaConfiguration.class);
        sgAppContext.refresh();
        TenantContextHolder.addTenantContext(SgTenant.class.getSimpleName(), sgAppContext);

    }

}
```

Application Context Properties

- Tenant specific application contexts are initialised after the context has been initialised
- Tenant contexts can override a service from the root context by simply defining the same name
- Tenant context will have access to root contexts but not vice versa
- Tenant context are cached in a map

Request Handling




```
@RestController
public class SalesOrderController {

    @RequestMapping(value = "/cosys/v1/addorder/{tenant}", method = RequestMethod.GET)
    public HttpEntity<Message> addOrder(@PathVariable("tenant") String tenant) {
        ApplicationContext appContext = TenantContextHolder.getTenantContext(tenant);
        OrderServices orderServices = (OrderServices) appContext.getBean("orderServices");
        SalesOrder salesOrder = orderServices.addOrder();

        Message message = new Message();
        message.setMessage("Added Order ref : " + salesOrder.getReferenceId());
        return new ResponseEntity<Message>(message, HttpStatus.OK);
    }

}
```

Limitation

- Server restart is required when new tenant is added
- Session Factory Overhead ~ 5 - 10 MB / tenant