Liferay portal modern architecting and development

MODULARITY PATTERNS USING OSGI

Rafik HARABI

Who am I ?

- Software Architect and Liferay Specialist.
- Building portal using Liferay since 2009 (more then 15 portals).


rafik.harabi@innovsquare.com


@innovsquare


https://github.com/innovsquare


linkedin.com/rafik.harabi

Who are you ?

- Before we get started...
  - We are building portals at work,
  - We are building Portal using Liferay,
  - We have heard about Liferay Portal and we want to learn more.

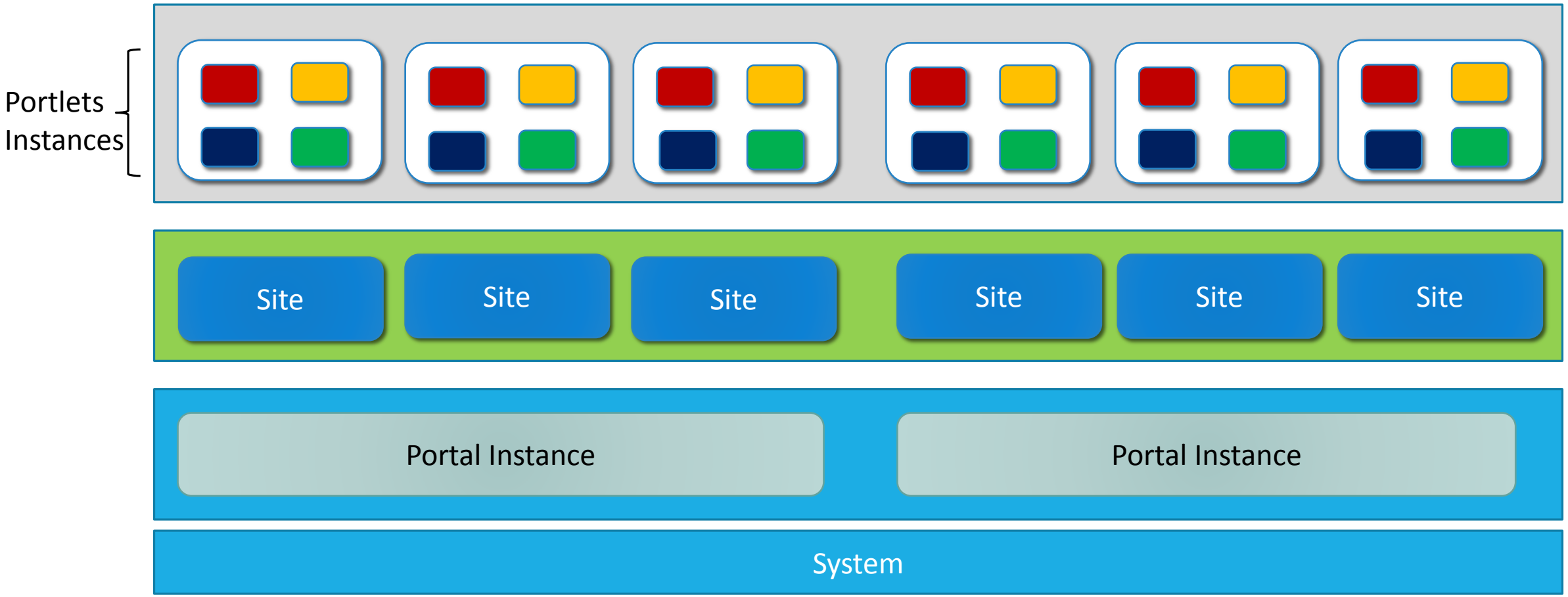
This talk ...

- Liferay monolithic architecture
- Modularity promises
- Liferay 7 modular architecture
- Building modules in Liferay 7: the OSGi way
- Customizing & extending modules
- Lessons learned & takeaways

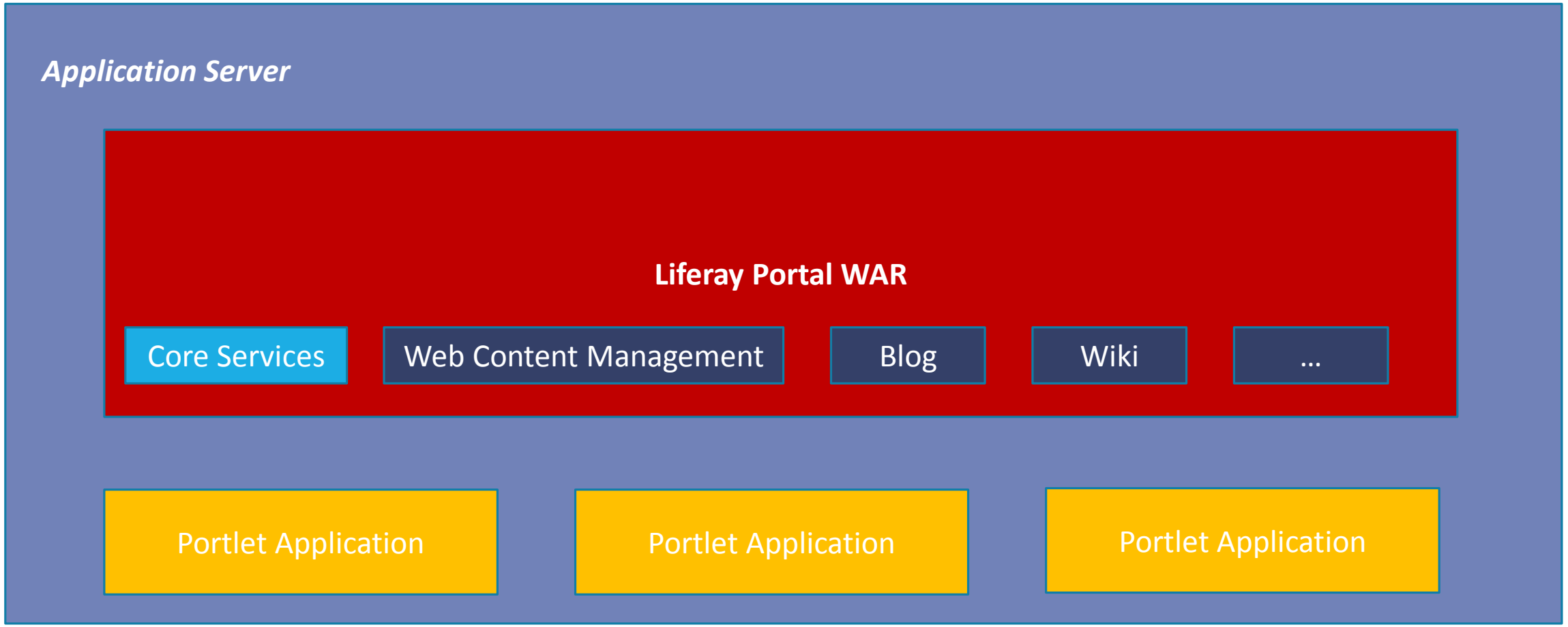
A few words about Liferay

- Open source leader Portal that implement Portlet API 1.0 (JSR 168) and Portlet 2.0 (JSR 286)
- Lines of Code : 5. 1 Millions
- About 70 Out of The Box Portlets
- Features: Web Content Management, Document Management, Workflow, Search, Enterprise Collaboration & Social Networking, …
- A marketplace: 490 apps http://liferay.com/marketplace

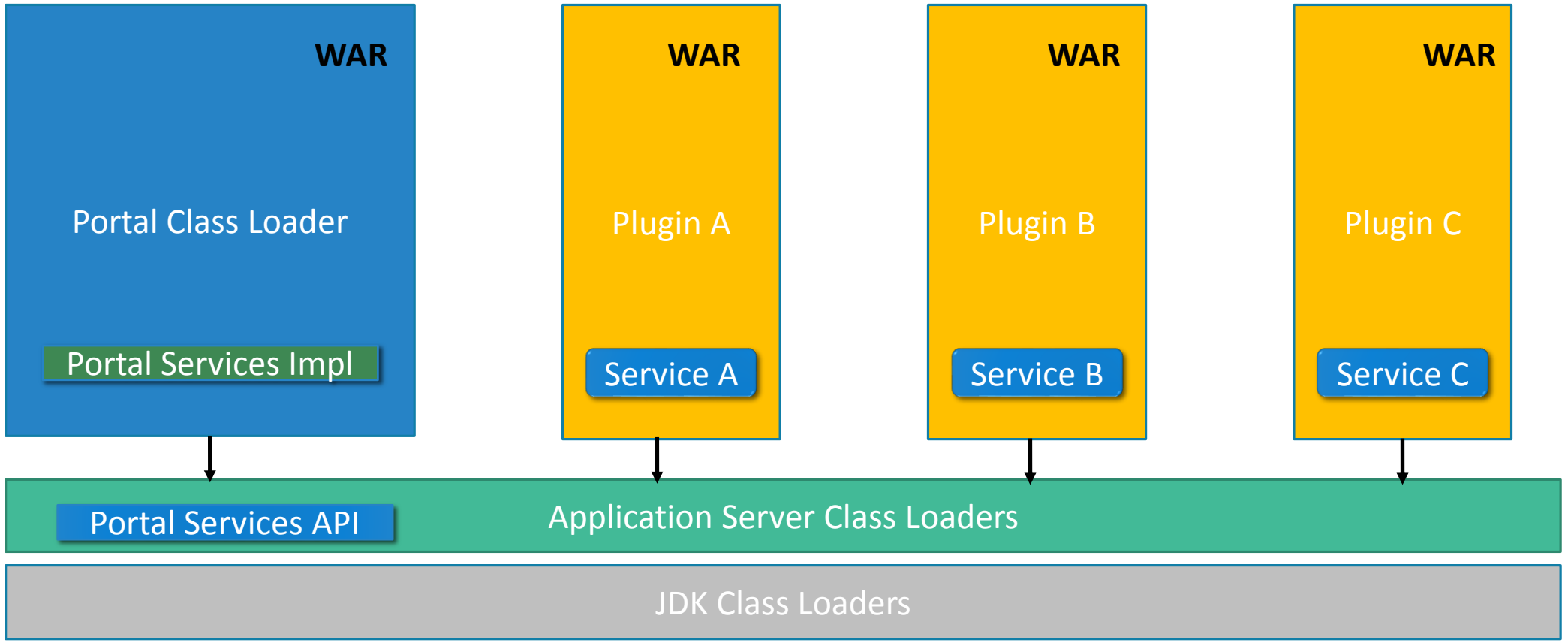
A few words about Liferay



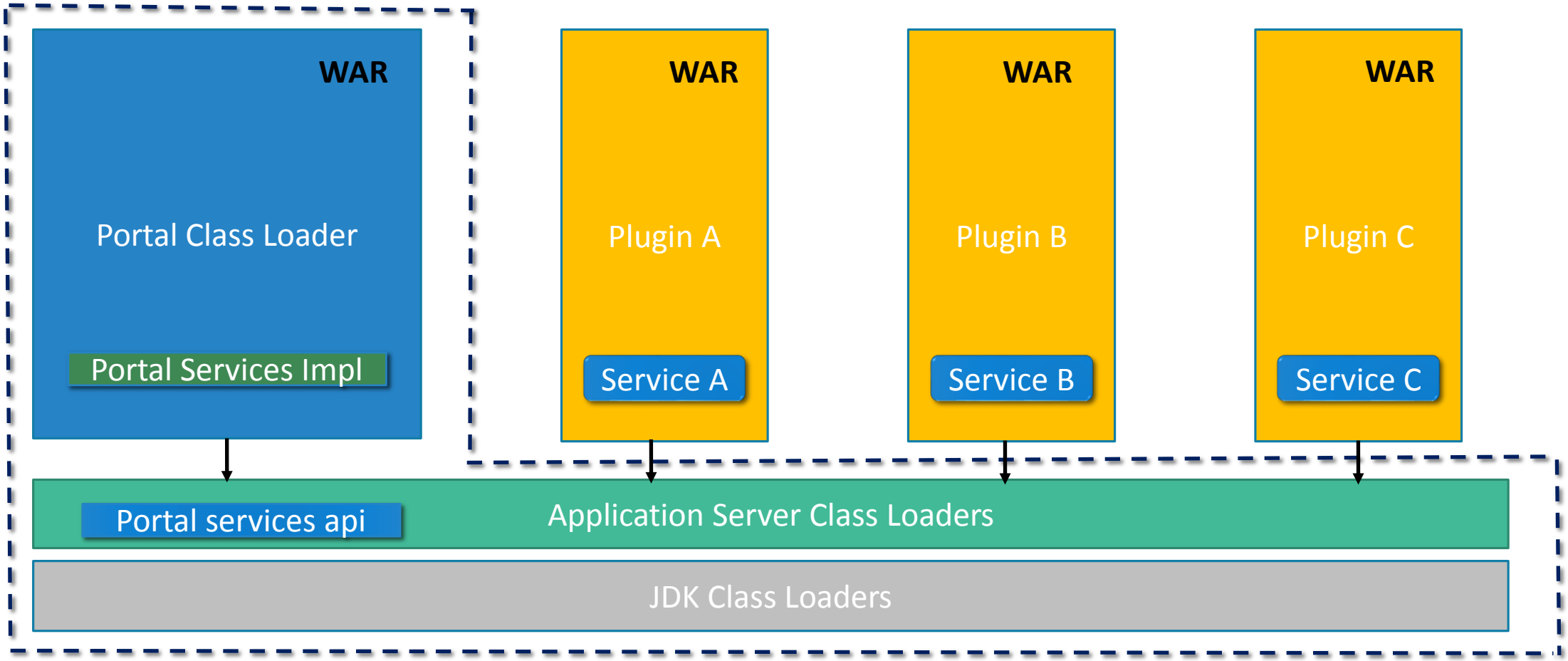
Liferay monolithic architecture



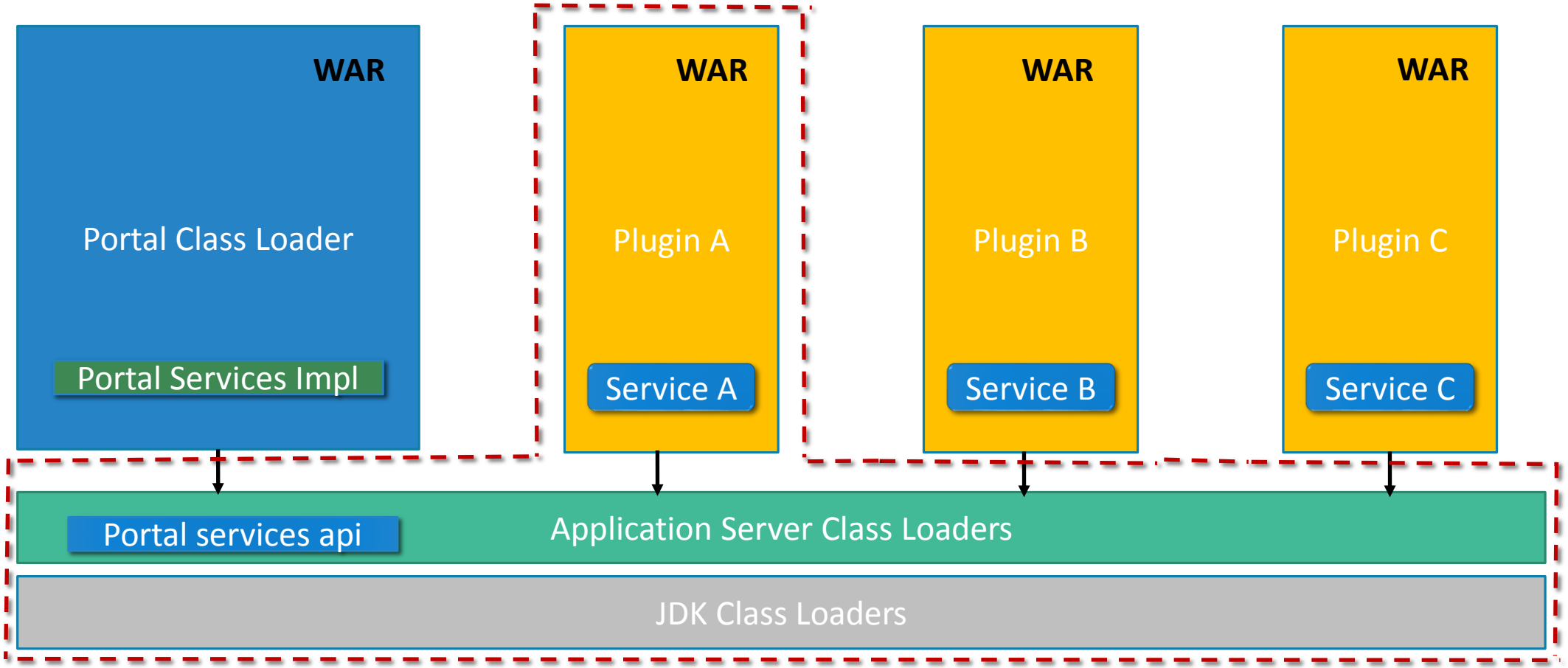
Liferay class loading hierarchy



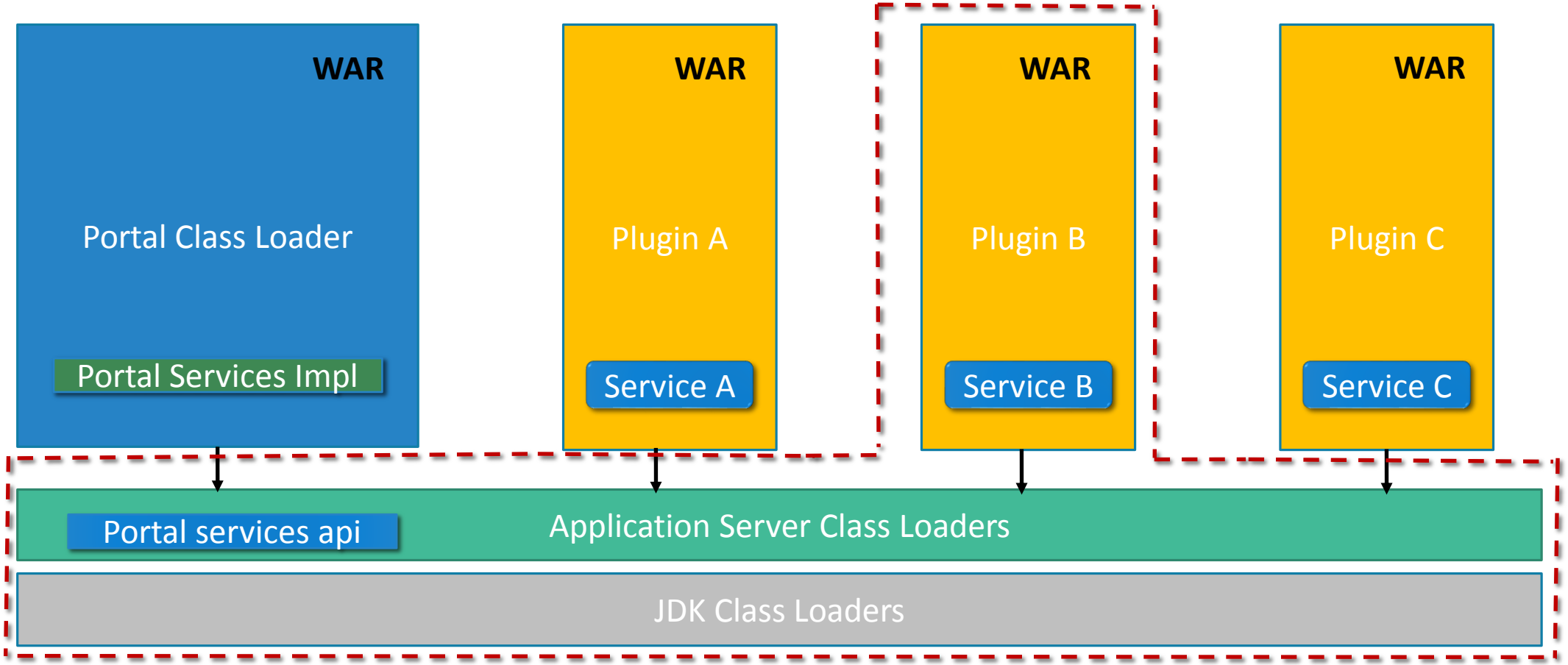
Liferay class loading hierarchy



Liferay class loading hierarchy



Liferay class loading hierarchy

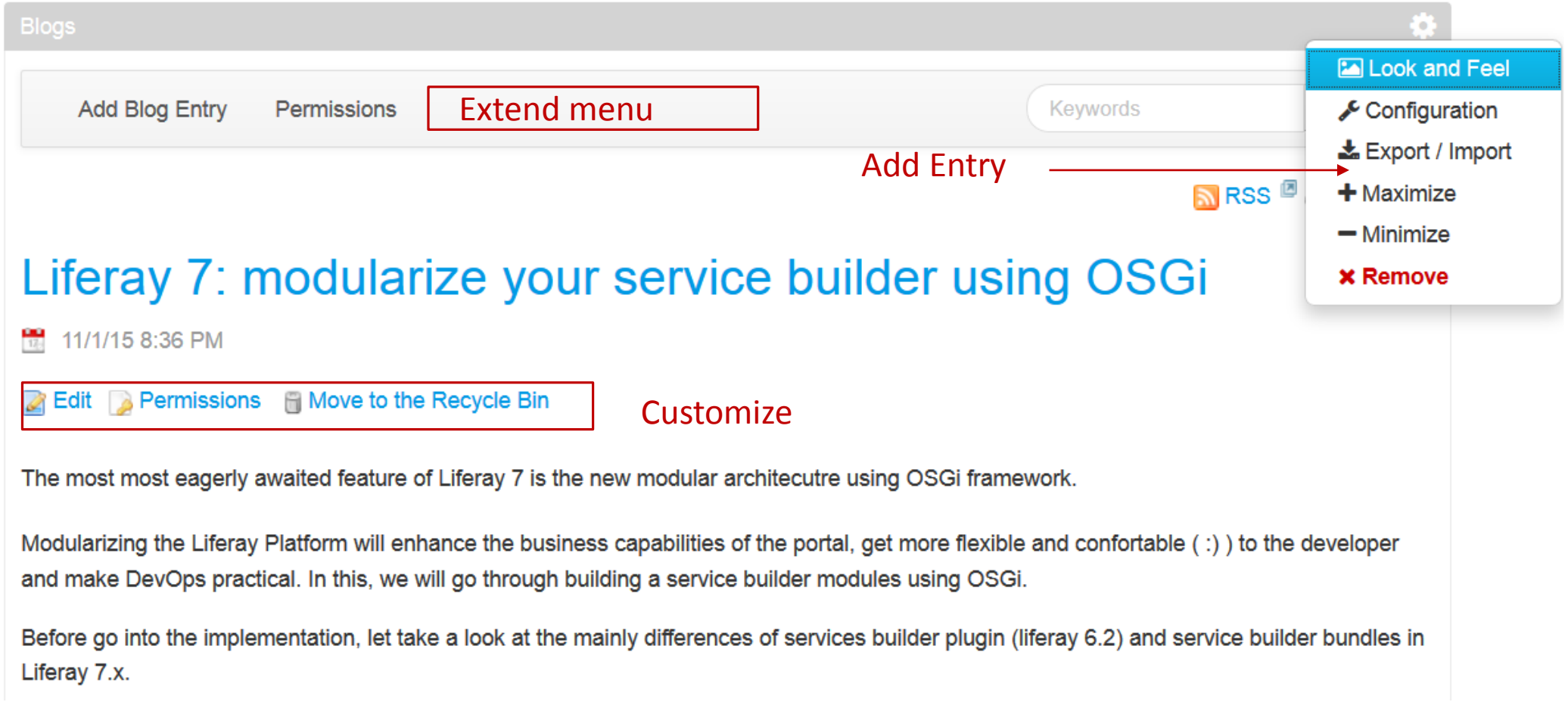


Portal customization capabilities

Collaboration Services: (Blogs/Forum/Wiki/Calendar/Polls/Messaging/Chat)	1	2	3	4	5	Totals
Collaborative Services: System provides several applications that can be configured for use by users depending on the site design. These services include the ability for end users to define and share content, messages, polls, and events.						
Collaboration Admin: System allows administrators the ability to configure and control which social media features and functionalities are accessible to individuals users.						
Blog: System provides blog post capabilities and features for end users. Users are be able to draft, publish, and edit blog postings for their account.						
Blog WYSIWYG: Users are able to create/edit blog posts using a rich text editor.						

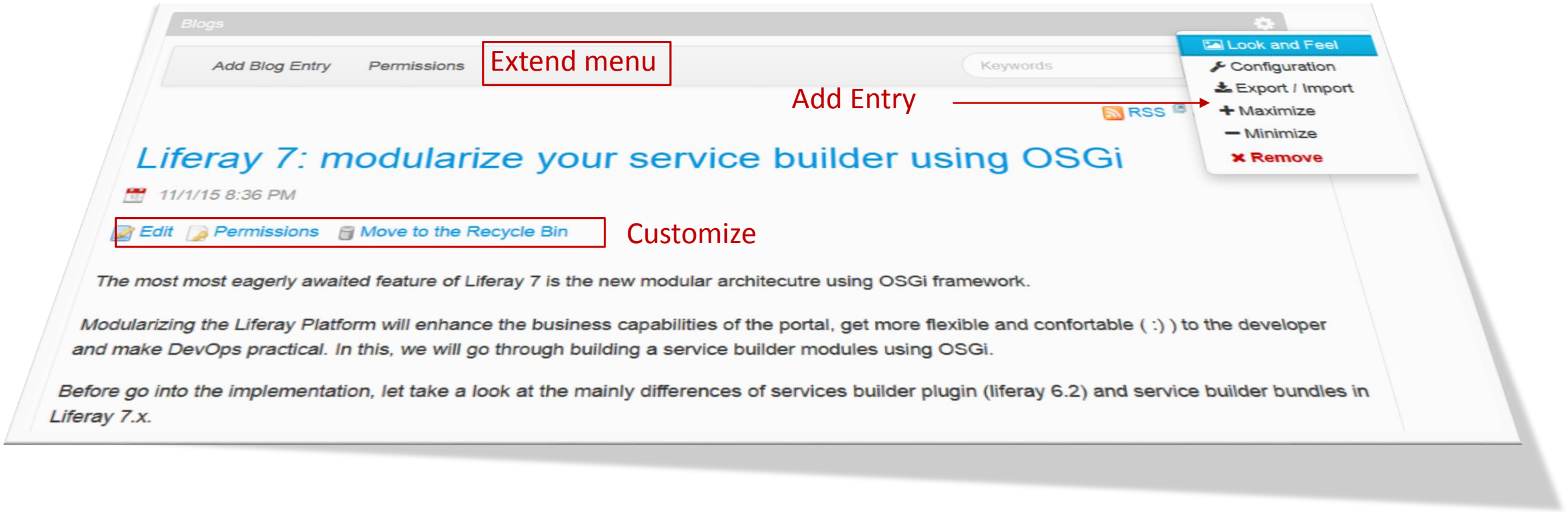
* Liferay Buyer's Checklist

Portal customization capabilities

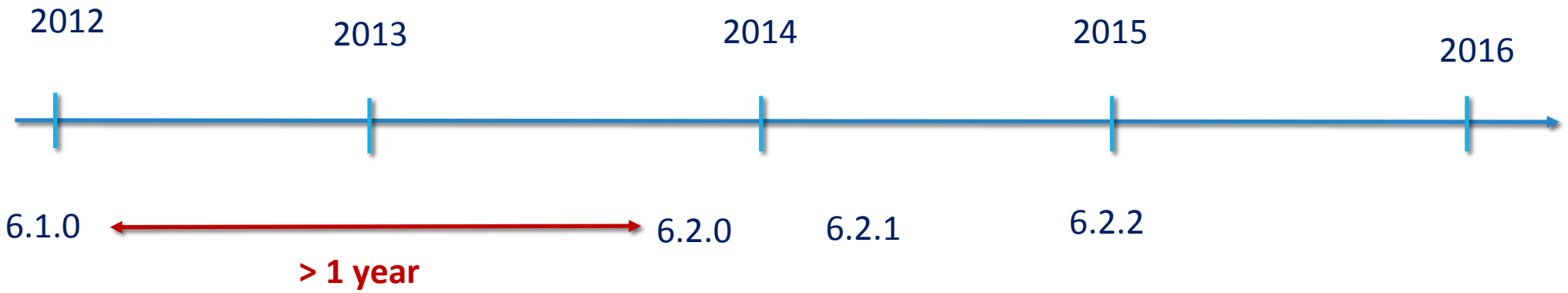


Portal customization capabilities

Customization should be a first class citizen



Liferay releases vs Business agility



- Business agility
- Recurrent incremental change

services building and delivery are going from few months to few weeks to few days

Limits of the Liferay monolithic architecture

- Invoking service between plugins /portlets:
  - No standard solution
  - Technical debt: hard to maintain
- An All-in-One Package
  - One big war of **230 MB**
  - Cannot deploy only what is needed: minimal version
  - Cannot manage portal features separately
- Deployment depends on app server
- Scalability: only one dimension scaling
- Marketplace: overriding JSP creates conflicts

Modularity promises

- Portlet independent versioning from Liferay Portal
  - ✓OSGi semantic versioning
- Business Agility:
  - ✓More frequent delivery of new features or improvements
  - ✓Easy and decoupled development process.
- Contract first approach / Loose coupling
- Dynamic extensions

Modularity promises

- Resiliency /design for failure
- Enhance Security : bundle isolation/seal
- Patching : just replace the bundle
- Microservices: small and independent (both for development and deployment)

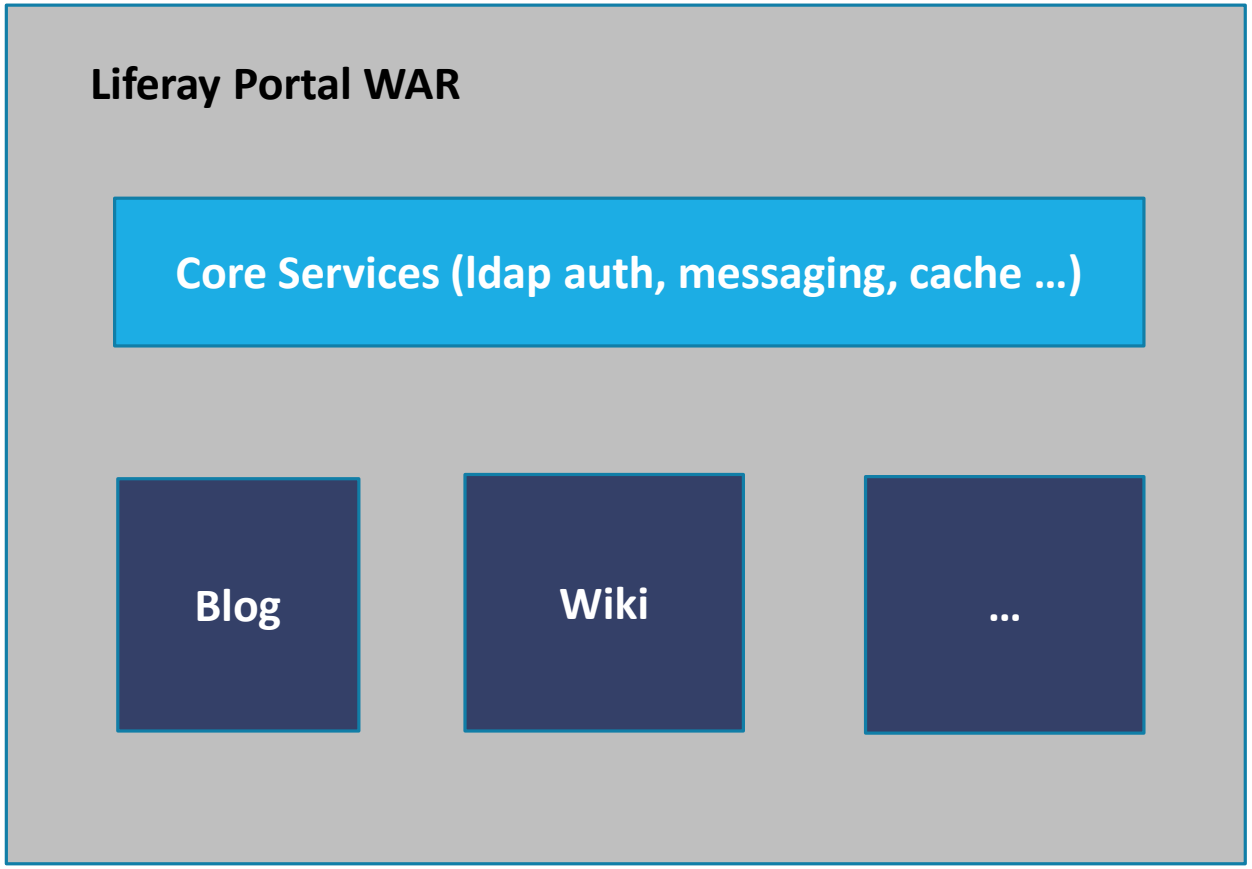
Make your product Powerfully customizable

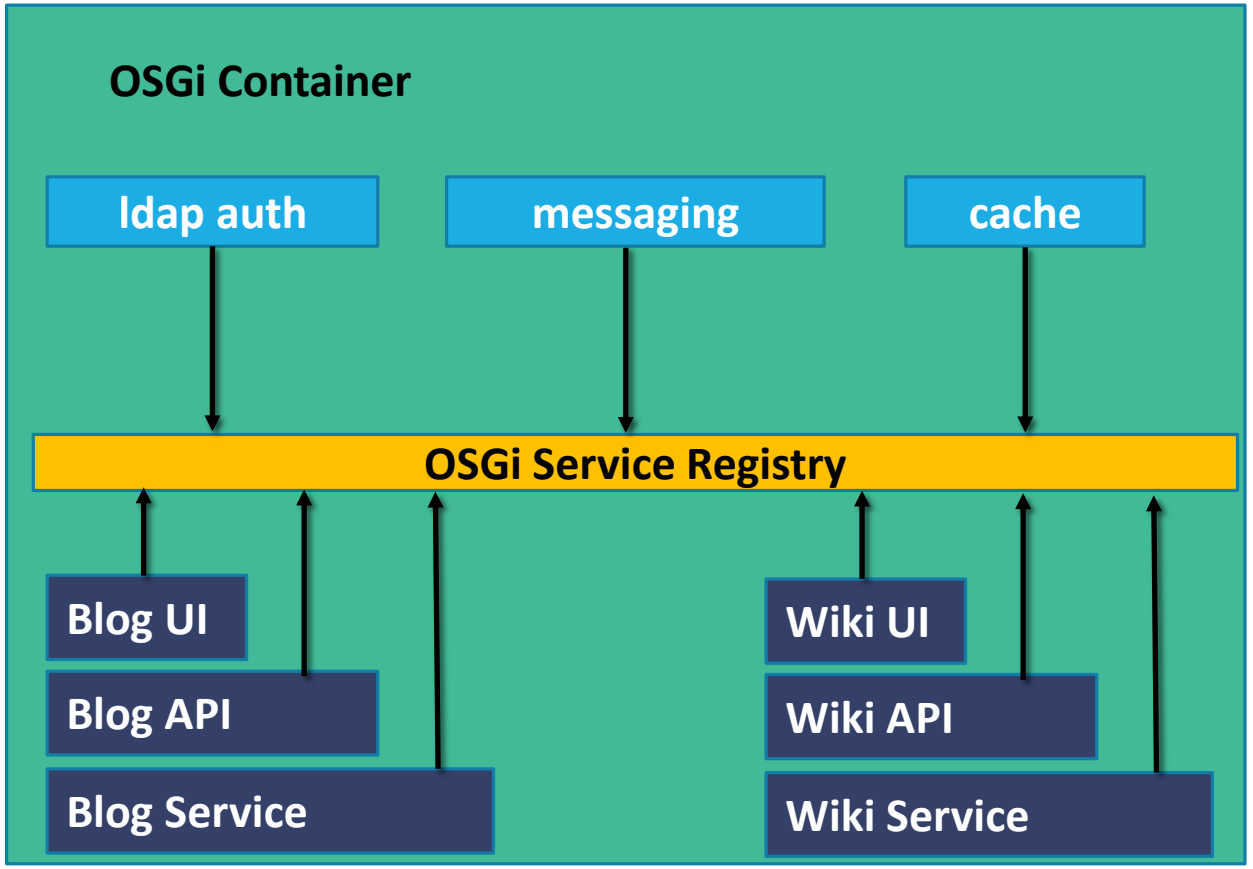
Modularity challenges

- Communications challenges
  => OSGi provides in VM-microservices.
  Zero configuration.

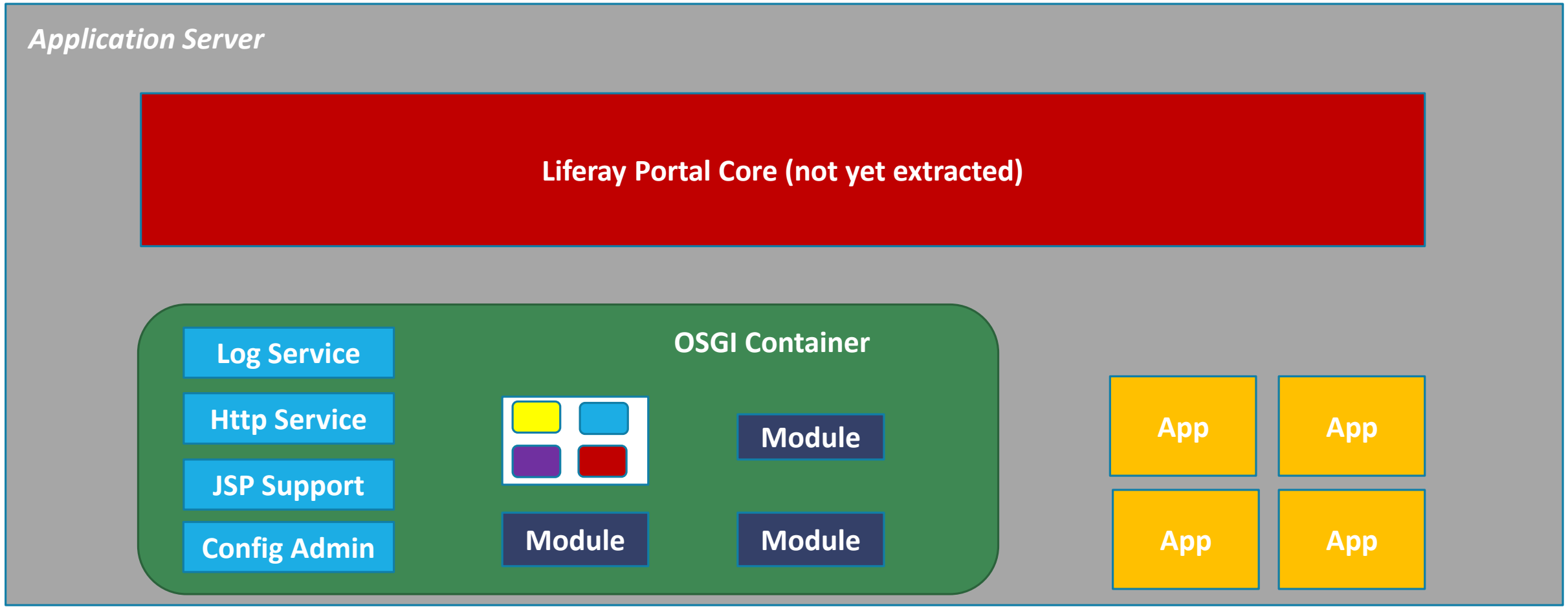
- How do I manage the configuration ?
  OSGi Framework provides Configuration Admin service.

From monolithic to microservices





Liferay 7 modular architecture

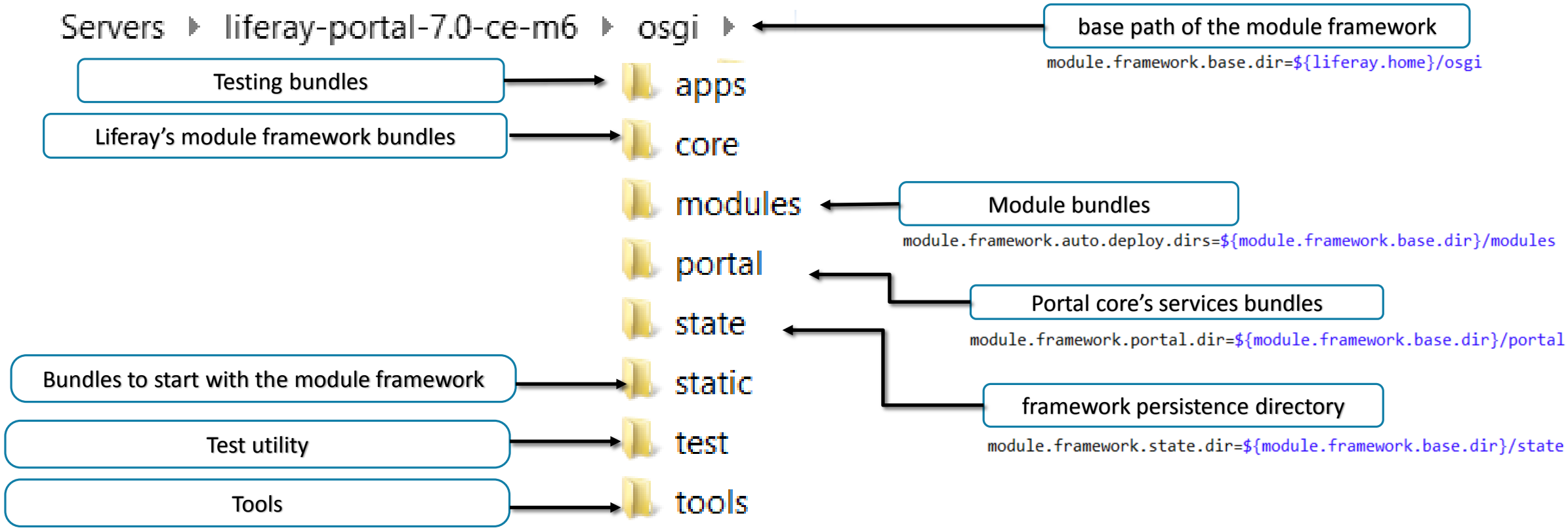


Liferay 7 modular architecture

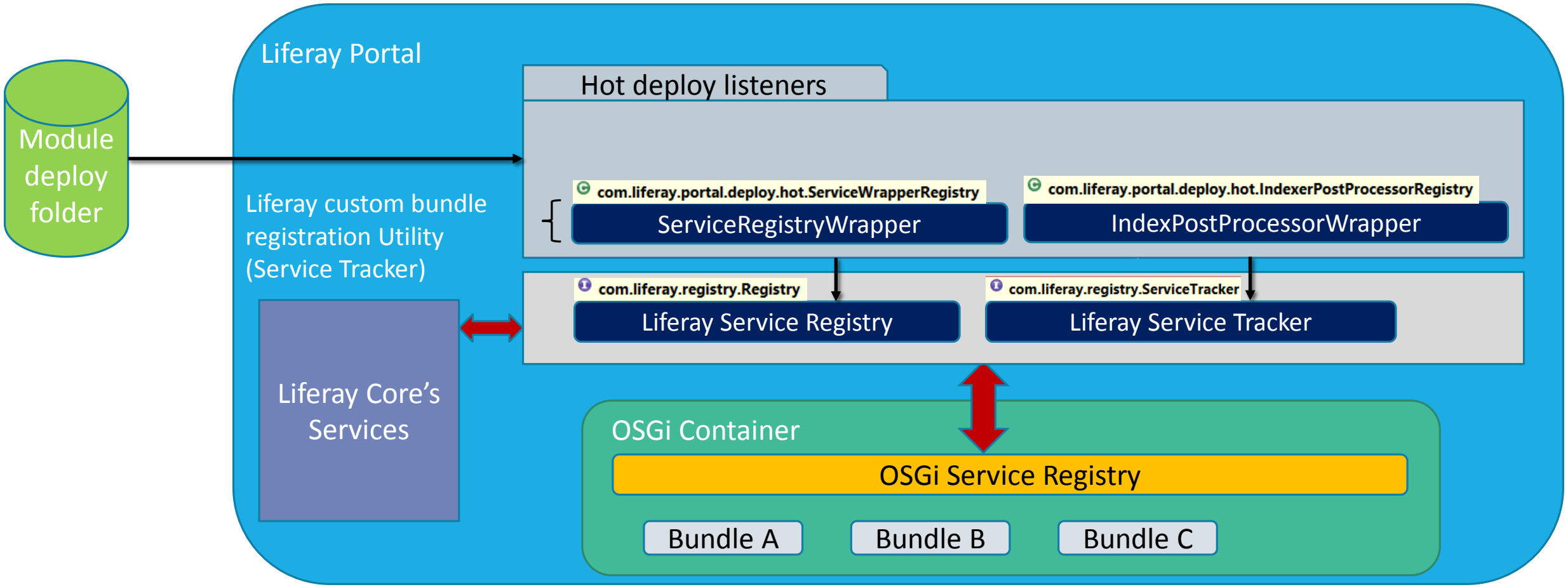
Statistics (based on Liferay 7 alpha1):

- Number of extracted bundles: 326
- Number of integration points > 200

Liferay module framework



Liferay module framework



Liferay module framework

- Liferay 7 owns the deployment Lifecycle: no longer relaying on application server for deployment.
- Dynamically manage module lifecycles.
- Liferay modules are versioned and explicitly declare dependencies.

Building modules with OSGi

- Liferay support various OSGi framework:
  - ✓ OSGi API
  - ✓ Blueprint
  - ✓ iPOJO
  - ✓ OSGi Declarative Services

Building modules with OSGi

Which technology is recommended by Liferay ?

✓ Liferay recommendation is to use declarative services.

Portlets using Declaratives Services

XML Configuration

6.2 and earlier

portlet.xml

liferay-portlet.xml

liferay-display.xml

Annotation (DS)

7.0

```
@Component(
        immediate = true, // start immediately when import packages are resolved
                        property = {
                                "com.liferay.portlet.display-category=category.sample",
                                "com.liferay.portlet.instanceable=true",
                                "javax.portlet.display-name=Sample OSGi DS Portlet",
                                "javax.portlet.security-role-ref=power-user,user"
                        },
                        service = Portlet.class // Expose the API, register as Portlet
)
public class SamplePortlet extends GenericPortlet {
```

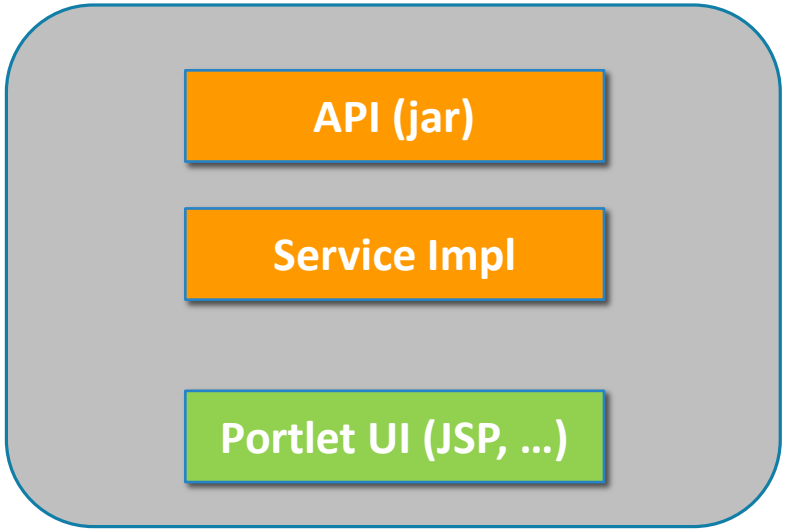
Portlet as a service

Modularize the service builder

- Service builder : Liferay service layer code scaffolding

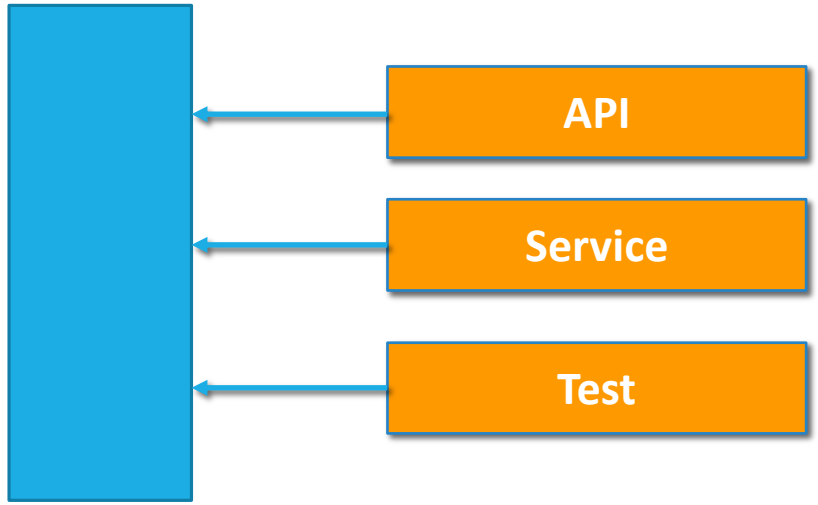
6.2 and earlier

Portlet Application (WAR)



7.0

Bundles (Jars)

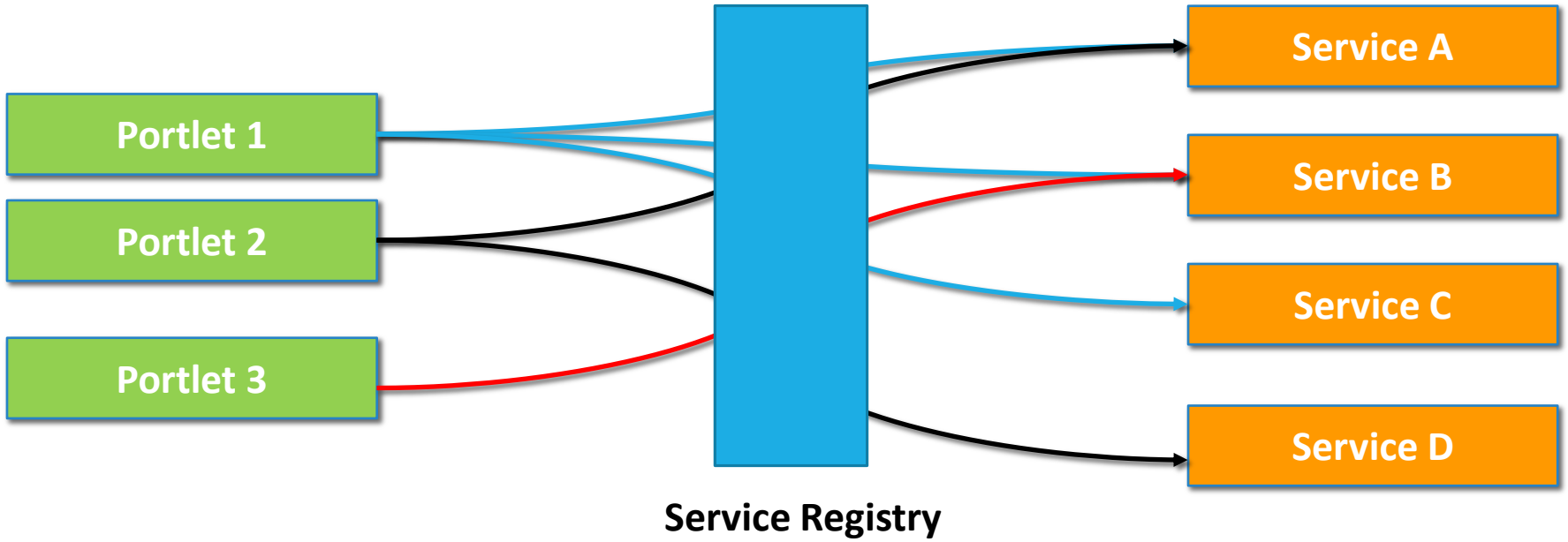


Service Registry

Sharing services between applications

- Zero Effort, Zero Configuration !

All what you need is to publish your services in the OSGi service registry.



Overriding Liferay's services

```
@Component(
        immediate = true,
        property = {},
        service = ServiceWrapper.class // Expose the API, register the hook as ServiceWrapper
)
public class UserLoginTrackerServiceHook extends UserLocalServiceWrapper {
```

```
@Override
public User updateLastLogin(long userId, String loginIP) throws PortalException {

    log.info("User '" + userId + "' has connected on " + new Date() + " from the IP address " + loginIP);

    return super.updateLastLogin(userId, loginIP);
}
```

Overriding core services

- deploy a service with a higher service ranking than the original

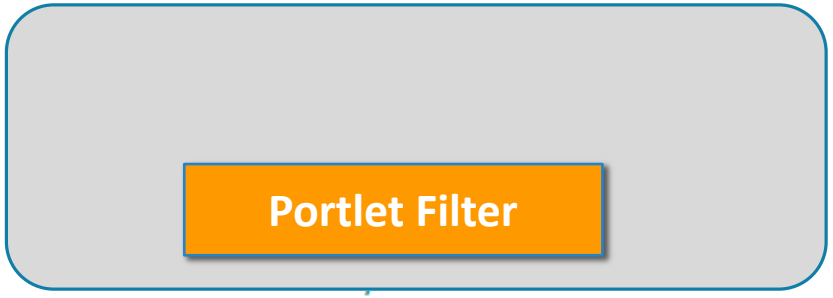
Using OSGi service ranking :

```
property= {"service.ranking:Integer=100"}
```

Portlet Filter

6.2 and earlier

Portlet Application (WAR)




```
<filter>
    <filter-name>LoginPortletFilter</filter-name>
    <filter-class>com.innovsquare.showcase.portlet.filter.LoginPortletFilter</filter-class>
    <lifecycle>RENDER_PHASE</lifecycle>
</filter>
```

- **Defined inside the portlet app !**
- **Hard to implement filter for OTB portlets**

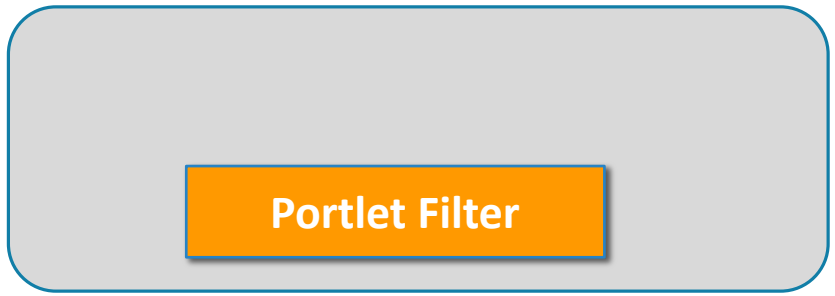
7.0

Bundles (Jars)

Portlet Filter

6.2 and earlier

Portlet Application (WAR)




```
<filter>
    <filter-name>LoginPortletFilter</filter-name>
    <filter-class>com.innovsquare.showcase.portlet.filter.LoginPortletFilter</filter-class>
    <lifecycle>RENDER_PHASE</lifecycle>
</filter>
```

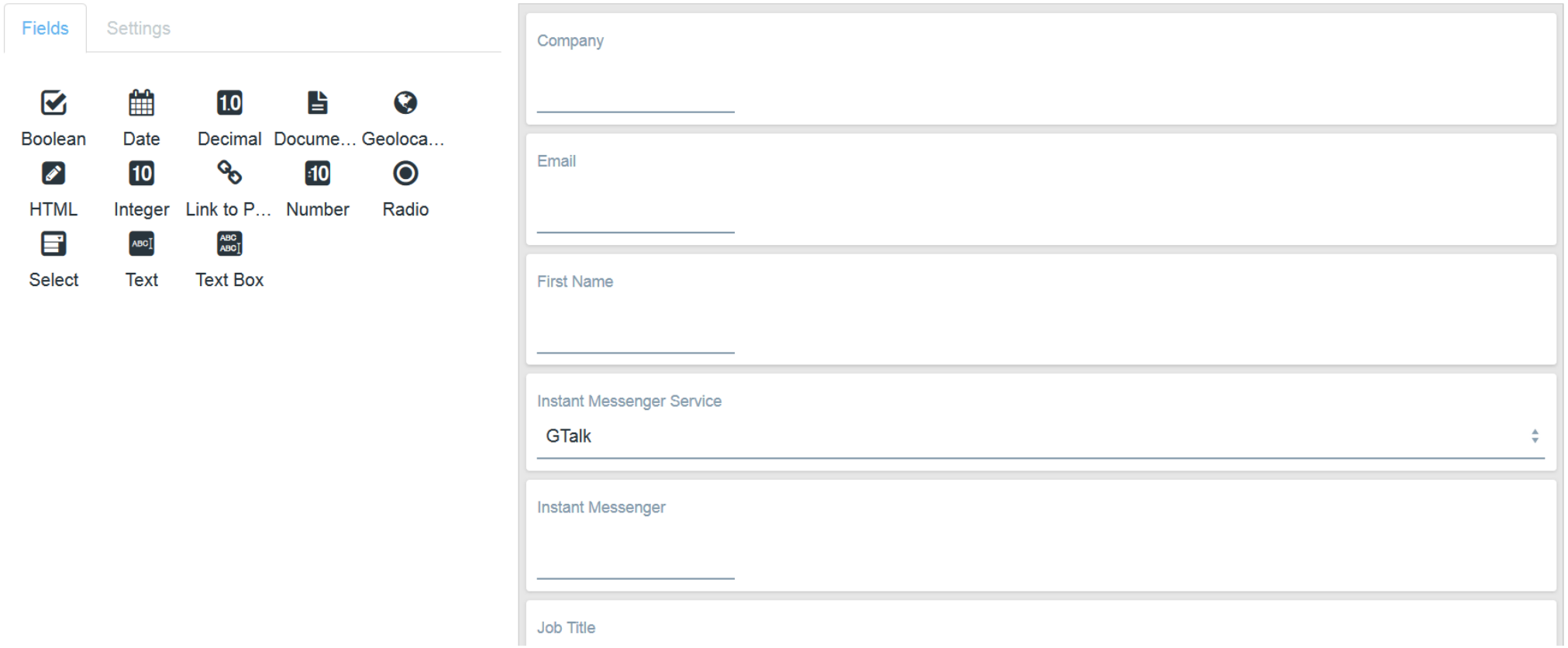
- **Defined inside the portlet app !**
- **Hard to implement filter for OTB portlets**

7.0

Bundles (Jars)

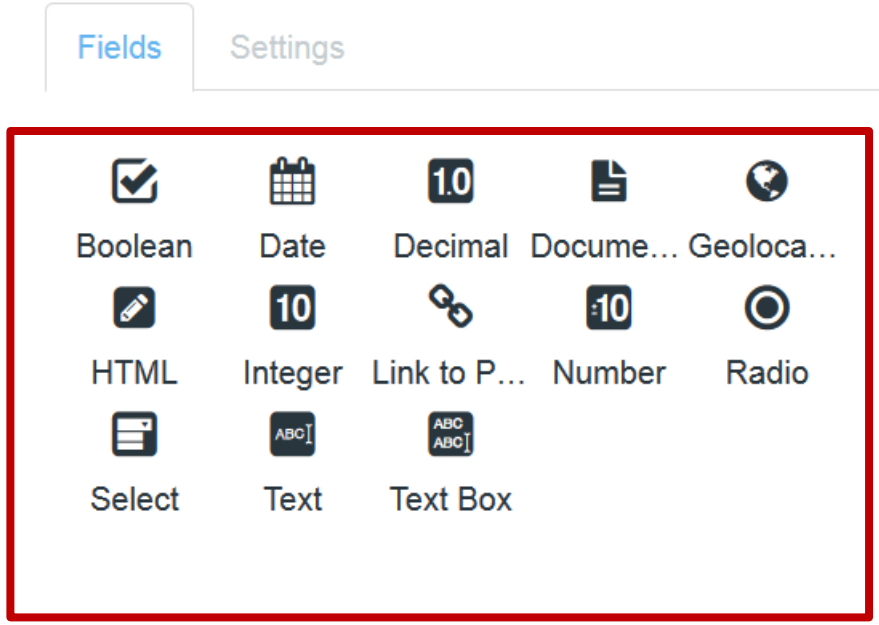

```
@Component(
        immediate = true,
        property = {
            "javax.portlet.name=com_innovsquare_signin_web_portlet_LoginPortlet"
        },
        service = PortletFilter.class
    )

public class LoginPortletFilter  implements RenderFilter  {

        @Override
        public void doFilter(RenderRequest request, RenderResponse response,
                FilterChain chain) throws IOException, PortletException {
            // stuff
        }
```

Split into modules : Form Builder as example

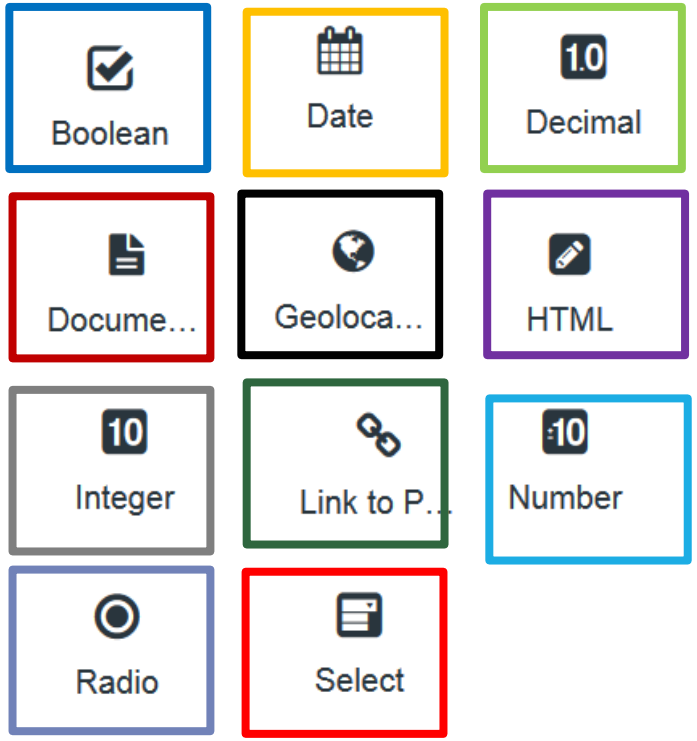


Split into microservices: Form Builder as example

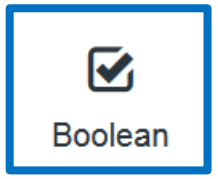
6.2 (Monolithic)



7 (OSGI)

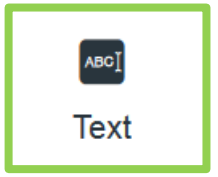


Split into microservices: Form Builder as example

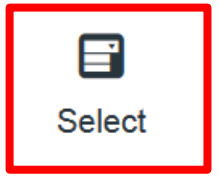


com.liferay.dynamic.data.mapping.type.checkbox.jar

com.liferay.dynamic.data.mapping.type.radio.jar

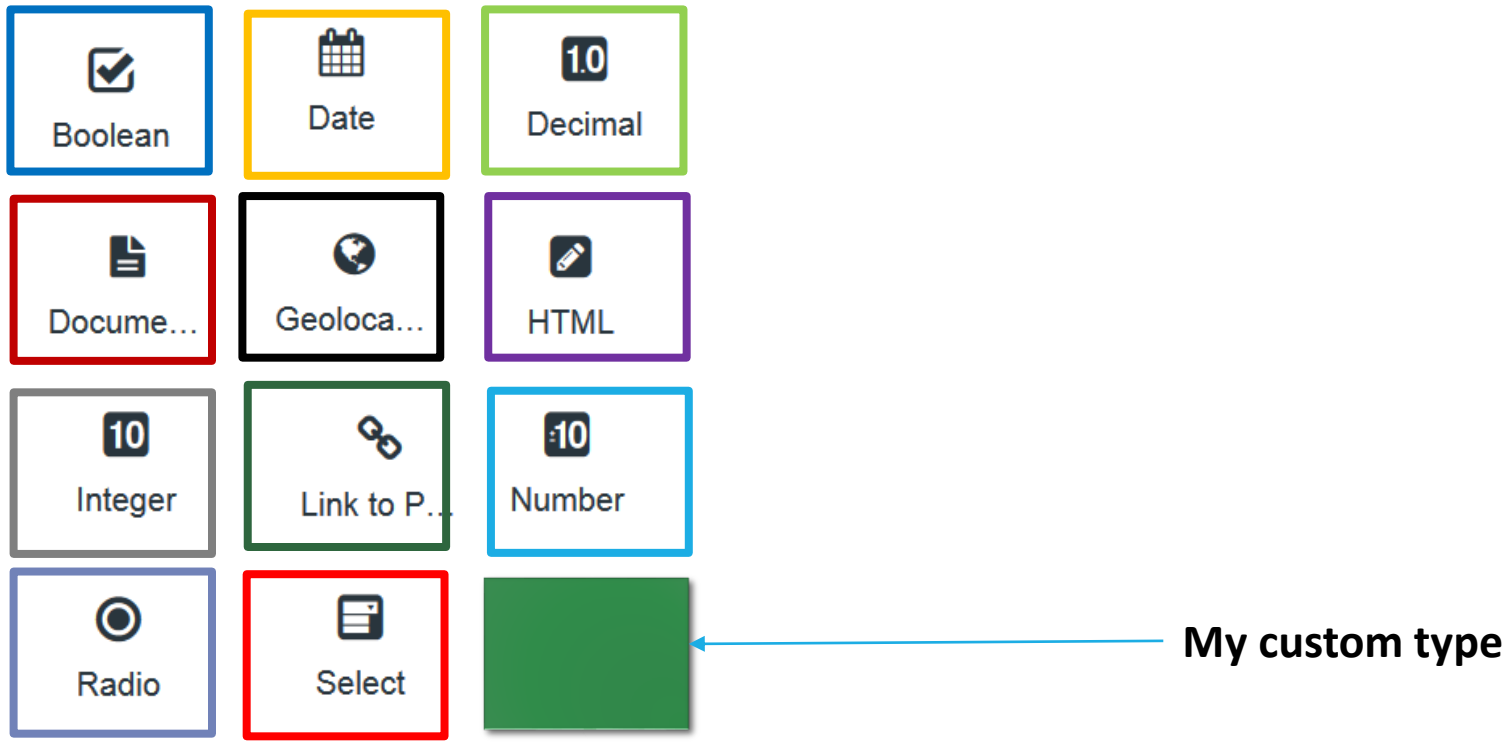


com.liferay.dynamic.data.mapping.type.text.jar



com.liferay.dynamic.data.mapping.type.select.jar

Split into microservices: build for extension



Portal Configuration API

Available Configuration for Liferay 6:

- Portal properties files:
  - don't have types
  - restart on every change
- Portal.properties file:
  - One big file with 10000 lines
- Portlets preferences:
  - XML based
  - don't support types

```
##
## Blogs Portlet
##

    #
    # Set the location of the XML file containing the configuration of the
    # default display templates for the Blogs portlet.
    #
    blogs.display.templates.config=com/liferay/portlet/blogs/dependencies/portlet-display-templates.xml

    #
    # Set the interval in minutes on how often CheckEntryMessageListener will
    # run to check for and display blog entries scheduled to display.
    #
    blogs.entry.check.interval=1

    #
    # Set the interval on which the LinkbackMessageListener will run. The value
    # is set in one minute increments.
    #
    blogs.linkback.job.interval=5
```

Portal Configuration API

Configuration management based on :

- OSGi Configuration Admin
- OSGi MetaType

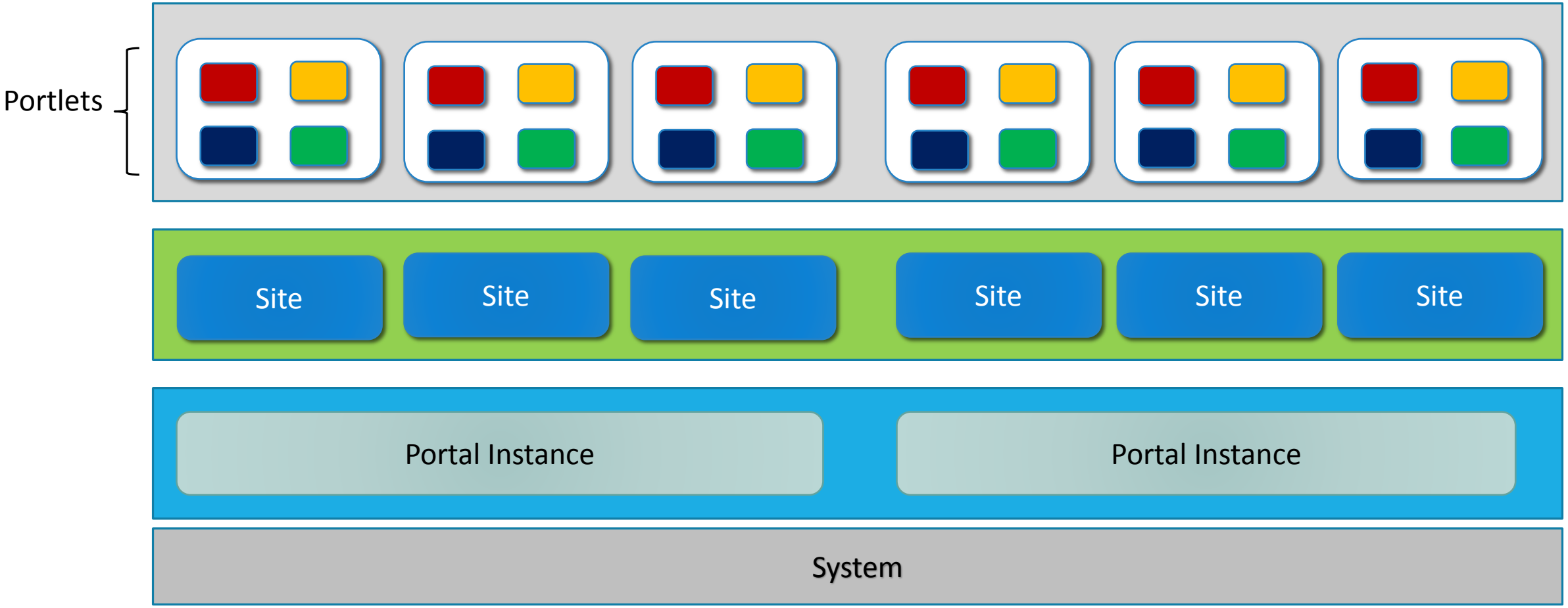
- ✓ Properties are typed
- ✓ Properties are well separated by modules
- ✓ Dynamically load properties on runtime

Portal Configuration API

```
##
## Blogs Portlet
##


    #
    # Set the location of the XML file containing the configuration of the
    # default display templates for the Blogs portlet.
    #
    blogs.display.templates.config=com/liferay/portlet/blogs/dependencies/portlet-display-templates.xml

    #
    # Set the interval in minutes on how often CheckEntryMessageListener will
    # run to check for and display blog entries scheduled to display.
    #
    blogs.entry.check.interval=1

    #
    # Set the interval on which the LinkbackMessageListener will run. The value
    # is set in one minute increments.
    #
    blogs.linkback.job.interval=5
```

```
@Meta.OCD(id = "com.liferay.blogs.configuration.BlogsConfiguration")
public interface BlogsConfiguration {

    /**
     * Set the location of the XML file containing the configuration of the
     * default display templates for the Blogs portlet.
     */
    @Meta.AD(
        deflt = "com/liferay/blogs/web/template/dependencies/portlet-display-templates.xml",
        required = false
    )
    public String displayTemplatesConfig();

    /**
     * Set the interval in minutes on how often CheckEntryMessageListener will
     * run to check for and display blog entries scheduled to display.
     */
    @Meta.AD(deflt = "1", required = false)
    public int entryCheckInterval();

    /**
     * Set the interval on which the LinkbackMessageListener will run. The value
     * is set in one minute increments.
     */
    @Meta.AD(deflt = "5", required = false)
    public int linkbackJobInterval();
```

Portal Configuration API - Scopes



Portal Configuration API –Customize properties

- Customize properties using Configuration Admin portlet:

Portal Configuration API –Customize properties

- Locate the Configuration class : annotated with `@Meta.OCD`

```
@Meta.OCD(
    id = "com.liferay.journal.configuration.JournalGroupServiceConfiguration"
)
public interface JournalGroupServiceConfiguration {
```

- Create a .cfg file with the id:

com.liferay.journal.configuration.JournalGroupServiceConfiguration.cfg

- Add properties with the new values:

*admin.email.from.address=*contentmanager@mycompany.com

*admin.email.from.name=*contentmanager@mycompany.com

- Drop it into the deploy folder of Liferay

UI customization & extension

- extensible user interfaces using the already built in mechanisms into the platform.
- Dynamic include using the Liferay extension points :

<liferay-util:dynamic-include key="com.liferay.frontend.editors.web"/>

Customizing UI Components

- Why: provide a better configuration for your needs.

```
@Component(
        property = {
            "editor.config.key=contentEditor",
            "editor.name=alloyeditor",
            "editor.name=ckeditor",
            "javax.portlet.name=33","javax.portlet.name=my-custom-portlet-id",
            "service.ranking:Integer=100"
        },

        service = EditorConfigContributor.class
    )
public class MyEditorAddon extends BaseEditorConfigContributor {
```

Customizing UI Components

```
@Override
public void populateConfigJSONObject(
        JSONObject jsonObject, Map<String, Object> inputEditorTaglibAttributes,
        ThemeDisplay themeDisplay,
        LiferayPortletResponse liferayPortletResponse) {

    JSONObject toolbars = jsonObject.getJSONObject("toolbars");
    if (toolbars != null) {
        JSONObject toolbarAdd = toolbars.getJSONObject("add");

        if (toolbarAdd != null) {
            JSONArray addButtons = toolbarAdd.getJSONArray("buttons");

            addButtons.put("camera");
        }
    }

}
```

Dev Tools

- BND Tools:

  Robust OSGi bundles build tools.

- Blade Tools: Liferay tools to build modules

  https://github.com/gamerson/blade.tools

- Liferay provide also a set of plugins to build custom modules:

  com.liferay.portal.tools.sample.sql.builder
  com.liferay.portal.tools.service.builder
  com.liferay.portal.tools.upgrade.table.builder
  com.liferay.portal.tools.wsdd.builder

Useful Resources

- Liferay developer network:

  https://dev.liferay.com/develop/

- Liferay DevCon 2015:

  https://www.liferay.com/fr/web/events2015/devcon/recap

- Liferay Blade project:

  https://github.com/rotty3000/blade

- Liferay Blade Tools:

  https://github.com/gamerson/blade.tools

Conclusion – lessons learned

- Be realistic & do it in steps: leaving Rome wasn't built in a day !
  - First step: In-VM microservices
- Choose robust tools and standards: OSGi is one of the best for java world.
- Provide dev and migration tools :very important for your customers.
- Focus on Single Responsibility Principle (SRP)

Conclusion – takeaways

- Each @Component can be replaced with your own.
- Reusable components: Taglibs, Item Selector, Portlet decorator …
- Customize and extend

Questions ?


eclipsecon Europe
Ludwigsburg, Germany, 3 - 5 November 2015
Evaluate the sessions at www.eclipsecon.org
+1
0
-1

You want to learn more ?

2 talks by Ray Augé :

Today :

16:45 - 17:20	**Better WebApp Development using OSGi** OSGi Raymond Auge [Liferay, Inc.]

Tomorrow:

14:30 - 15:05	**Massive Enterprise Product Migration to OSGi** OSGi Raymond Auge [Liferay, Inc.]