OSGi for mere mortals

Simple and robust software is good for your work-life balance.

Bertrand Delacrétaz
Senior Developer, Adobe CQ5 team, Basel

Apache Software Foundation Member and (current) Director

http://grep.codeconsult.ch - twitter: @bdelacretaz - bdelacretaz@apache.org

ApacheCon EU 2012, Sinsheim, November 2012

slides revision: 2012-11-06

OSGI FOR MERE MORTALS

ApacheCon Europe 2012
Rhein-Neckar-Arena, Sinsheim, Germany
5–8 November 2012

Adobe

BTW...we are hiring!
jobs-basel@adobe.com

slides at
slideshare.net/bdelacretaz/osgi-for-mere-mortals

code at
github.com/bdelacretaz/OSGi-for-mere-mortals

what?

a simple RESTful server
built from scratch on OSGi

to show that it's not rocket science,
even starting from scratch...

OSGI FOR
MERE MORTALS

RESTful server?

POST to store content:

```
$ date | curl -T - -X POST -D - http://localhost:8080/store/example

HTTP/1.1 201 Created
Location: /store/example
Content-Type: text/plain; charset=utf-8
Content-Length: 178
Server: Jetty(6.1.x)

Stored at /store/example
StoredBy:ch.x42.osgi.samples.osgi101.app.servlets.StorageServlet
StoredAt:Fri Nov 04 10:41:30 CET 2011
Path:/store/example

Fri Nov  4 10:41:30 CET 2011
```

OSGI FOR MERE MORTALS

RESTful server!

GET to retrieve content:

```
$ curl http://localhost:8080/store/example

StoredBy:ch.x42.osgi.samples.osgi101.app.servlets.StorageServlet
StoredAt:Fri Nov 04 10:41:30 CET 2011
Path:/store/example

Fri Nov  4 10:41:30 CET 2011
```

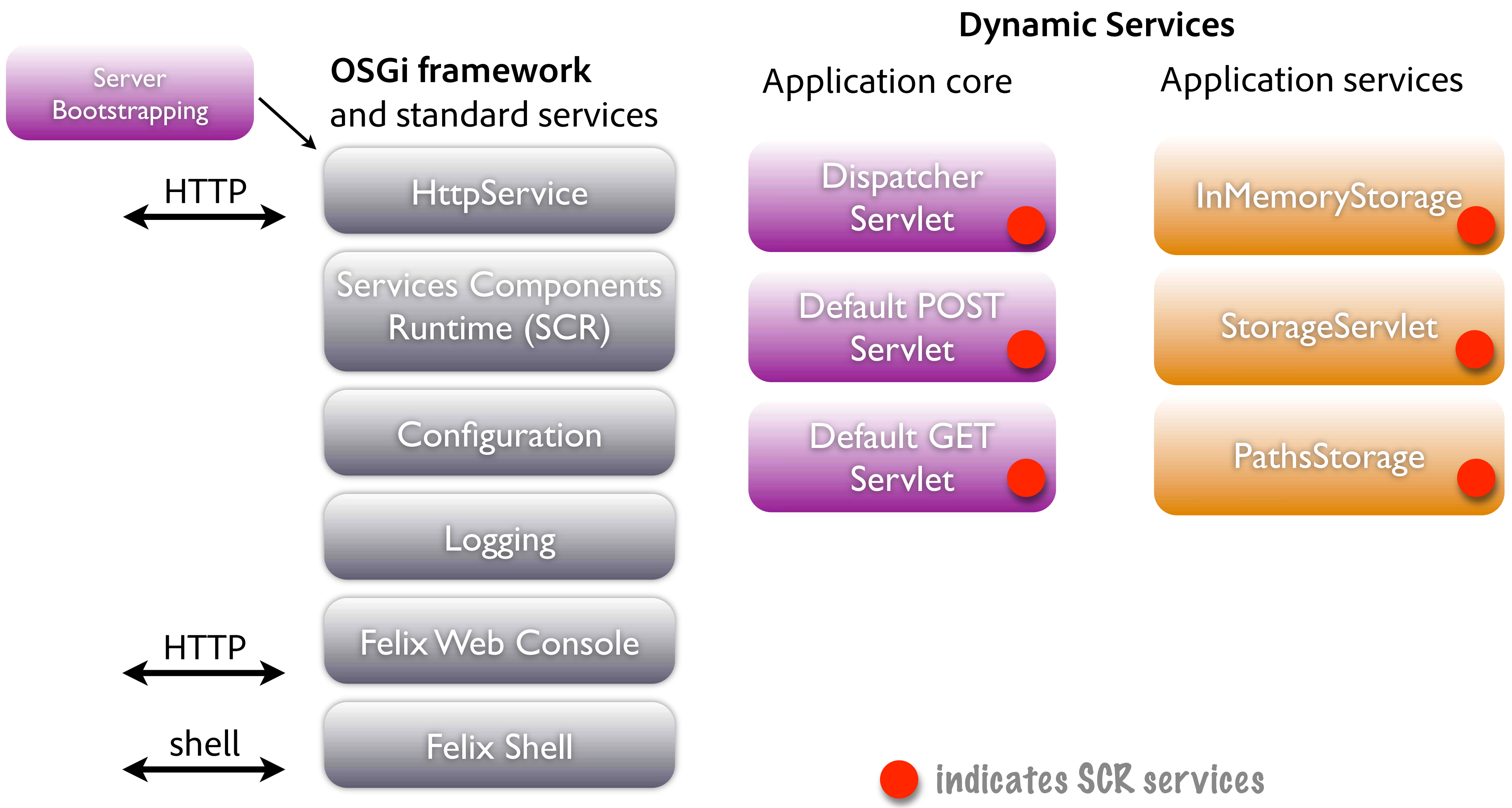
In terms of functionality, that's it!

It's not big...

```
107 ./app/pom.xml
 35 ./app/.../InMemoryStorage.java
 34 ./app/.../PathsStorage.java
147 ./app/.../StorageServlet.java
101 ./core/pom.xml
  6 ./core/...CoreConstants.java
 32 ./core/...DefaultGetServlet.java
 32 ./core/...DefaultPostServlet.java
116 ./core/...DispatcherServlet.java
  8 ./core/...Storage.java
208 ./launcher/pom.xml
 84 ./launcher/...OsgiBootstrap.java
 52 ./pom.xml
```

962 lines in total, including 468 lines in pom.xml files...

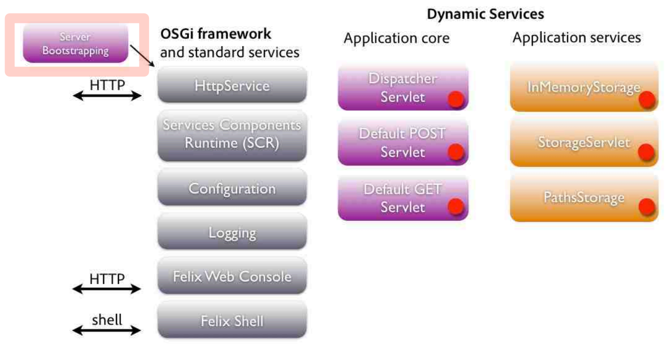
Components



● indicates SCR services

also:
maven-bundle plugin
maven-scr-plugin
maven-dependency-plugin

OSGi FOR MERE MORTALS

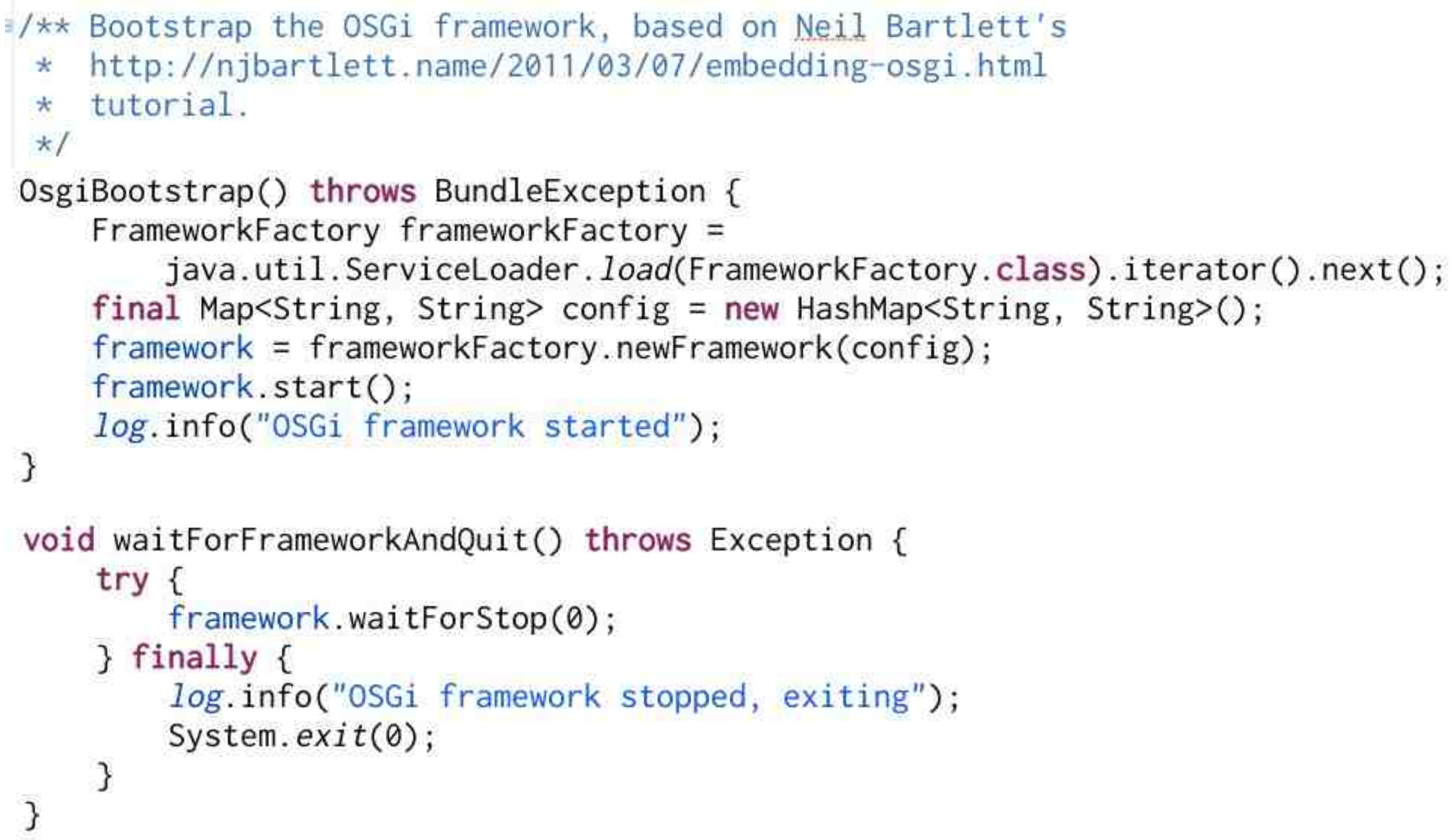


Bootstrapping

and loading bundles

Framework start and stop

Called from a plain main() method

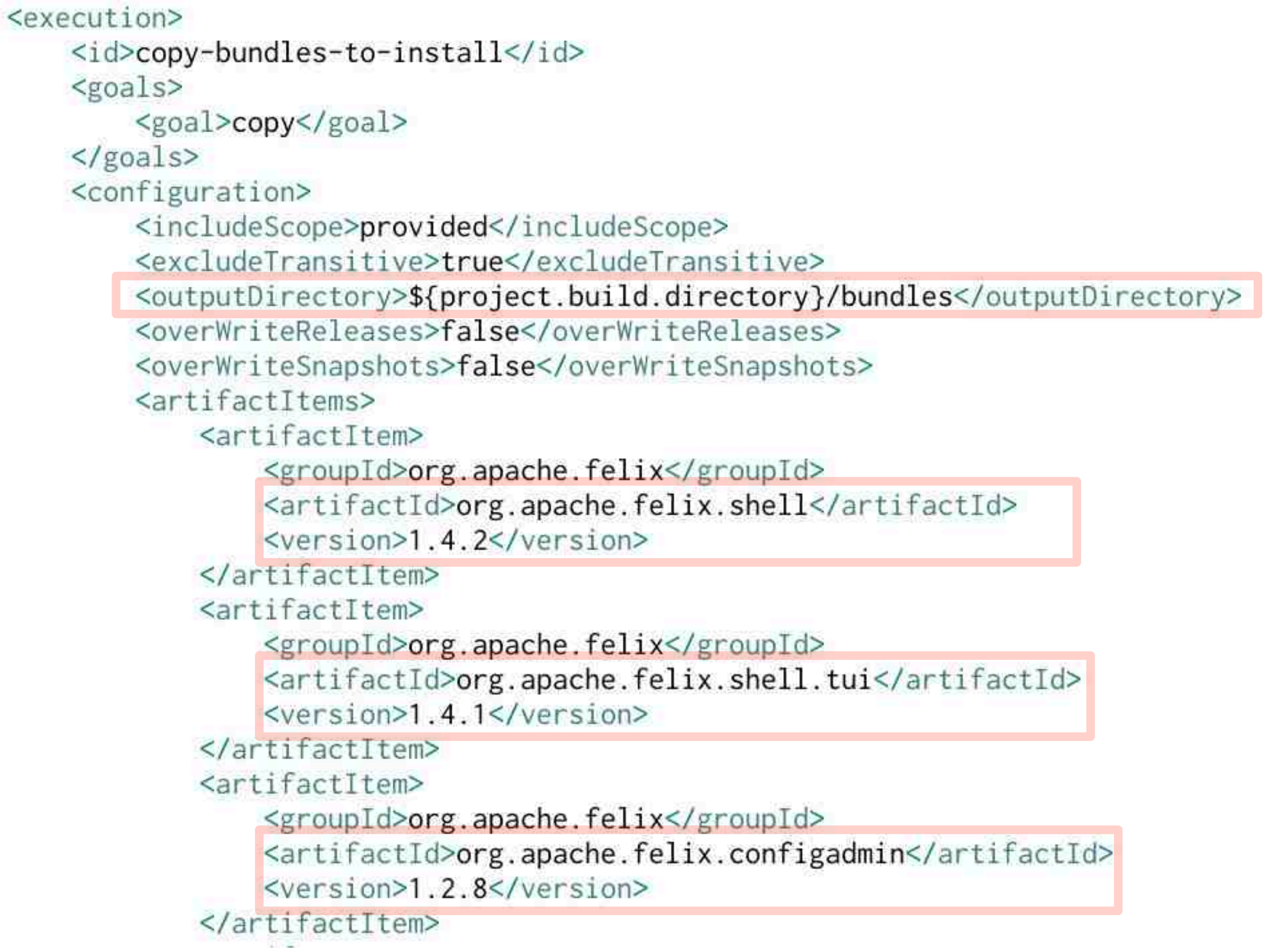


```java
/** Bootstrap the OSGi framework, based on Neil Bartlett's
 *  http://njbartlett.name/2011/03/07/embedding-osgi.html
 *  tutorial.
 */
OsgiBootstrap() throws BundleException {
    FrameworkFactory frameworkFactory =
        java.util.ServiceLoader.load(FrameworkFactory.class).iterator().next();
    final Map<String, String> config = new HashMap<String, String>();
    framework = frameworkFactory.newFramework(config);
    framework.start();
    log.info("OSGi framework started");
}

void waitForFrameworkAndQuit() throws Exception {
    try {
        framework.waitForStop(0);
    } finally {
        log.info("OSGi framework stopped, exiting");
        System.exit(0);
    }
}
```

OSGI FOR MERE MORTALS

Get bundles from Maven repo

Copy to a folder using maven-dependency-plugin in launcher



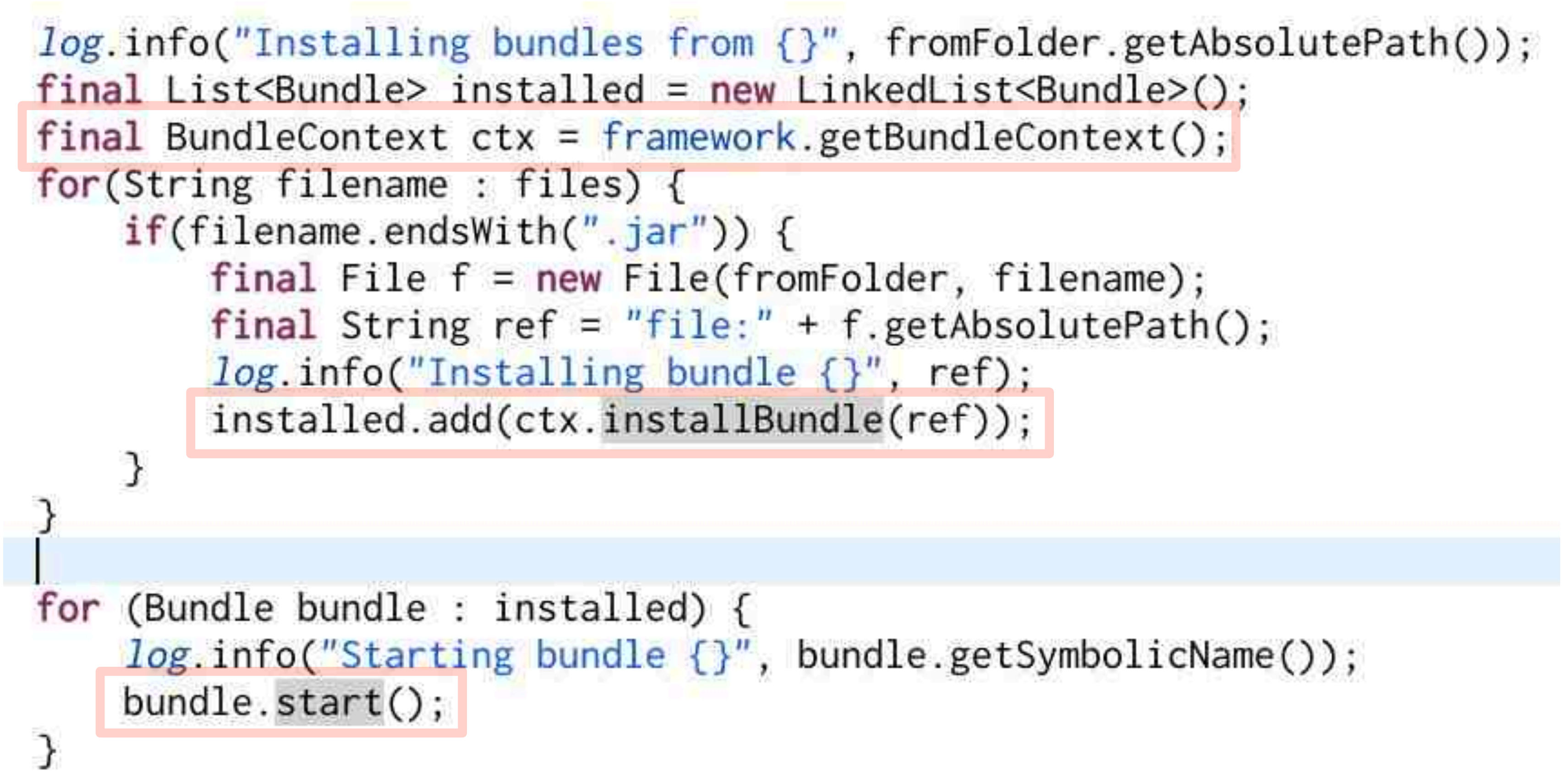
```
<execution>
    <id>copy-bundles-to-install</id>
    <goals>
        <goal>copy</goal>
    </goals>
    <configuration>
        <includeScope>provided</includeScope>
        <excludeTransitive>true</excludeTransitive>
        <outputDirectory>${project.build.directory}/bundles</outputDirectory>
        <overWriteReleases>false</overWriteReleases>
        <overWriteSnapshots>false</overWriteSnapshots>
        <artifactItems>
            <artifactItem>
                <groupId>org.apache.felix</groupId>
                <artifactId>org.apache.felix.shell</artifactId>
                <version>1.4.2</version>
            </artifactItem>
            <artifactItem>
                <groupId>org.apache.felix</groupId>
                <artifactId>org.apache.felix.shell.tui</artifactId>
                <version>1.4.1</version>
            </artifactItem>
            <artifactItem>
                <groupId>org.apache.felix</groupId>
                <artifactId>org.apache.felix.configadmin</artifactId>
                <version>1.2.8</version>
            </artifactItem>
```

OSGI FOR MERE MORTALS

Install bundles from filesystem

BundleContext.installBundle(URL)

Install all bundles first, then start all

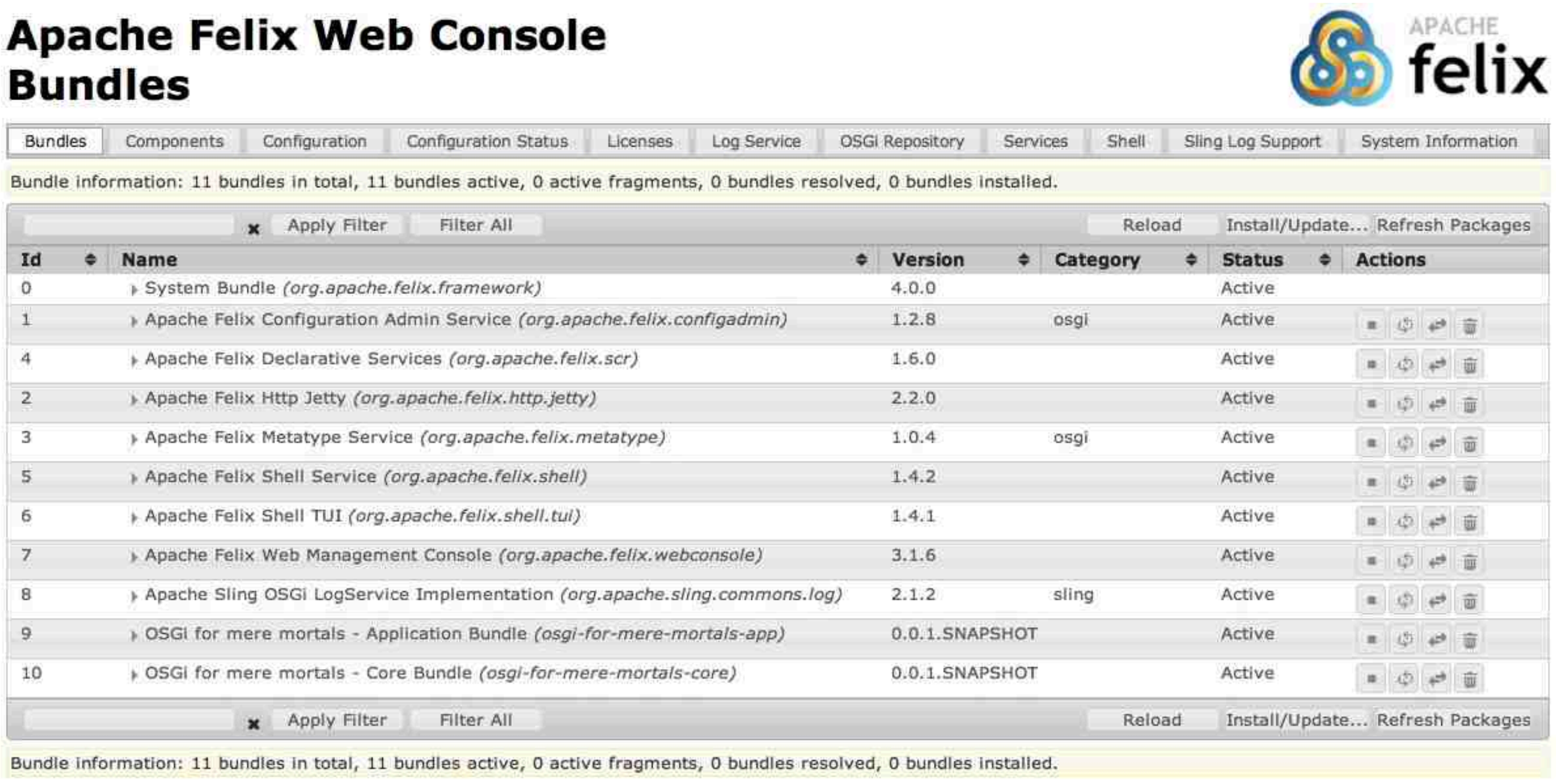


```java
log.info("Installing bundles from {}", fromFolder.getAbsolutePath());
final List<Bundle> installed = new LinkedList<Bundle>();
final BundleContext ctx = framework.getBundleContext();
for(String filename : files) {
    if(filename.endsWith(".jar")) {
        final File f = new File(fromFolder, filename);
        final String ref = "file:" + f.getAbsolutePath();
        log.info("Installing bundle {}", ref);
        installed.add(ctx.installBundle(ref));
    }
}

for (Bundle bundle : installed) {
    log.info("Starting bundle {}", bundle.getSymbolicName());
    bundle.start();
}
```

OSGI FOR MERE MORTALS

Live bundles list

From the Felix console at /system/console



Apache Felix Web Console
Bundles

Bundle information: 11 bundles in total, 11 bundles active, 0 active fragments, 0 bundles resolved, 0 bundles installed.

Id	Name	Version	Category	Status	Actions
0	System Bundle *(org.apache.felix.framework)*	4.0.0		Active	
1	Apache Felix Configuration Admin Service *(org.apache.felix.configadmin)*	1.2.8	osgi	Active	
4	Apache Felix Declarative Services *(org.apache.felix.scr)*	1.6.0		Active	
2	Apache Felix Http Jetty *(org.apache.felix.http.jetty)*	2.2.0		Active	
3	Apache Felix Metatype Service *(org.apache.felix.metatype)*	1.0.4	osgi	Active	
5	Apache Felix Shell Service *(org.apache.felix.shell)*	1.4.2		Active	
6	Apache Felix Shell TUI *(org.apache.felix.shell.tui)*	1.4.1		Active	
7	Apache Felix Web Management Console *(org.apache.felix.webconsole)*	3.1.6		Active	
8	Apache Sling OSGi LogService Implementation *(org.apache.sling.commons.log)*	2.1.2	sling	Active	
9	OSGi for mere mortals - Application Bundle *(osgi-for-mere-mortals-app)*	0.0.1.SNAPSHOT		Active	
10	OSGi for mere mortals - Core Bundle *(osgi-for-mere-mortals-core)*	0.0.1.SNAPSHOT		Active	

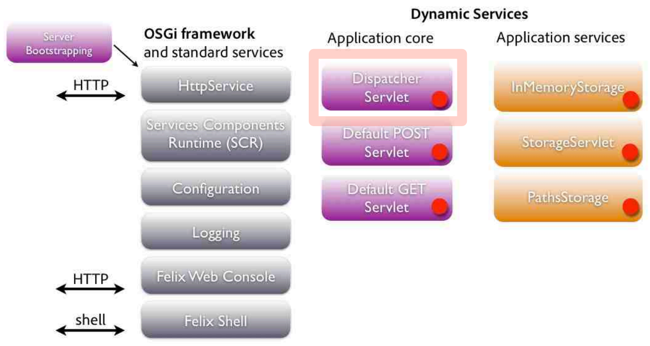
Bundle information: 11 bundles in total, 11 bundles active, 0 active fragments, 0 bundles resolved, 0 bundles installed.

We're good to go!

Our OSGi bundles

HttpService	org.apache.felix.http.jetty
Logging	org.apache.sling.commons.log
Services Components Runtime (SCR)	org.apache.felix.scr
Configuration	org.apache.felix.metatype org.apache.felix.configadmin
Felix Shell	org.apache.felix.shell org.apache.felix.shell.tui
Felix Web Console	org.apache.felix.webconsole
Application Core	osgi-for-mere-mortals-core
Application Services	osgi-for-mere-mortals-app

OSGI FOR MERE MORTALS

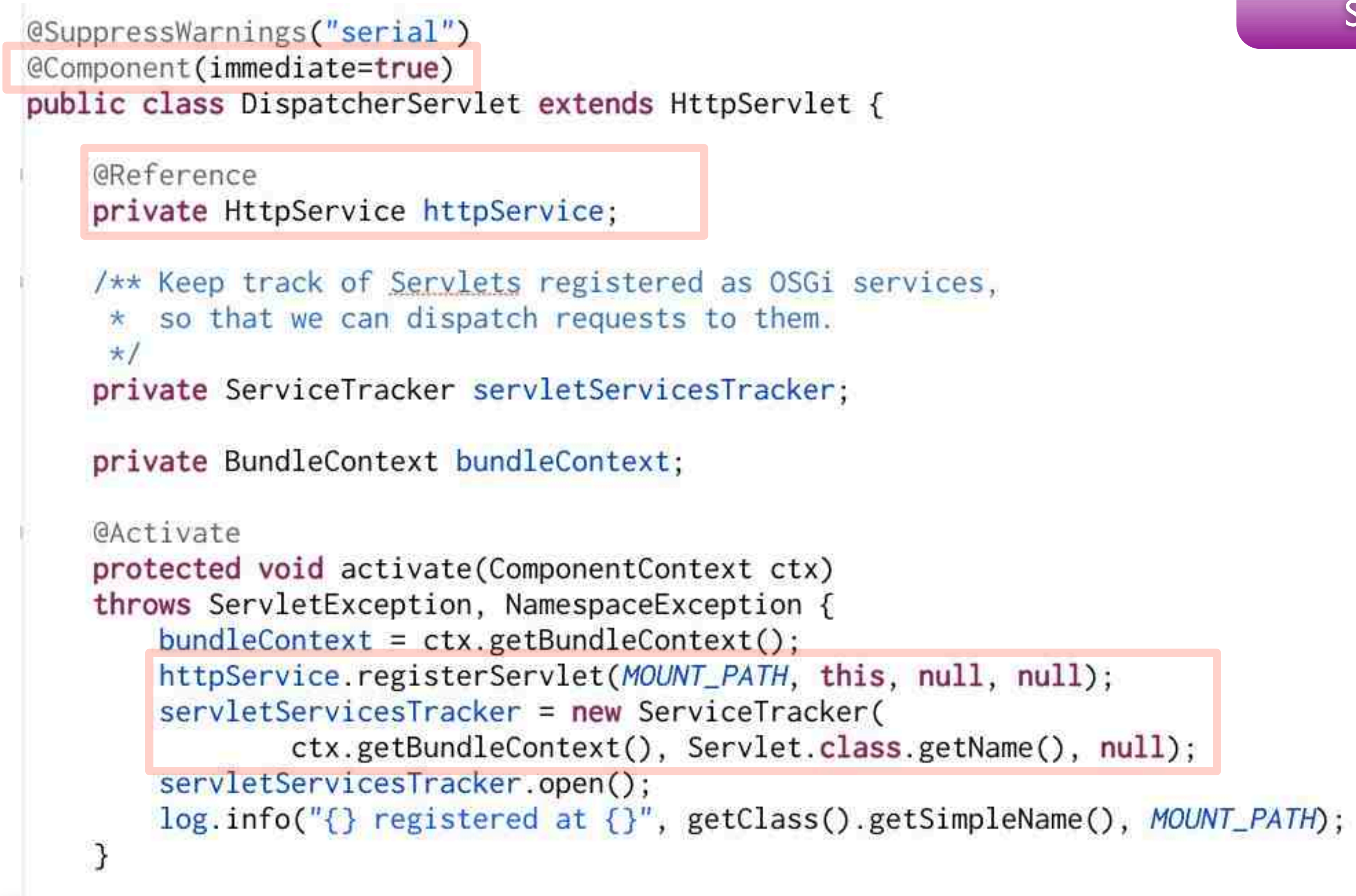


Dispatcher Servlet

DispatcherServlet Component

Register with HttpService and watch for Servlet services

Dispatcher Servlet



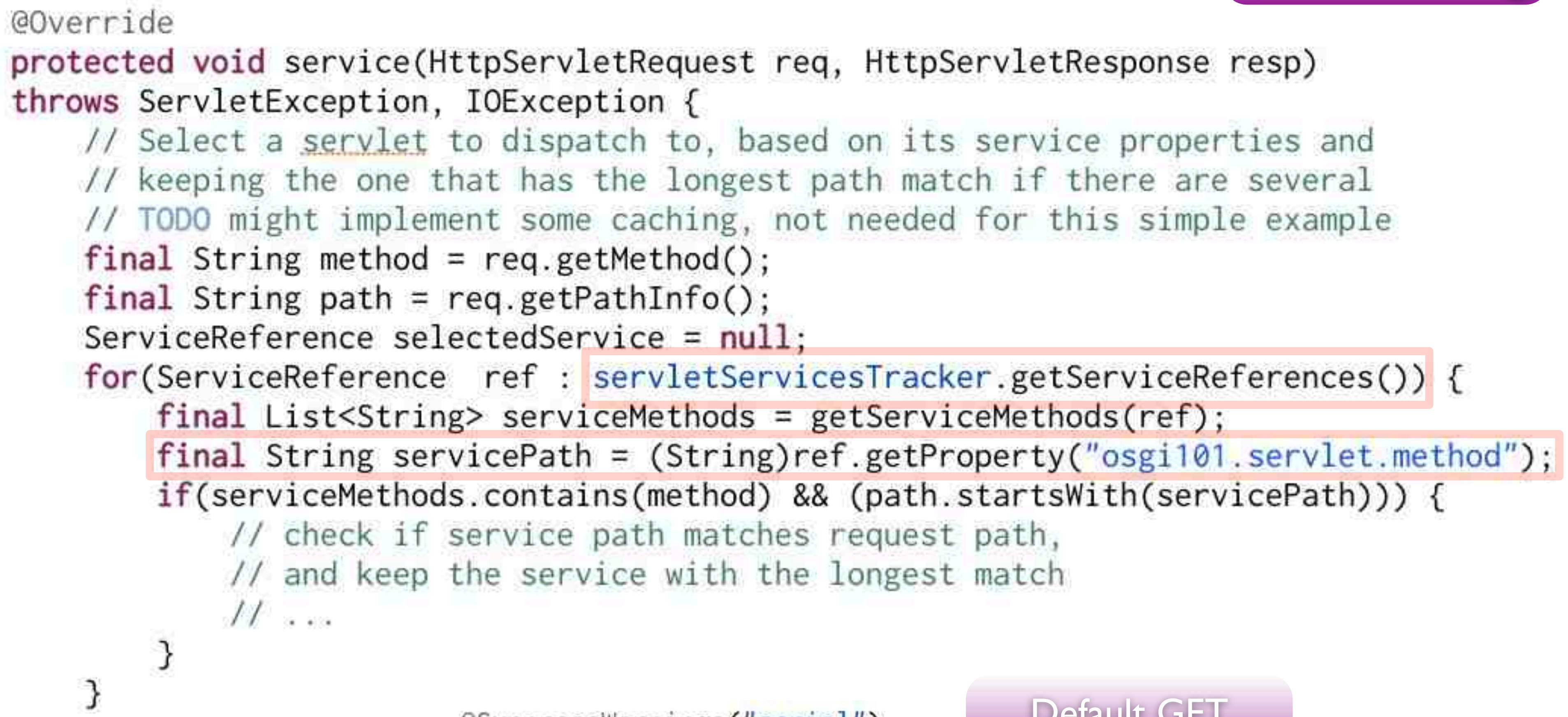
```java
@SuppressWarnings("serial")
@Component(immediate=true)
public class DispatcherServlet extends HttpServlet {

    @Reference
    private HttpService httpService;

    /** Keep track of Servlets registered as OSGi services,
     *  so that we can dispatch requests to them.
     */
    private ServiceTracker servletServicesTracker;

    private BundleContext bundleContext;

    @Activate
    protected void activate(ComponentContext ctx)
    throws ServletException, NamespaceException {
        bundleContext = ctx.getBundleContext();
        httpService.registerServlet(MOUNT_PATH, this, null, null);
        servletServicesTracker = new ServiceTracker(
                ctx.getBundleContext(), Servlet.class.getName(), null);
        servletServicesTracker.open();
        log.info("{} registered at {}", getClass().getSimpleName(), MOUNT_PATH);
    }
```

OSGI FOR MERE MORTALS

DispatcherServlet

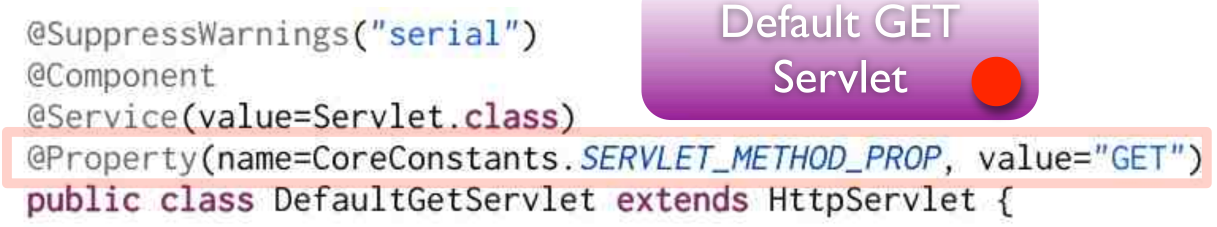
Dispatch to OSGi services which are Servlets

Dispatcher Servlet ●



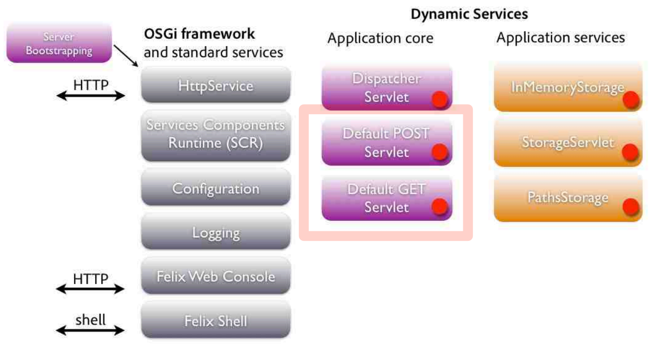
```
@Override
protected void service(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
    // Select a servlet to dispatch to, based on its service properties and
    // keeping the one that has the longest path match if there are several
    // TODO might implement some caching, not needed for this simple example
    final String method = req.getMethod();
    final String path = req.getPathInfo();
    ServiceReference selectedService = null;
    for(ServiceReference  ref : servletServicesTracker.getServiceReferences()) {
        final List<String> serviceMethods = getServiceMethods(ref);
        final String servicePath = (String)ref.getProperty("osgi101.servlet.method");
        if(serviceMethods.contains(method) && (path.startsWith(servicePath))) {
            // check if service path matches request path,
            // and keep the service with the longest match
            // ...
        }
    }
```

Default GET Servlet ●



```
@SuppressWarnings("serial")
@Component
@Service(value=Servlet.class)
@Property(name=CoreConstants.SERVLET_METHOD_PROP, value="GET")
public class DefaultGetServlet extends HttpServlet {
```

OSGI FOR MERE MORTALS

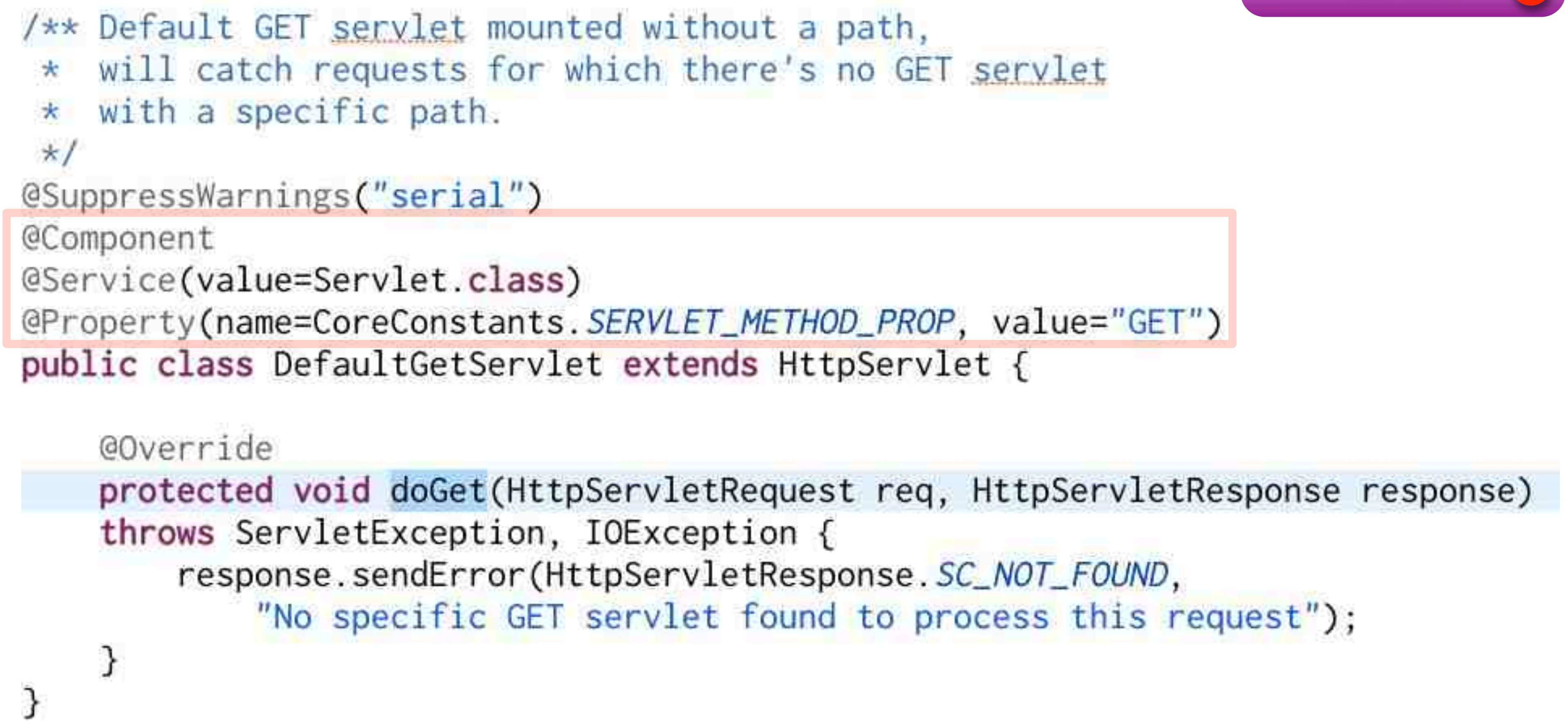


Default GET Servlet

Default GET servlet

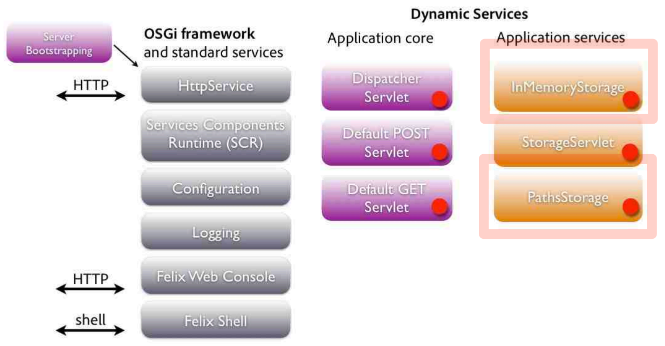
Just a servlet, with some SCR annotations

Default GET Servlet



```java
/** Default GET servlet mounted without a path,
 *  will catch requests for which there's no GET servlet
 *  with a specific path.
 */
@SuppressWarnings("serial")
@Component
@Service(value=Servlet.class)
@Property(name=CoreConstants.SERVLET_METHOD_PROP, value="GET")
public class DefaultGetServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse response)
    throws ServletException, IOException {
        response.sendError(HttpServletResponse.SC_NOT_FOUND,
            "No specific GET servlet found to process this request");
    }
}
```

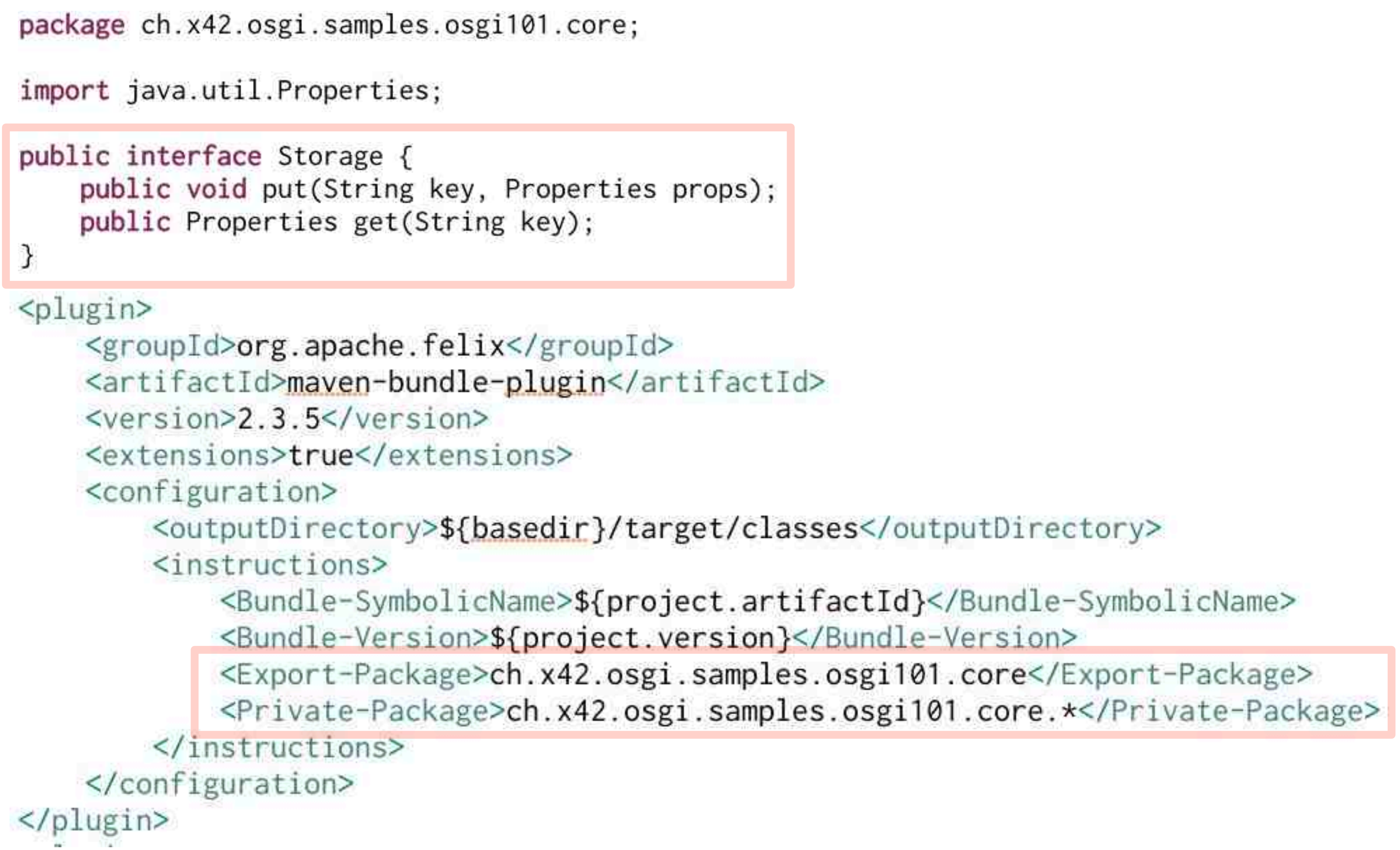
OSGI FOR MERE MORTALS



Storage Services

Storage service interface

Defined in the core bundle, package is exported

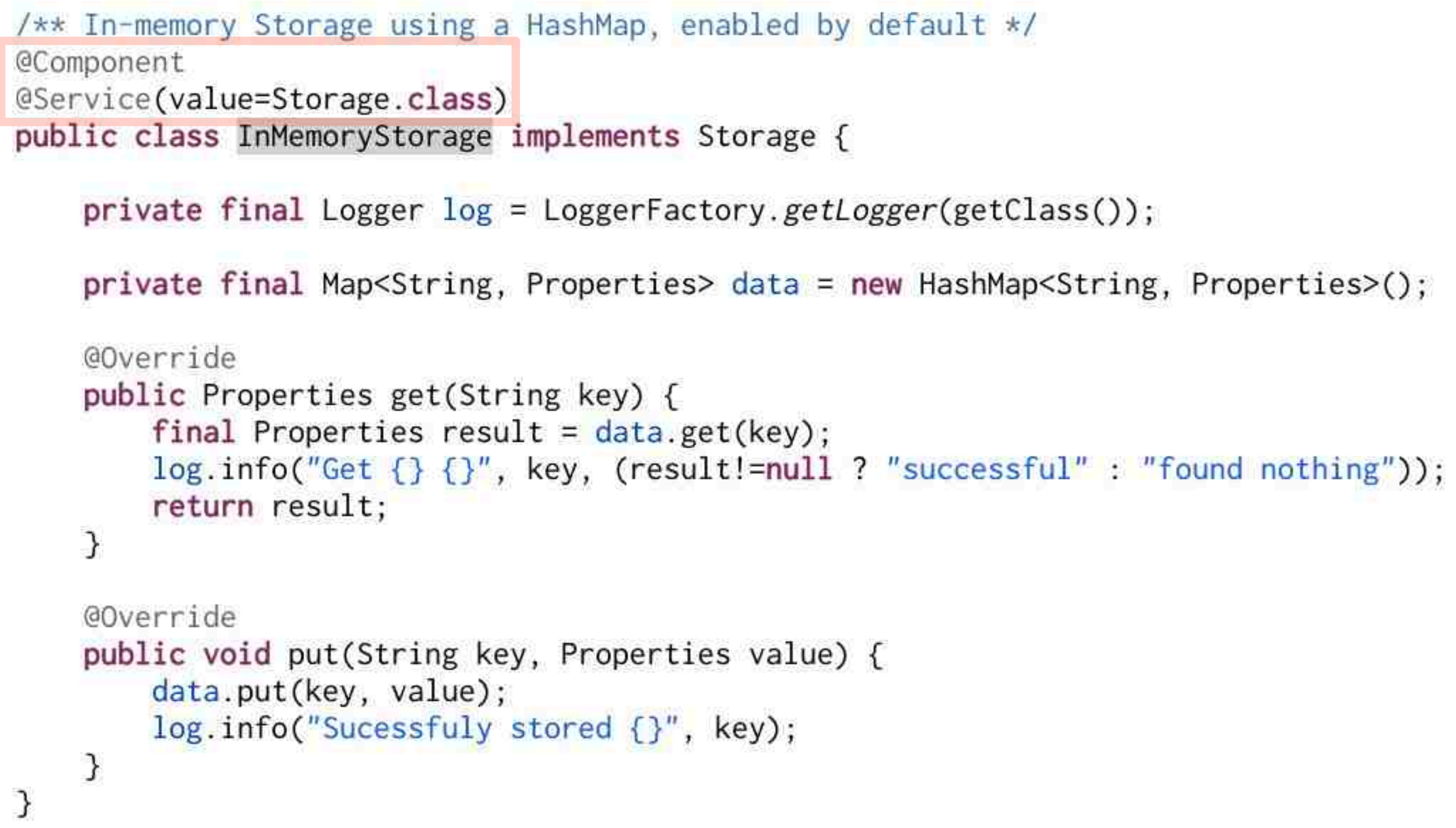


```java
package ch.x42.osgi.samples.osgi101.core;

import java.util.Properties;

public interface Storage {
    public void put(String key, Properties props);
    public Properties get(String key);
}
```

```xml
<plugin>
    <groupId>org.apache.felix</groupId>
    <artifactId>maven-bundle-plugin</artifactId>
    <version>2.3.5</version>
    <extensions>true</extensions>
    <configuration>
        <outputDirectory>${basedir}/target/classes</outputDirectory>
        <instructions>
            <Bundle-SymbolicName>${project.artifactId}</Bundle-SymbolicName>
            <Bundle-Version>${project.version}</Bundle-Version>
            <Export-Package>ch.x42.osgi.samples.osgi101.core</Export-Package>
            <Private-Package>ch.x42.osgi.samples.osgi101.core.*</Private-Package>
        </instructions>
    </configuration>
</plugin>
```

Storage service#1: in memory

Active by default



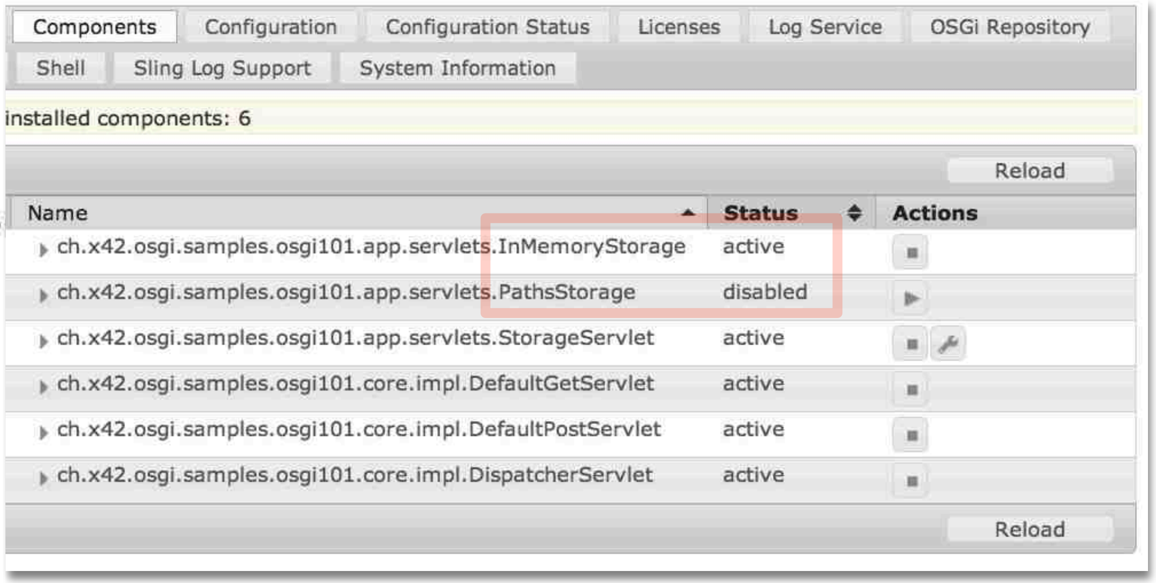
```java
/** In-memory Storage using a HashMap, enabled by default */
@Component
@Service(value=Storage.class)
public class InMemoryStorage implements Storage {

    private final Logger log = LoggerFactory.getLogger(getClass());

    private final Map<String, Properties> data = new HashMap<String, Properties>();

    @Override
    public Properties get(String key) {
        final Properties result = data.get(key);
        log.info("Get {} {}", key, (result!=null ? "successful" : "found nothing"));
        return result;
    }

    @Override
    public void put(String key, Properties value) {
        data.put(key, value);
        log.info("Sucessfuly stored {}", key);
    }
}
```

OSGI FOR MERE MORTALS

Storage service#2: just paths

Alternate service, inactive by default, can be activated in console

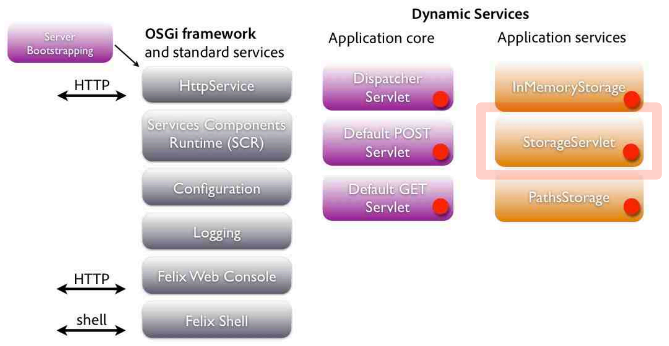
```
/** Storage that stores paths only, disabled by default,
 *  used to demonstrate switching services dynamically */
@Component(enabled=false)
@Service(value=Storage.class)
public class PathsStorage implements Storage {

    private final Set<String> paths = new HashSet<String>();

    @Override
    public Properties get(String key) {
        final Properties props = new Properties();
        int index=0;
        for(String path : paths) {
            props.put("path." + index++, path);
        }
        return props;
    }

    @Override
    public void put(String key, Prope
        paths.add(key);
    }
}
```



installed components: 6

Name	Status	Actions
ch.x42.osgi.samples.osgi101.app.servlets.InMemoryStorage	active	■
ch.x42.osgi.samples.osgi101.app.servlets.PathsStorage	disabled	▶
ch.x42.osgi.samples.osgi101.app.servlets.StorageServlet	active	■
ch.x42.osgi.samples.osgi101.core.impl.DefaultGetServlet	active	■
ch.x42.osgi.samples.osgi101.core.impl.DefaultPostServlet	active	■
ch.x42.osgi.samples.osgi101.core.impl.DispatcherServlet	active	■

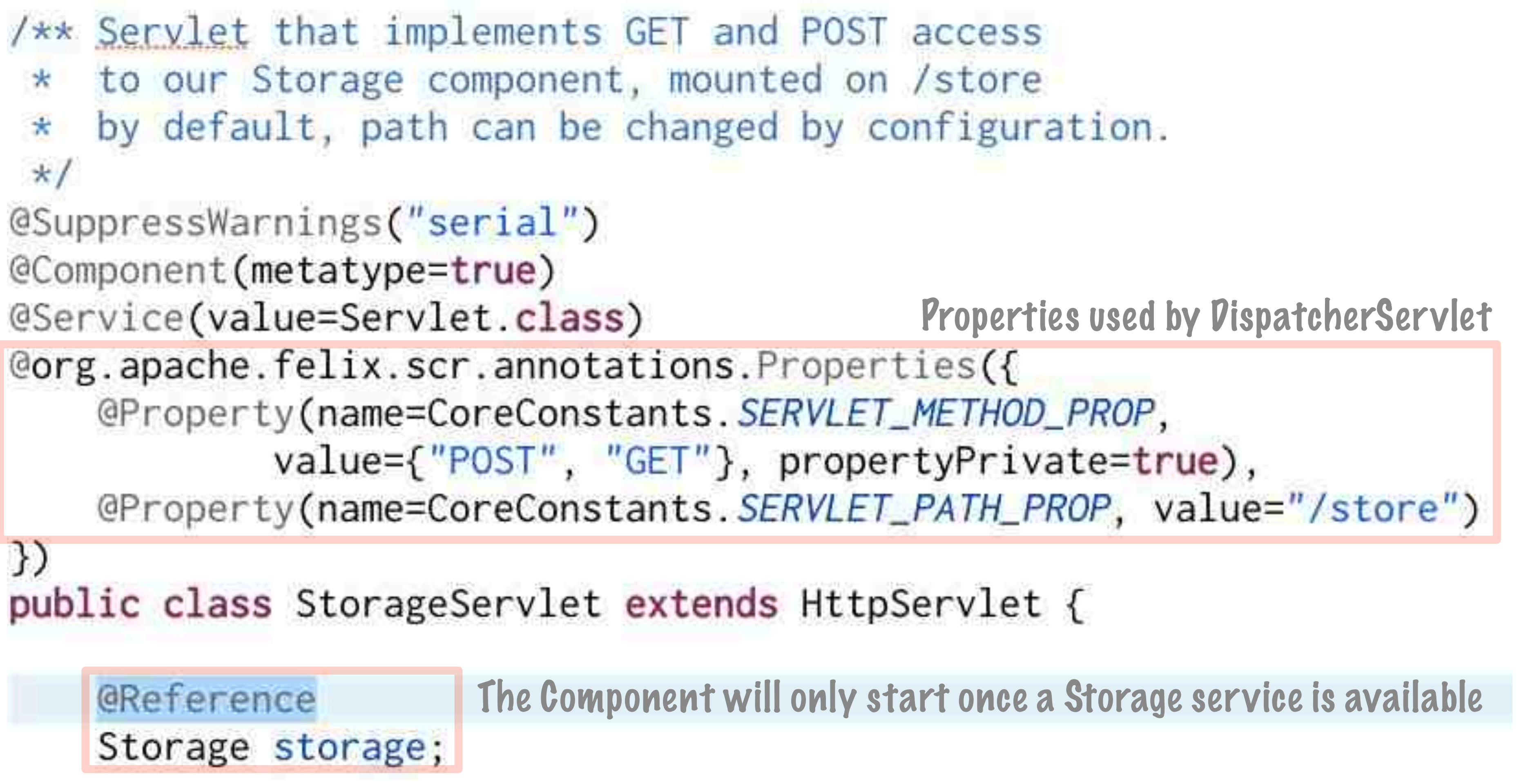
OSGI FOR MERE MORTALS



Storage Servlet

StorageServlet

Uses @Reference to get a Storage service

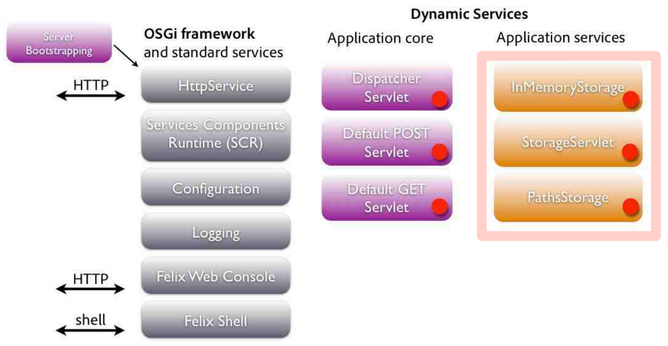


```
/** Servlet that implements GET and POST access
 *  to our Storage component, mounted on /store
 *  by default, path can be changed by configuration.
 */
@SuppressWarnings("serial")
@Component(metatype=true)
@Service(value=Servlet.class)
@org.apache.felix.scr.annotations.Properties({
    @Property(name=CoreConstants.SERVLET_METHOD_PROP,
            value={"POST", "GET"}, propertyPrivate=true),
    @Property(name=CoreConstants.SERVLET_PATH_PROP, value="/store")
})
public class StorageServlet extends HttpServlet {

    @Reference
    Storage storage;
```

Properties used by DispatcherServlet

The Component will only start once a Storage service is available

OSGI FOR MERE MORTALS

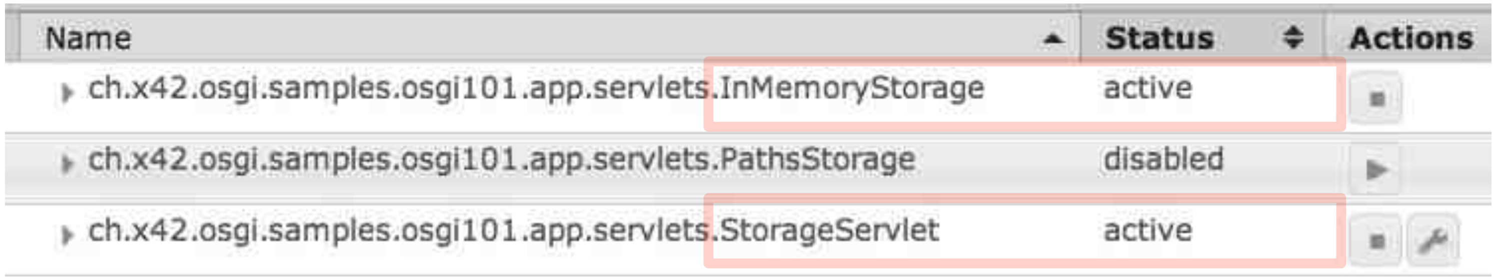


Alternate Storage

demo

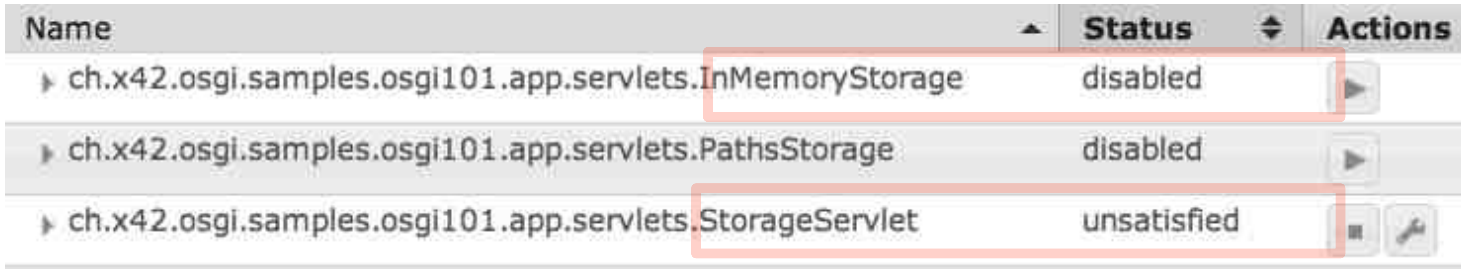
Replace In-memory with paths Storage

By disabling one and enabling another Component



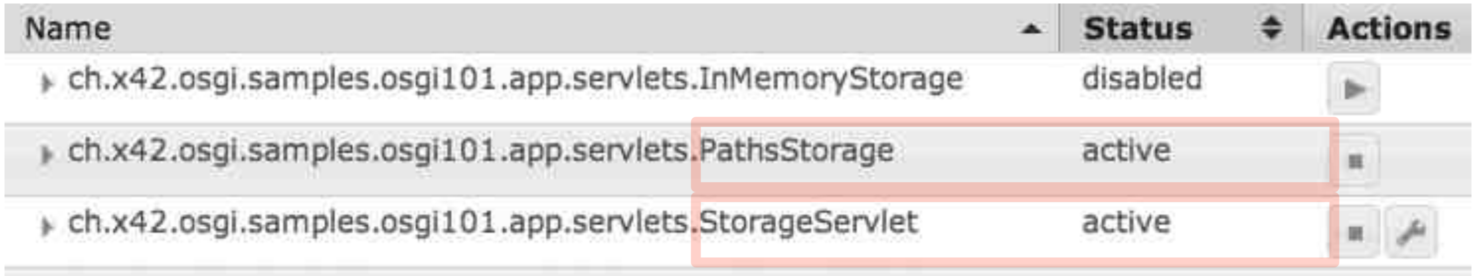
Name	Status	Actions
ch.x42.osgi.samples.osgi101.app.servlets.InMemoryStorage	active	
ch.x42.osgi.samples.osgi101.app.servlets.PathsStorage	disabled	
ch.x42.osgi.samples.osgi101.app.servlets.StorageServlet	active	

1) disable inMemoryStorage -> StorageServlet stops



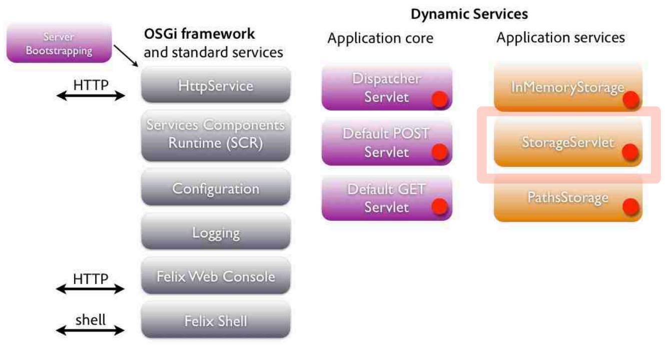
Name	Status	Actions
ch.x42.osgi.samples.osgi101.app.servlets.InMemoryStorage	disabled	
ch.x42.osgi.samples.osgi101.app.servlets.PathsStorage	disabled	
ch.x42.osgi.samples.osgi101.app.servlets.StorageServlet	unsatisfied	

2) enable PathsStorage -> StorageServlet is back



Name	Status	Actions
ch.x42.osgi.samples.osgi101.app.servlets.InMemoryStorage	disabled	
ch.x42.osgi.samples.osgi101.app.servlets.PathsStorage	active	
ch.x42.osgi.samples.osgi101.app.servlets.StorageServlet	active	

OSGI FOR MERE MORTALS

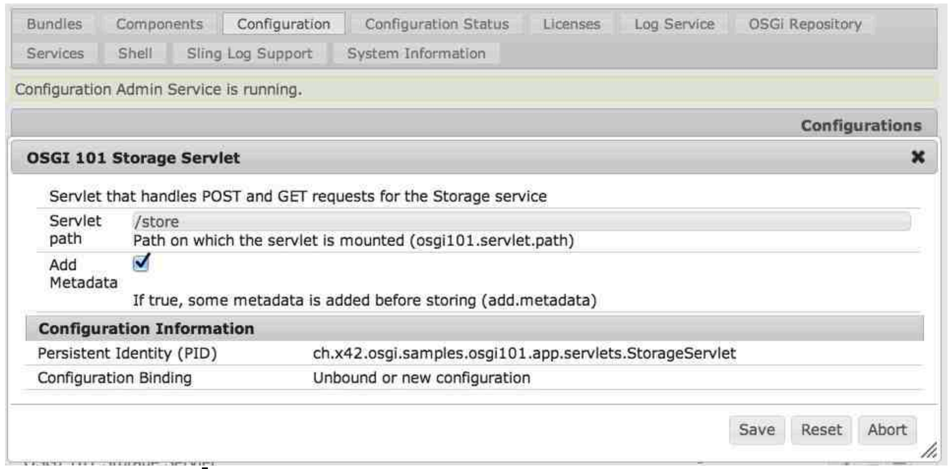


Configuration

demo

Configuring the StorageServlet

ConfigurationAdmin service, console and annotations



Configuration Admin Service is running.

Configurations

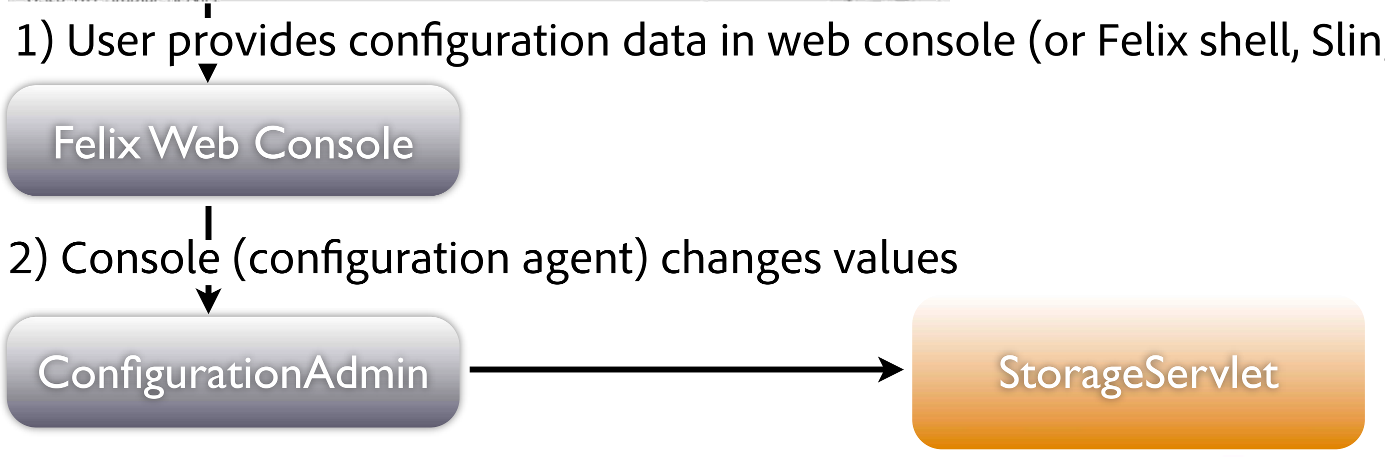
OSGI 101 Storage Servlet

Servlet that handles POST and GET requests for the Storage service

Servlet path	/store	Path on which the servlet is mounted (osgi101.servlet.path)
Add Metadata	✓	If true, some metadata is added before storing (add.metadata)

Configuration Information

Persistent Identity (PID)	ch.x42.osgi.samples.osgi101.app.servlets.StorageServlet
Configuration Binding	Unbound or new configuration



1) User provides configuration data in web console (or Felix shell, Sling Installer, ...)

2) Console (configuration agent) changes values

3) ConfigurationAdmin deactivates and reactivates Component with new Config

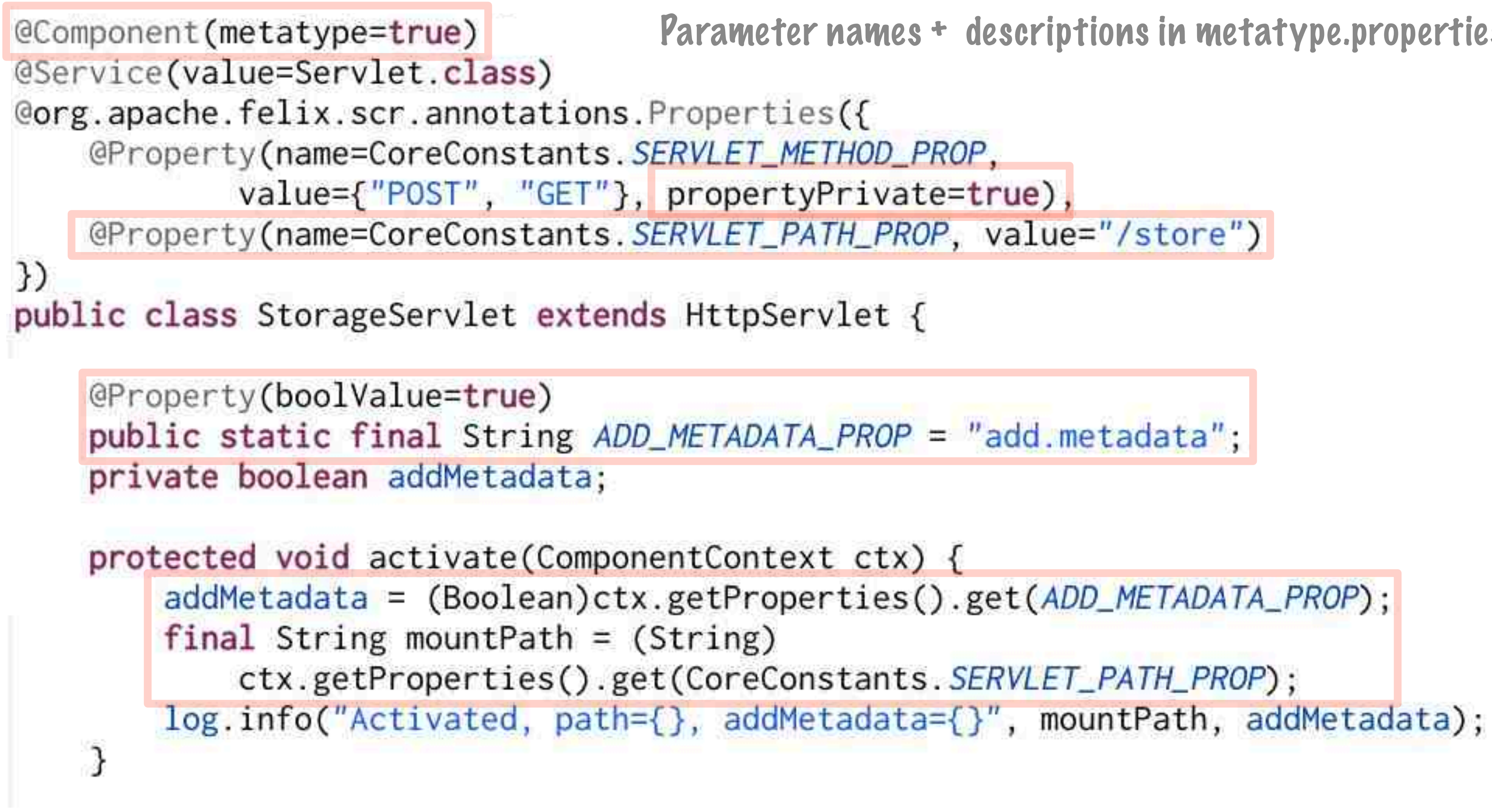
```
@Property(boolValue=true)
public static final String ADD_METADATA_PROP = "add.metadata";
private boolean addMetadata;
@Property(name=CoreConstants.SERVLET_PATH_PROP, value="/store")
```

0) Component has metatype=true and some non-private @Property annotations

OSGI FOR MERE MORTALS

StorageServlet Configuration code

Annotations + read values in activate()



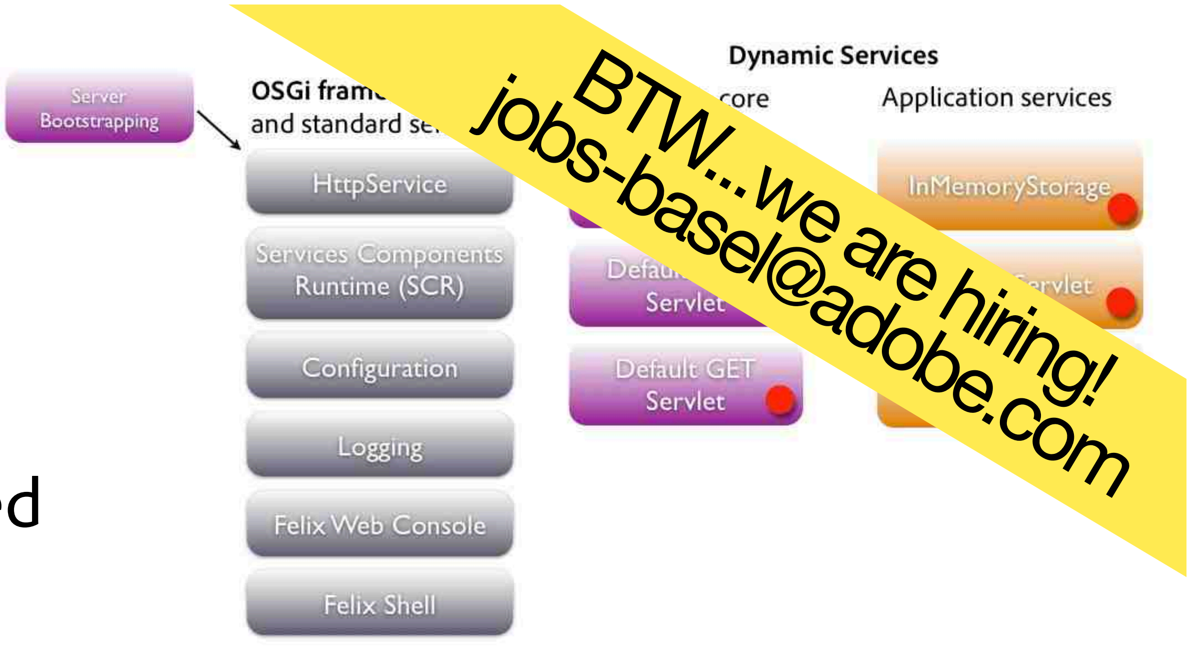
Parameter names + descriptions in metatype.properties

```java
@Component(metatype=true)
@Service(value=Servlet.class)
@org.apache.felix.scr.annotations.Properties({
    @Property(name=CoreConstants.SERVLET_METHOD_PROP,
            value={"POST", "GET"}, propertyPrivate=true),
    @Property(name=CoreConstants.SERVLET_PATH_PROP, value="/store")
})
public class StorageServlet extends HttpServlet {

    @Property(boolValue=true)
    public static final String ADD_METADATA_PROP = "add.metadata";
    private boolean addMetadata;

    protected void activate(ComponentContext ctx) {
        addMetadata = (Boolean)ctx.getProperties().get(ADD_METADATA_PROP);
        final String mountPath = (String)
            ctx.getProperties().get(CoreConstants.SERVLET_PATH_PROP);
        log.info("Activated, path={}, addMetadata={}", mountPath, addMetadata);
    }
```

OSGI FOR MERE MORTALS

Conclusions

Conclusions



BTW...we are hiring!
jobs-basel@adobe.com

Powerful out of the box infrastructure, not much code to write.

Modular component-based architecture.

Dynamic (re)configuration, both in terms of components and parameters.

OSGi materializes the component-based programming vision in Java, and using it is not rocket science!

Code at github.com/bdelacretaz/OSGi-for-mere-mortals
Your host today: @bdelacretaz, grep.codeconsult.ch