ORACLE®

JavaOne™

ORACLE®

Keep Learning with Oracle University

ORACLE®

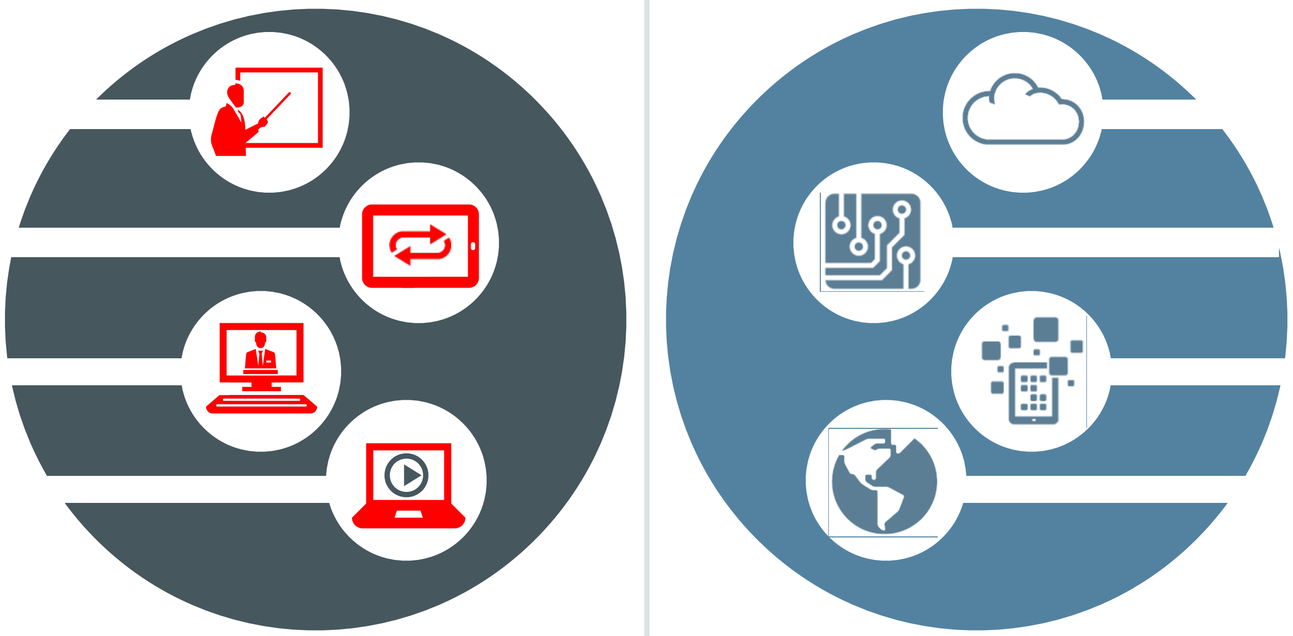
UNIVERSITY

Classroom Training

Learning Subscription

Live Virtual Class

Training On Demand



Cloud

Technology

Applications

Industries

education.oracle.com

Session Surveys

Help us help you!!

- Oracle would like to invite you to take a moment to give us your session feedback. Your feedback will help us to improve your conference.
- Please be sure to add your feedback for your attended sessions by using the Mobile Survey or in Schedule Builder.

New and Noteworthy in Jersey 2

Petr Janouch
Software Developer
Oracle, Application Server Group
October 28, 2015

JavaOne
ORACLE

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Goals of The Presentation

- To show new additions to Jersey project
- To demonstrate some lesser-known features
- To show where the project is heading

Jersey 2 update

- Jersey 2 provides reference implementation of JAX-RS 2.0
- Included in GlassFish 4.x
- And in WebLogic 12.1.3 (Jersey 2 as a shared library)
- And in WebLogic 12.2.1
- Provides ouf-the-box support for other containers
- 2.22.1 is the current version

Notable additional features

- Integration with various HTTP containers and client transports
- Support for SSE
- MVC view templates
- Reactive/Async Client ★
- Security
- Test Framework
- Monitoring and Tracing ★
- Various data bindings

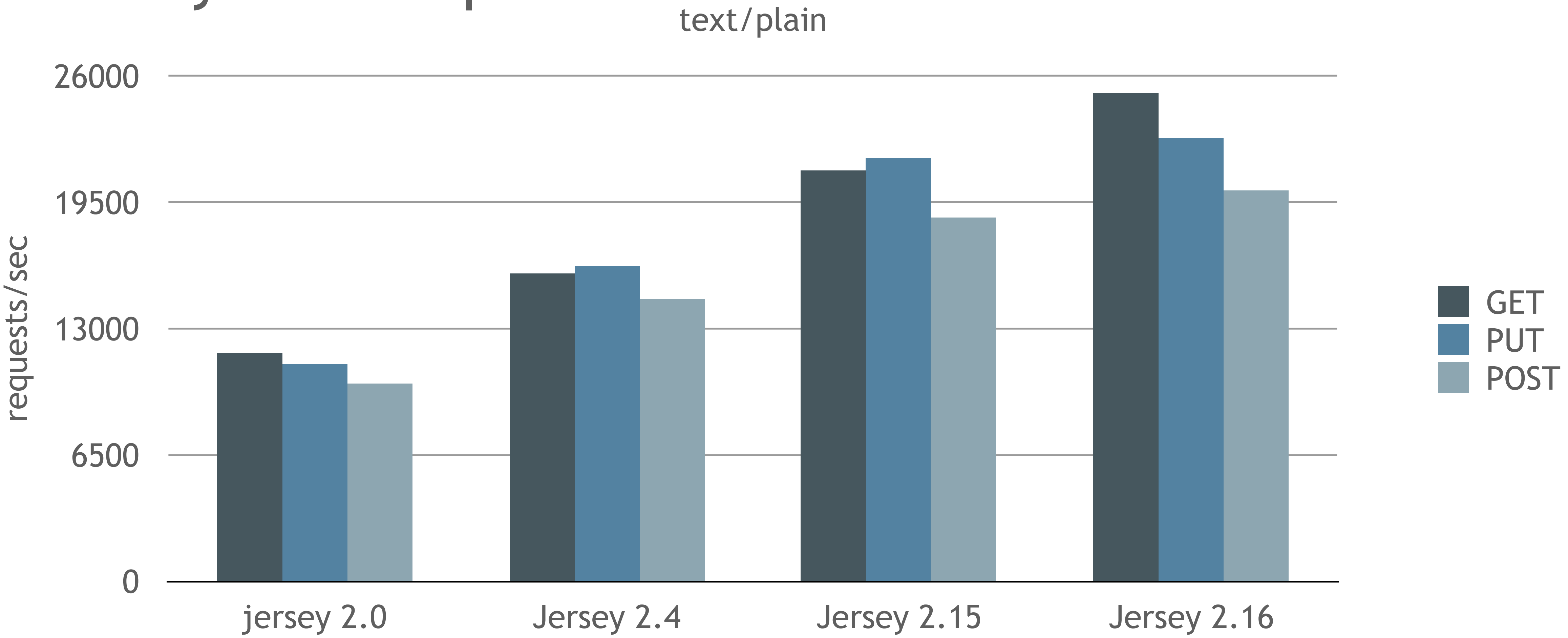
Agenda

1. Performance improvements
2. Weld SE support
3. Client on Android
4. Dynamic reloading example
5. Jersey 3.x

Agenda

1. Performance improvements
2. Weld SE support
3. Client on Android
4. Dynamic reloading example
5. Jersey 3.x

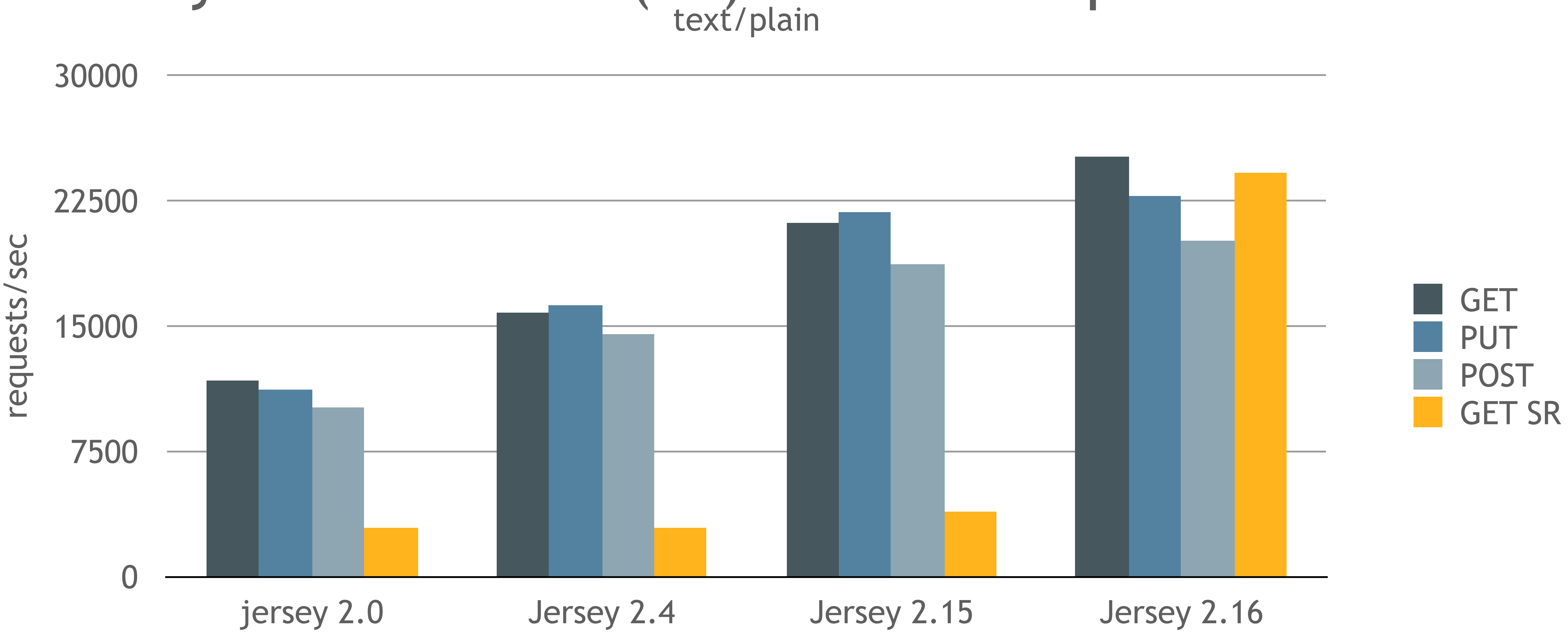
Jersey core improvements



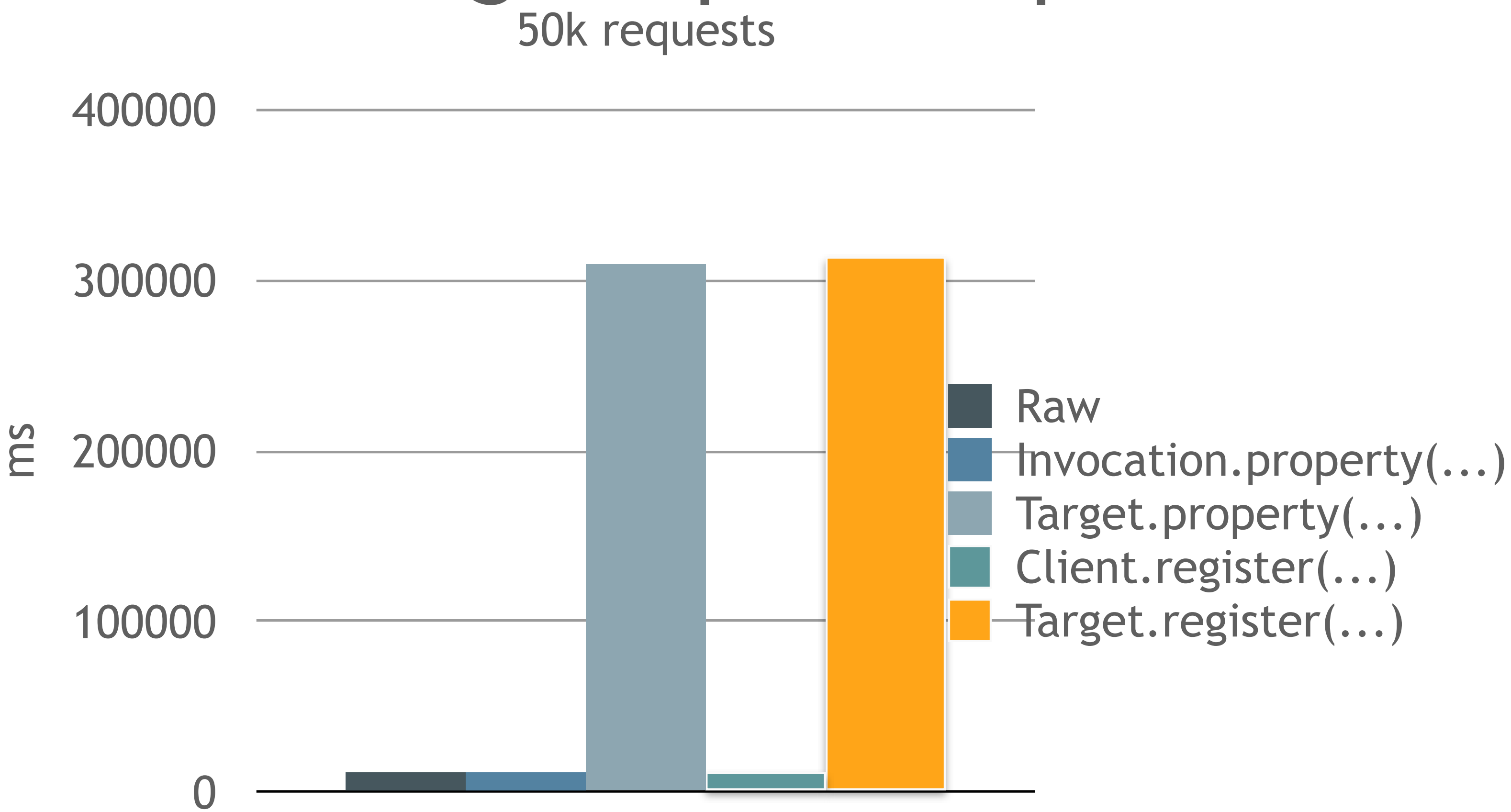
Subresource reminder

```
@Path(“resource”)
public class Resource {

    @GET
    @Path("hello")
    public String getHello() {
        return "Hello from Sub-Resource Method";
    }

    @Path("subresource")
    public SubResource getSubResource() {
        return new SubResource();
    }
}

public class SubResource {
    @GET
    public String getHello() {
        return "Hello, from Sub-Resource Locator";
    }
}
```

Jersey sub-resource (SR) locators improvements



Client usage impact on performance (Noteworthy)



```
c = ClientBuilder.newClient()
t = c.target(“www.acme.com”)

// raw:
t.request().get();

// Invocation.property:
t.request.property(…)

// Target.property(…)
t.property(…)
```

Agenda

1. Performance improvements
2. Weld SE support
3. Client on Android
4. Dynamic reloading example
5. Jersey 3.x

Weld (CDI) SE support

- CDI 1.2 linked to Java EE container
- Weld SE
  - DI with qualifiers and alternatives
  - @ApplicationScoped, @Dependent, @Singleton scopes
  - Interceptors and decorators
  - Stereotypes
- Jersey adds @RequestScoped

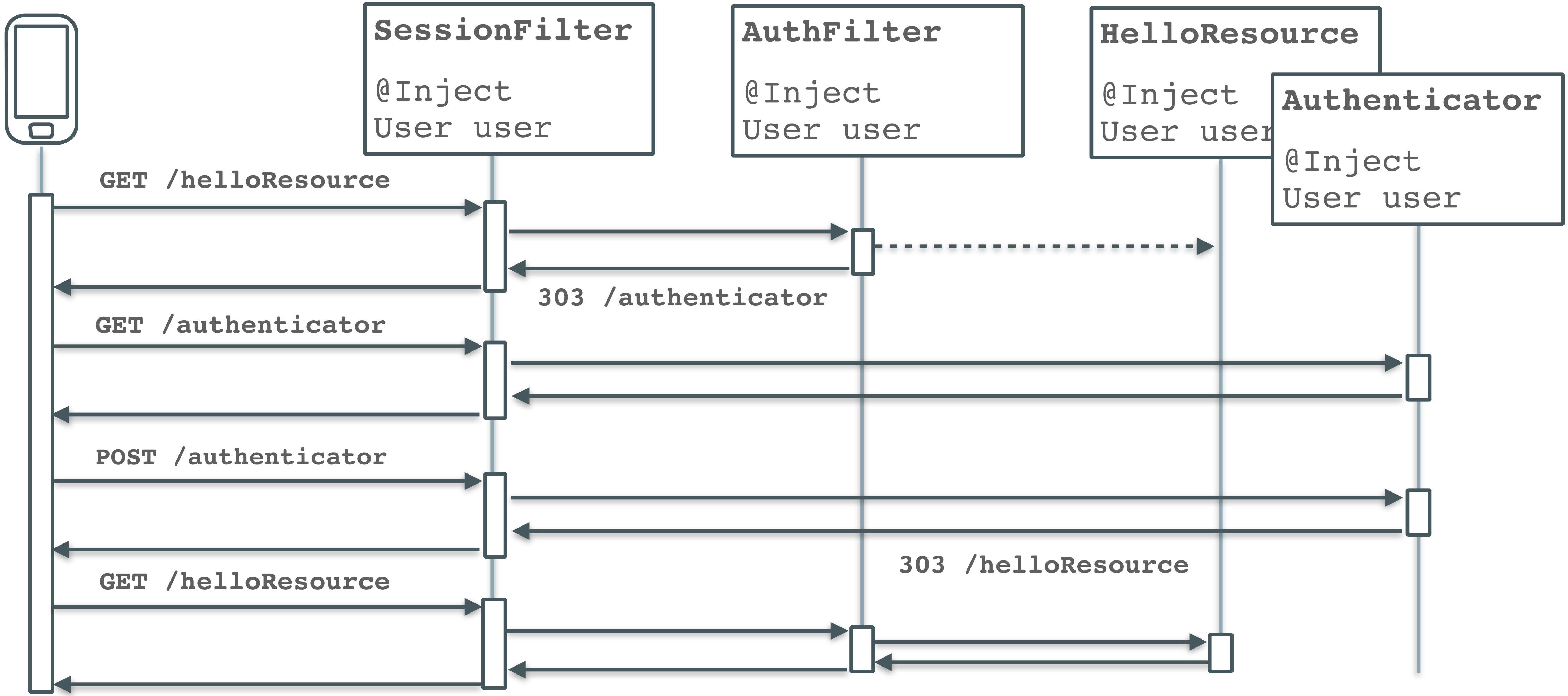
Weld SE integration

```
Weld weld = new Weld();
weld.initialize();

HttpServer httpServer = GrizzlyHttpServerFactory.createHttpServer(myUri, new
MyApp(), true);

public MyApp() extends ResourceConfig {super(HelloResource.class);}

Path(“hello”)
@RequestScoped
public class HelloReource {

    @GET
    public String sayHello() {
        return “Hello”;
    }
}
```

Weld SE demo

- Integration with various HTTP containers and client transports ★
- Support for SSE
- MVC view templates ★
- Reactive/Async Client
- Security ★
- Test Framework
- Monitoring and Tracing
- Various data bindings

Weld SE demo


SessionFilter
@Inject
User user
AuthFilter
@Inject
User user
HelloResource
@Inject
User user
Authenticator
@Inject
User user
GET /helloResource
303 /authenticator
GET /authenticator
POST /authenticator
303 /helloResource
GET /helloResource

Weld SE demo

```
@RequestScoped
public class User {
    private String name;
    private boolean authenticated = false;

    . . .
}

@RequestScoped
public class OriginalDestination {
    private URI uri;

    . . .
}

@NameBinding
@Retention(RetentionPolicy.RUNTIME)
public @interface FormAuthenticated {}
```

Weld SE demo

```
@Priority(100)
public class SessionFilter implements ContainerRequestFilter, ContainerResponseFilter {

    @Inject private User user;
    @Inject private OriginalDestination originalDestination;
    @Inject private Session session;

    public void filter(ContainerRequestContext requestContext) {
        User.fillFromSession(user, session);
        OriginalDestination.fillFromSession(originalDestination, session);
    }

    public void filter(ContainerRequestContext requestContext, ContainerResponseContext responseContext){
        if (user.isAuthenticated()) {
            user.storeInSession(session);
        } else {
            if (originalDestination.getUri() != null) {
                originalDestination.storeInSession(session);
            }
        }
    }
}
```

Weld SE demo

```
@Priority(200)
@FormAuthenticated
public class AuthenticationFilter implements ContainerRequestFilter {

    @Inject private User user;
    @Inject private OriginalDestination originalDestination;

    public void filter(final ContainerRequestContext requestContext) {
        if (!user.isAuthenticated()) {
            originalDestination.setUri(
                requestContext.getUriInfo().getRequestUri());
            requestContext.abortWith(Response.seeOther(
                UriBuilder.fromResource(Authenticator.class).build())
                .build());
        }
    }
}
```

Weld SE demo

```
@RequestScoped
@Path("/hello")
public class HelloResource {

    @Inject private User user;

    @FormAuthenticated
    @GET
    @Produces(MediaType.TEXT_HTML)
    public Viewable sayHello() {
        Map<String, String> binding = new HashMap<>();
        binding.put("user", user.getName());
        return new Viewable("/freemarker/hello", binding);
    }
}
```

Weld SE demo

```
@ApplicationScoped
@Path("authentication")
public class Authenticator {

    @Inject private OriginalDestination originalDestination;
    @Inject private User currentUser;

    @GET
    @Produces(MediaType.TEXT_HTML)
    public Viewable getLoginPage() {
        return new Viewable("/freemarker/authentication");
    }

    @POST
    @Produces(MediaType.TEXT_HTML)
    public Object handleLogin(@FormParam("username") String userName, @FormParam("password") String password) {
        if (authenticate(userName, password)) {
            return Response.seeOther(originalDestination.getUri()).build();
        }

        Map<String, Object> binding = new HashMap<>();
        binding.put("errorMessage", "Invalid username and password combination");
        return new Viewable("/freemarker/authentication", binding);
    }
}
```

Weld SE demo

```
public class MyApp extends ResourceConfig {

    public MyApp() {
        super(Authenticator.class, HelloResource.class,
            AuthenticationFilter.class, SessionFilter.class);
        register(FreemarkerMvcFeature.class);
    }

    public static void main(String[] args) throws Exception {
        Weld weld = new Weld();
        weld.initialize();

        HttpServer httpServer = GrizzlyHttpServerFactory
            .createHttpServer("http://localhost:8080", new MyApp(), false);
       httpServer.start();
        System.in.read();
    }
}
```

Weld SE demo summary

- https://github.com/PetrJanouch/JavaOne2015-Weld-SE-demo

Agenda

1 Performance improvements

2 Weld se support

3 Client on Android

4 Dynamic reloading example

5 Jersey 3.x

Jersey client primer

```
Client client = ClientBuilder.newClient(new ClientConfig()
            .register(MyClientResponseFilter.class)
            .register(new AnotherClientFilter()));

String entity = client.target("http://example.com/rest")
            .register(FilterForExampleCom.class)
            .path("resource/helloworld")
            .queryParam("greeting", "Hi World!")
            .request(MediaType.TEXT_PLAIN_TYPE)
            .header("some-header", "true")
            .get(String.class);
```

Jersey client - features

- Fluent API
- Many connectors (Grizzly, Jetty, Apache, …)
- Secure (SSL, Digest, Basic, OAuth, …)
- Various data bindings
- Filters
- Reactive extensions

Jersey client on Android

- JAX-B refactored from core into a separate module (in 2.16)
- But some references to JDK dark corners still remain:

```
$jdeps -P jersey-client-app-with-dependencies
    . . .
      -> javax.activation                                   Full JRE
    . . .
      -> javax.imageio                                      Full JRE
      -> javax.imageio.spi                                  Full JRE
      -> javax.imageio.stream                               Full JRE
    . . .
```

Jersey client on Android - workaround

```
public static class AndroidFriendlyFeature implements Feature{

  @Override
  public boolean configure(FeatureContext context) {
      context.register(new AbstractBinder() {
          @Override
          protected void configure() {
              addUnbindFilter(new Filter() {
                  @Override
                  public boolean matches(Descriptor d) {
                      String implClass = d.getImplementation();
                      return implClass.startsWith(
                        "org.glassfish.jersey.message.internal.DataSource")
                       || implClass.startsWith(
                        "org.glassfish.jersey.message.internal.RenderedImage");
                  }
              });
          }
      });
      return true;
  }
}

client = ClientBuilder.newClient().register(AndroidFriendlyFeature.class);
```

Agenda

1. Performance improvements
2. Weld se support
3. Client on Android
4. Dynamic reloading example
5. Jersey 3.x

Dynamic reloading example

- Jersey application can be forced to reload itself
- Used mainly for updating configuration
- Can be used to speed up development? Yes

Dynamic reloading example

```
ResourceConfig resourceConfig = createResourceConfig(new File(configFileName));
resourceConfig.registerInstances(new ContainerLifecycleListener() {

    public void onStartup(final Container container) {
        this.container = container;
        Timer t = new Timer(true);
        t.schedule(new FileCheckTask(), 0);
    }
}

private void reloadApp(final File configFile) {
    ResourceConfig rc = createResourceConfig(configFile);
    this.container.reload(rc);
}
```

Dynamic reloading example

```
public class AppClassLoader extends ClassLoader {

    private final Map<String, ClassFile> classFiles = new HashMap<>();

    @Override
    public Class<?> loadClass(String name) throws ClassNotFoundException {
        /* we are cheating here, the parent already has the class,
           but we prefer our bytecode to be used. */
        ClassFile cc = classFiles.get(name);
        if (cc == null) {
            return super.loadClass(name);
        }
        byte[] byteCode = cc.getByteCode();
        return defineClass(name, byteCode, 0, byteCode.length);
    }
}
```

Dynamic reloading example

```
Class<?> compile(String className, SimpleJavaFileObject sourceCode){
    ClassFile classFile = new ClassFile(className);
    List<SimpleJavaFileObject> compilationUnits = Arrays.asList(sourceCode);
    AppClassLoader cl = new AppClassLoader(
    Thread.currentThread().getContextClassLoader());
    FileManager fileManager = new FileManager(
        javac.getStandardFileManager(. . .), Arrays.asList(classFile), cl);
    CompilationTask task = javac.getTask(null, fileManager, null,
        getClOptions(), null, compilationUnits);
    task.call();
    return cl.loadClass(className);
}
```

Dynamic reloading example summary

- https://github.com/jersey/jersey/tree/master/examples/reload

Agenda

1 Performance improvements

2 Weld se support

3 Client on Android

4 Dynamic reloading example

5 Jersey 3.x

Jersey 3.0

- Jersey 2.x branched off and 3.x on the master
- Based on JAX-RS 2.1
  - Non-blocking IO
  - SSE support
  - Support for reactive programming
- Java 8 friendly
- Backwards compatible with 2.x

Asynchronous Jersey - reminder

- JAX-RS can be partially asynchronous:

```
@GET
public void asyncGet(@Suspended AsyncResponse ar) {
    client.target("someOtherService").request().async()
        .get(new InvocationCallback<String>() {
            public void completed(String result) {
                ar.resume(result);
            }
    . . .
}
```

- So what is left? I/O
  - based on InputStream, OutputStream

Servlet 3.1 non-blocking reminder

```
ServletInputStream inputStream = request.getInputStream();
inputStream.setReadListener(new ReadListener() {
    @Override
    public void onDataAvailable() throws IOException {
        while (inputStream.isReady()) {
            inputStream.read();
        }
    }

    @Override
    public void onAllDataRead() throws IOException {...}

    @Override
    public void onError(Throwable t) {...}
});
```

Non-blocking I/O

- Extra performance boost
- Inspired by but not based on Servlet 3.1
- Beneficial for large and streamed entities
- A brand new client connector
  - Getting rid of HttpUrlConnection
  - First version already in incubator
  - Much better performance than HttpUrlConnection even in blocking mode

Q/A