Service Oriented Web Development with OSGi

Carsten Ziegeler | cziegeler@apache.org

OSGi User Forum Germany 2015

About

cziegeler@apache.org @cziegeler

- RnD Team at Adobe Research Switzerland
- Member of the Apache Software Foundation
  - Apache Felix and Apache Sling (PMC and committer)
  - And other Apache projects
- OSGi Core Platform and Enterprise Expert Groups
- Member of the OSGi Board
- Book / article author, technical reviewer, conference speaker

OSGi Preconceptions

Too Complicated

Dynamics Unmanageable

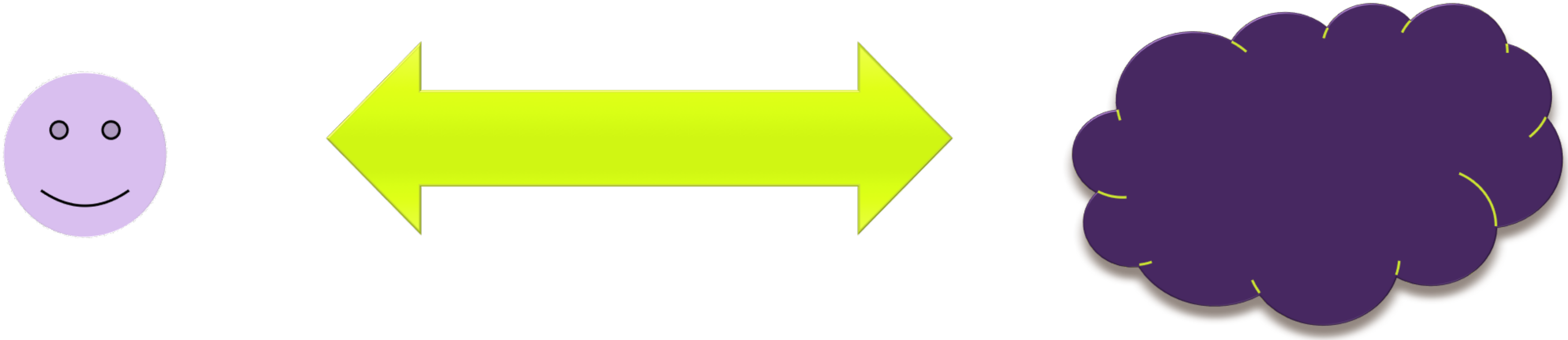
Not suitable for the enterprise

Too slow

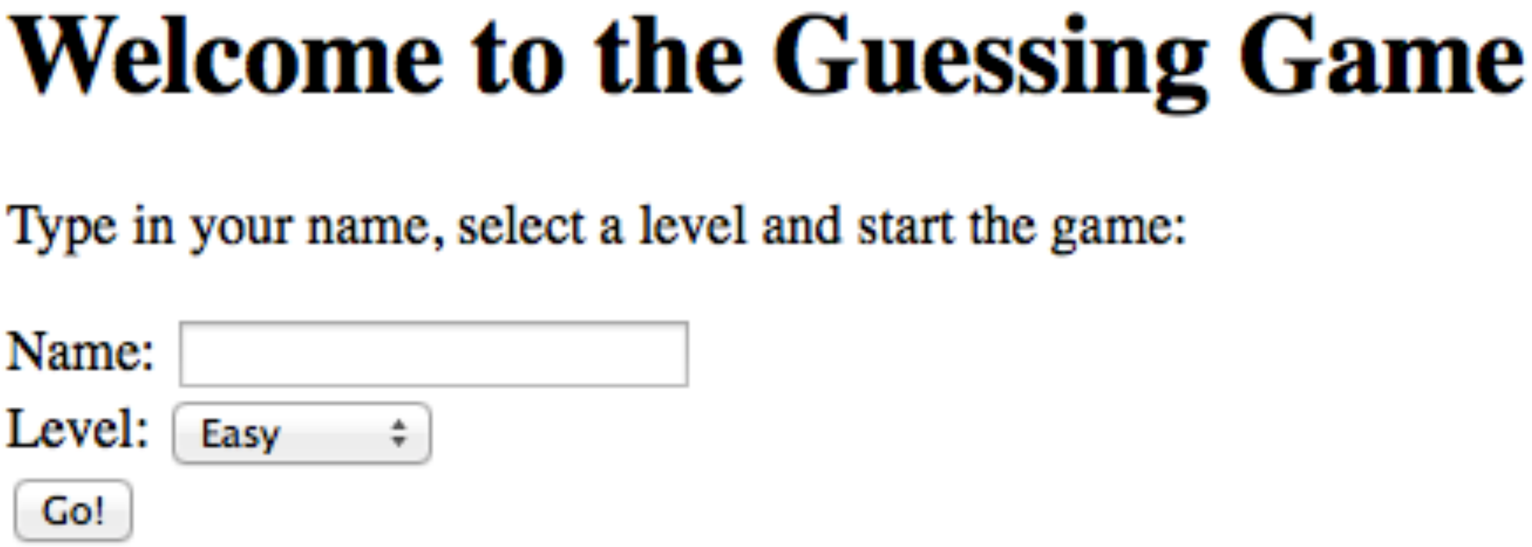
No POJOs

No tooling

No dependency injection



The Next Big Thing



Welcome to the Guessing Game

Type in your name, select a level and start the game:

Name:

Level: Easy

Building Blocks

- Module aka Bundle
- Services
- Components

Game Design

```
public enum Level {

    EASY,
    MEDIUM,
    HARD
}

public interface GameController {

    Game startGame(final String name,
                   final Level level);

    int nextGuess(final Game status,
                  final int guess);

    int getMax(final Level level);
}
```

Implementation

```
@Component
public class GameControllerImpl implements GameController {

    ...
```

Configuration

```
public @interface Config {

    int easy_max() default 10;
    int medium_max() default 50;
    int hard_max() default 100;

}
```

```
private Config configuration;

@Activate
protected void activate(final Config config) {

    this.configuration = config;
}
```

```
  public int getMax(final Level level) {

    int max = 0;

    switch (level) {
      case EASY : max = configuration.easy_max(); break;
      case MEDIUM : max = configuration.medium_max();
break;
      case HARD : max = configuration.hard_max(); break;
    }
     return max;
  }
```

Web?

```
@Component( service = Servlet.class ,

property="osgi.http.whiteboard.servlet.pattern=/game")
public class GameServlet extends HttpServlet {
```

```
public class GameServlet extends HttpServlet {

    @Reference
    private GameController controller;
```

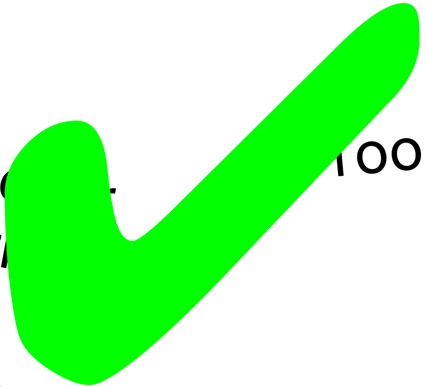
Use Semantic Versioning with the help of baselining.

Too Complicated

Dynamics not manageable

Not suitabl
the enterpr

Too slow



No POJOs

No tooling

No dependency injection

Recipe

- OSGi Declarative Services (Compendium Chapter 112)
  - + RFC 190 Declarative Services Enhancements (OSGi R6)
  - + RFC 212 Field Injection for Declarative Services (OSGi R6)
- OSGi Http Whiteboard Service
  - + RFC 189 (OSGi R6)
- OSGi Configuration Admin (Compendium Chapter 104)
- OSGi Metatype Service (Compendium Chapter 105)
  - + RFC 208 Metatype Annotations

Implementations from Apache Felix

Management

Apache Felix
Web Console

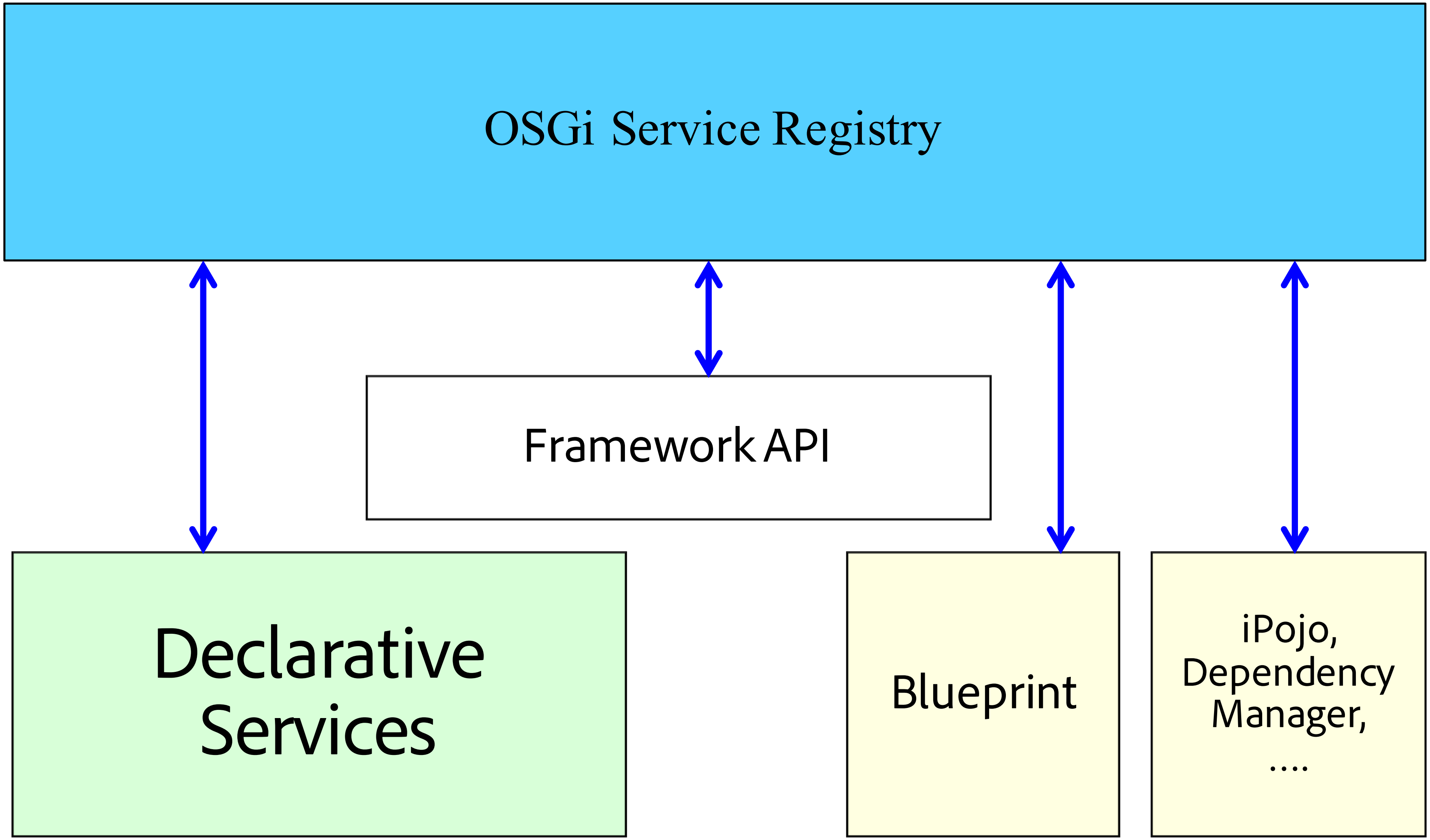
Metatype

```
@ObjectClassDefinition(
    name = "Game Configuration",
    description = "The configuration for the guessing game.")

public @interface Config {

    @AttributeDefinition(name="Easy",
                    description="Maximum value for easy")
    int easy_max() default 10;
```

Metatype

```
@Component
@Designate( ocd = Config.class )

public class GameControllerImpl
             implements GameController {
```

Component Container Interaction



Service Scopes

- Singleton
- Bundle
- Prototype

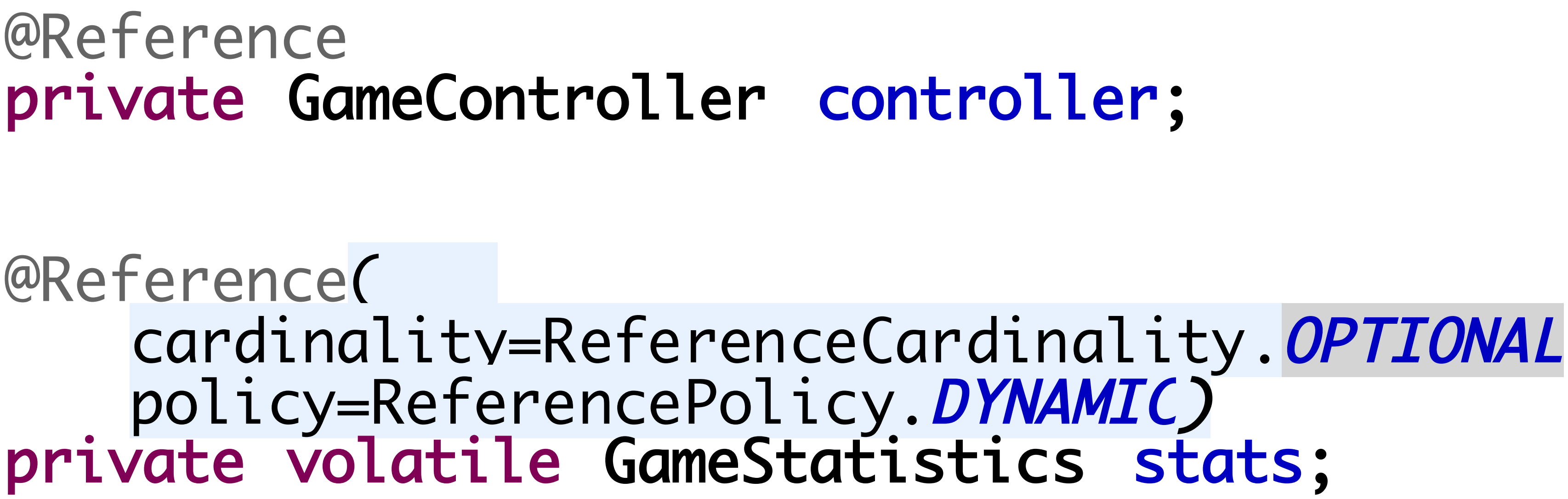
Servlets

```
@Component( service = Servlet.class ,
    scope=ServiceScope.PROTOTYPE,

property="osgi.http.whiteboard.servlet.pattern=/game")

public class GameServlet extends HttpServlet {

      public void init() {...}
      public void destroy() {...}
```

Dynamics

- Lazy instantiation
- Reconfiguration
- Reference policy and cardinality

Unary References



```
@Reference
private GameController controller;


@Reference(
    cardinality=ReferenceCardinality.OPTIONAL
    policy=ReferencePolicy.DYNAMIC)
private volatile GameStatistics stats;
```

Multiple References

```
@Reference(
   cardinality=ReferenceCardinality.MULTIPLE)
private volatile List<Highscore> highscores;
```

Multiple References

```
@Reference

private final Set<Highscore> highscores =
         new ConcurrentSkipListSet<Highscore>();
```

Reconfiguration

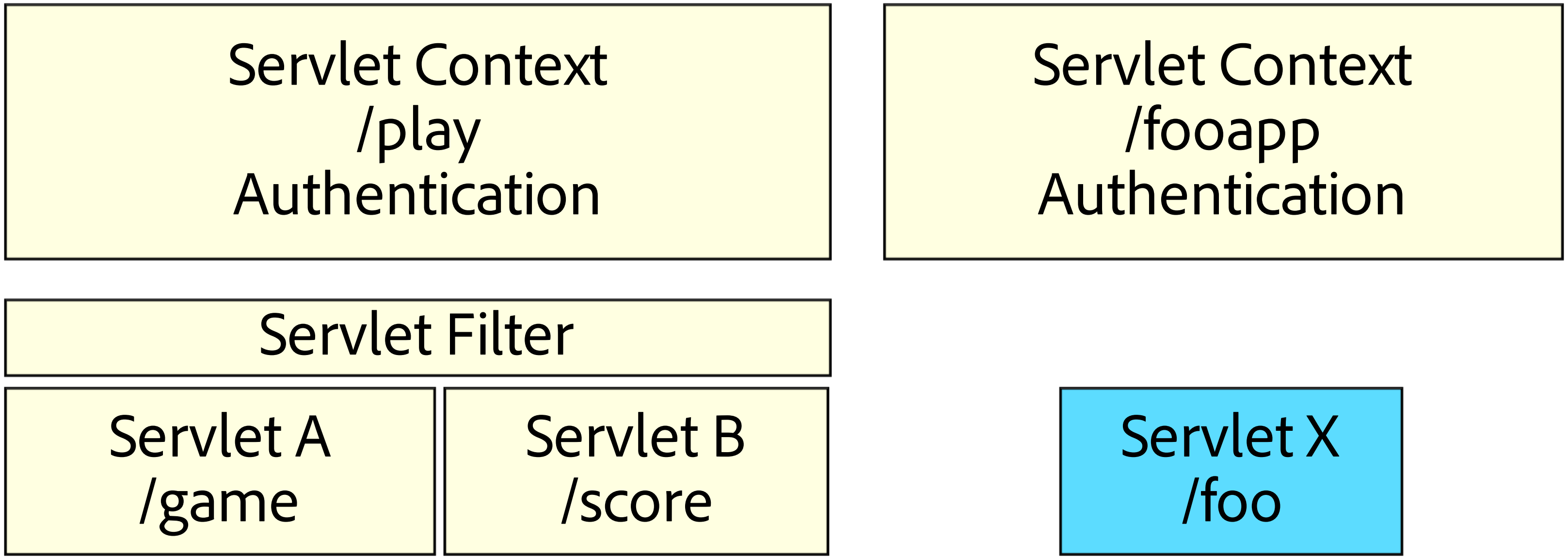
```
private volatile Config configuration;

@Activate
@Modified

protected void activate(final Config config) {

    this.configuration = config;
}
```

Granularity

- Components
- Services
- Bundles

- Servlet/Filter/Listener
- Web app

Web Contexts



Servlet Context
/play
Authentication

Servlet Context
/fooapp
Authentication

Servlet Filter

Servlet A
/game

Servlet B
/score

Servlet X
/foo

Web Contexts

```
@Component( service = Servlet.class ,

  property={"osgi.http.whiteboard.servlet.pattern=/game",
          "osgi.http.whiteboard.context.select=mygame"}

public class ServletA extends HttpServlet {
```

```
@Component( service = Servlet.class ,

  property={"osgi.http.whiteboard.servlet.pattern=/score",
          "osgi.http.whiteboard.context.select=mygame"}

public class ServletB extends HttpServlet {
```

Try it out today!

- HTTP Whiteboard Service
  - Servlet contexts (grouping, authentication)
  - Servlets
  - Filters
  - Listeners

Try it out today!

- Declarative Services
    - Easy too use
    - Pojos
    - DI with handling dynamics

Try it out today!

- Tooling
- Open Source Solutions
- Building large scale enterprise apps

QnA