

OSGi and Spring Data for simple (Web) Application Development

Christian Baranowski

Content of my talk in a sentence

„Java development with Bndtools and bnd is so much fun!“

My Talk in three Words - „*Bndtools is cool!*“

Welcome

- Christian Baranowski (Twitter: @tux2323)
- Software Developer @ SEITENBAU
 - Software Engineering
 - Custom Software Solutions
 - E-Government Solutions
 - Identity Management and SSO Solutions
- www.seitenbau.de



Bndtools

Easy, powerful and productive way to develop OSGi applications. Based on bnd and Eclipse.

<http://bndtools.org/>

„Development should be fun, so you need the right tools!“

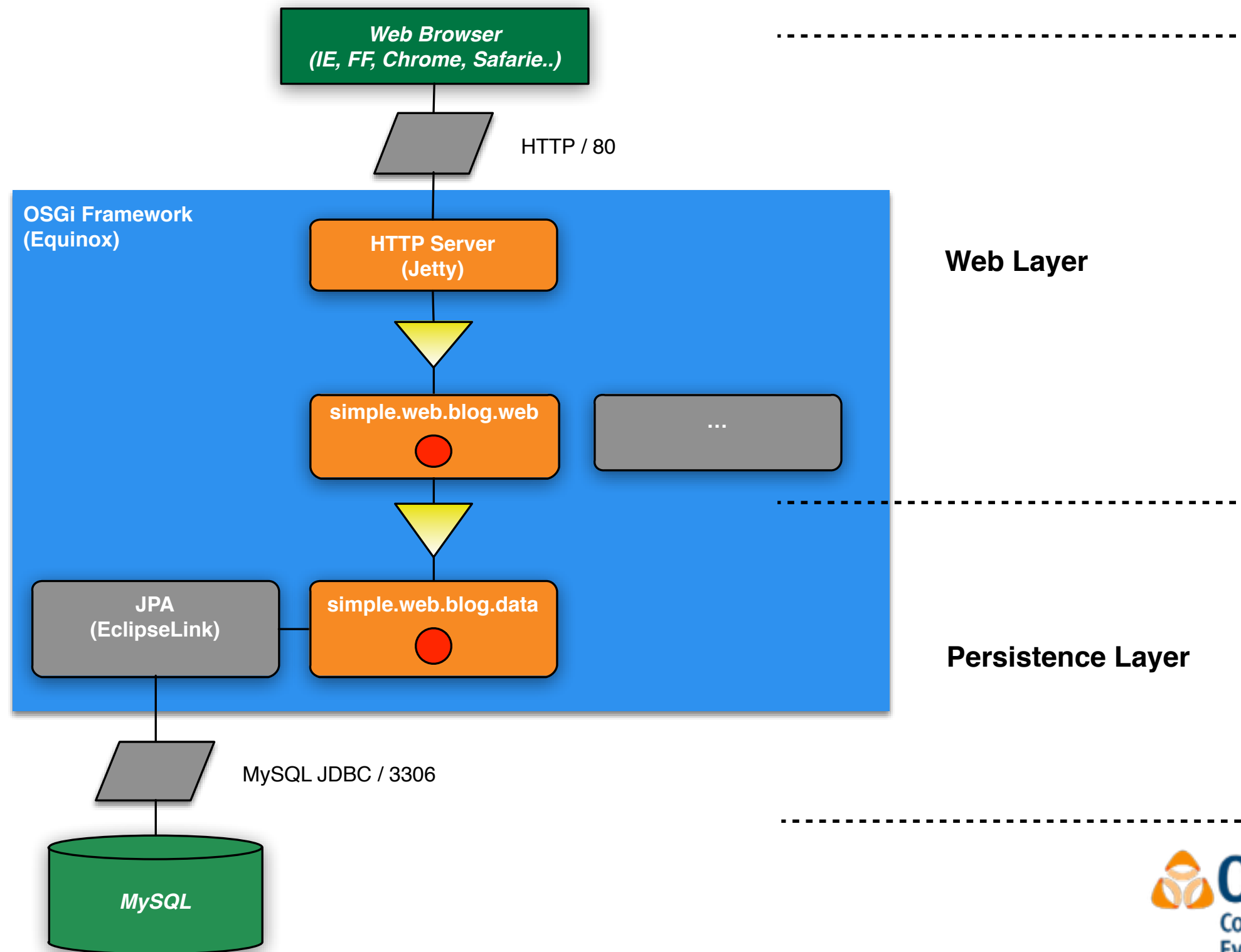


enRoute

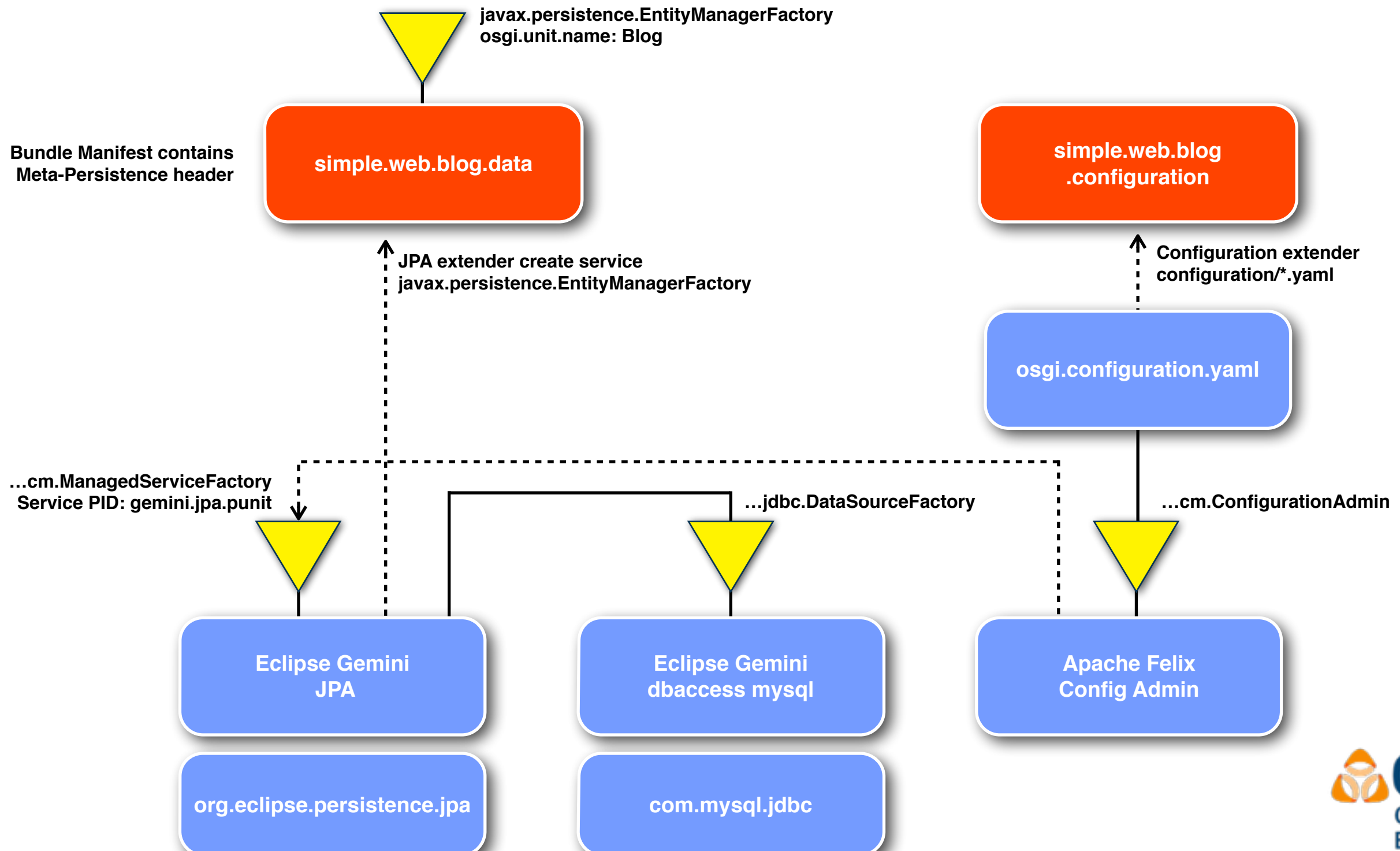


- Getting started with OSGi → enRoute project
<http://enroute.osgi.org/>
- The talk is based on the ideas from the enRoute blog demo project
- enRoute OSGi blog sample project by Peter Kriens
<https://github.com/osgi/osgi.enroute.blog/>
- Step by step tutorial from Peter Kriens
<http://goo.gl/Y569g5>
- Last OSGi Code Camp (Ludwigsburg 2013) was based on this step by step tutorial

Running Blog Example

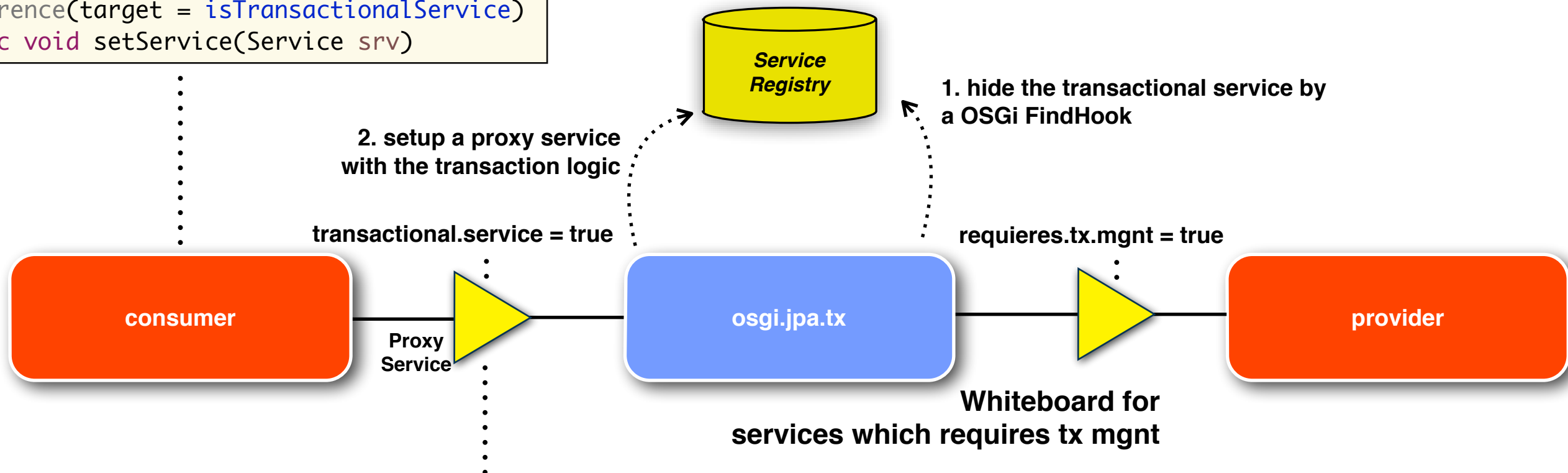


Persistence Layer



Simple Transaction Management

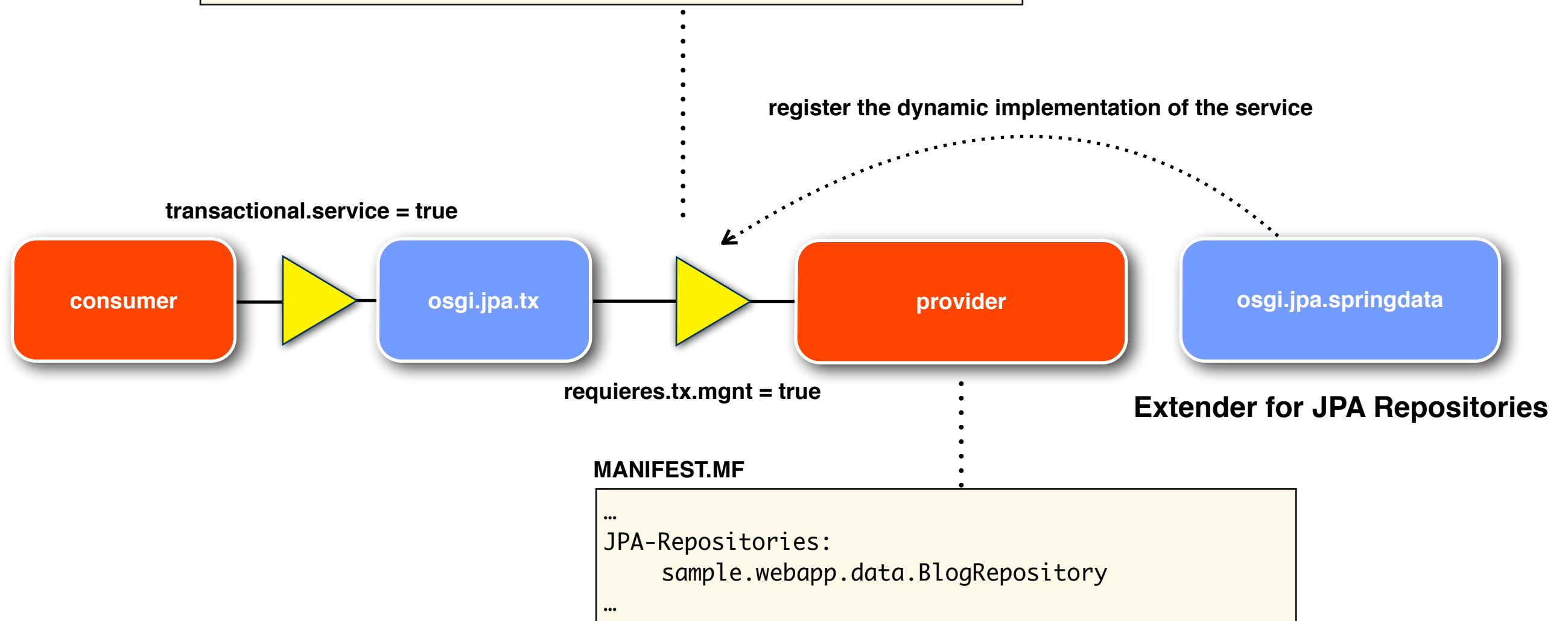
```
@Reference(target = isTransactionalService)
public void setService(Service srv)
```



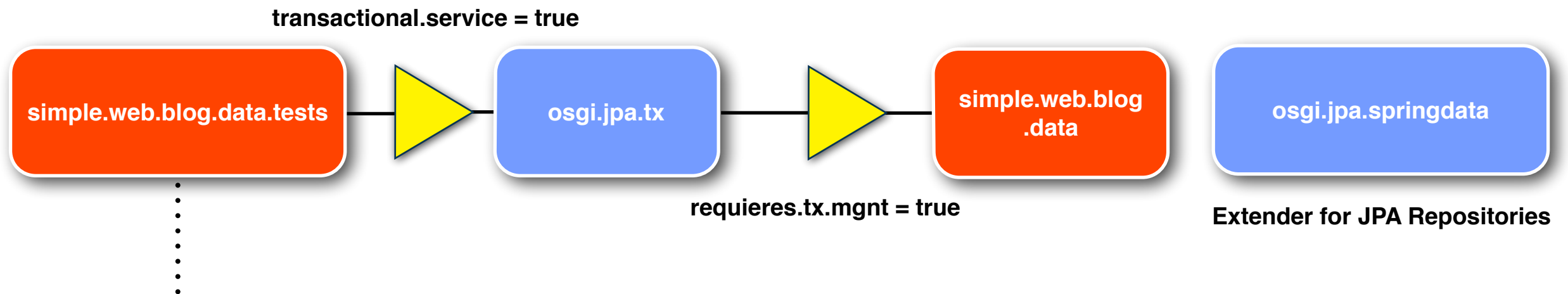
```
if (tx.isTransactionOpen()) {
    return method.invoke(bundleContext.getService(serviceReference), args);
}
try {
    tx.begin();
    Object result = method.invoke(bundleContext.getService(serviceReference), args);
    tx.commit();
    return result;
} catch (Exception exp) {
    tx.rollback();
    throw exp;
} finally {
    bundleContext.ungetService(txMgrServiceReference);
}
```

Spring Data Extender

```
public interface BlogRepository extends JpaRepository<Blog, Long> {  
    List<Blog> findByTitleContaining(String part);  
}
```



Spock based OSGi Integration Tests



```
class BlogRepositorySpec extends Specification {

    @OSGiService
    BlogRepository blogRepository

    def setup() {
        blogRepository.deleteAll()
        blogRepository.save(new Blog(title: 'OSGi Web Dev'))
        blogRepository.save(new Blog(title: 'OSGi V.S Java EE'))
    }

    def findBlogPostByTitleContainingOSGi() {
        when:
        def list = blogRepository.findByTitleContaining("OSGi")
        then:
        list.size() == 2
    }
}
```

Web Layer



<https://github.com/alrra/browser-logos>

Jersey MVC

Handlebars View (list.hbs):

```
<html>
<head>
{{#resource type="css"}} /css/app.css {{/resource}}
</head>
<body>
<table class="table table-striped">
  <thead>
    <th>#id</th><th>Title</th><th>Content</th><th></th>
  </thead>
  ... {{#html-table-content columns="id, title, content" resource="blog"}}
    {{/html-table-content}}
  </table>
  {{#html-pagination}} {{/html-pagination}}
</body>
</html>
```

Handlebars
Helpers

com.github.jknack.
handlebars

Controller (BlogController):

```
@GET
@Produces( MediaType.TEXT_HTML )
@Template(name="list.hbs")
public Page<Blog> list(
    @QueryParam("page")      @DefaultValue("0")      Integer page,
    @QueryParam("size")      @DefaultValue("10")      Integer size) {
    ... return blogRepository.findAll(new PageRequest(page, size));
}
```

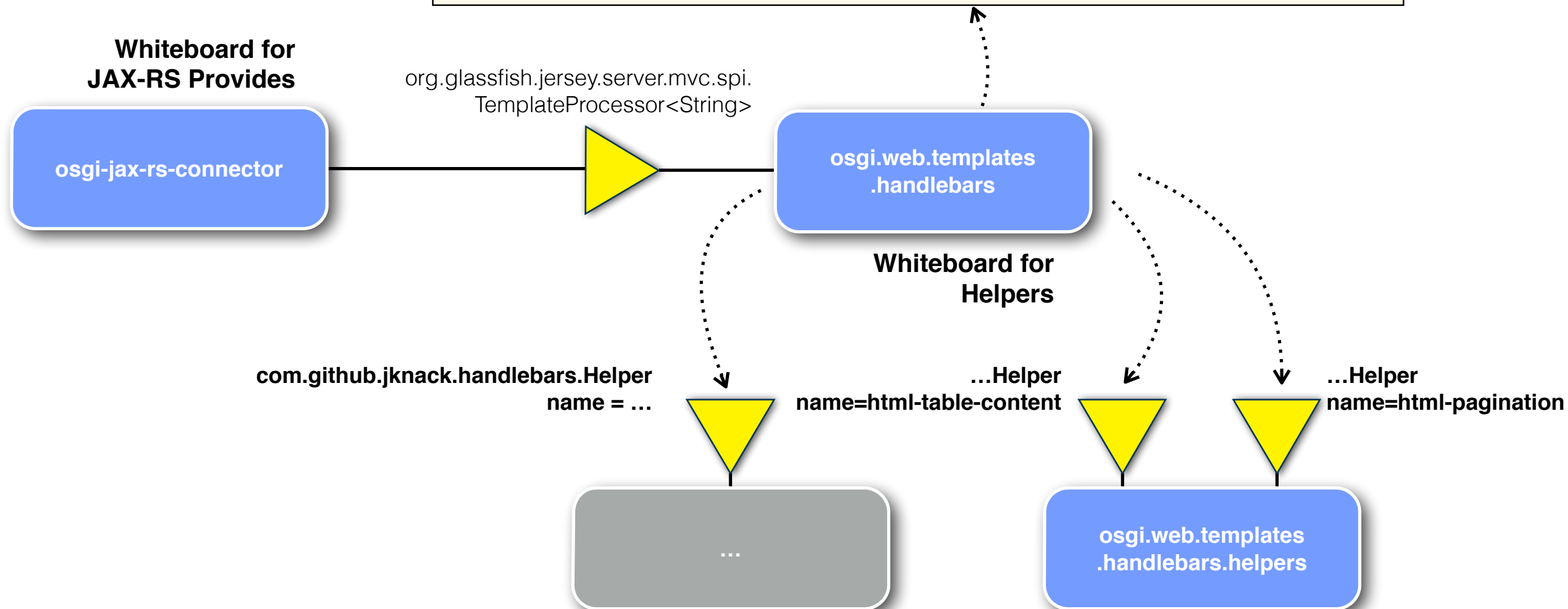
return the model

Controller Method

osgi-jax-rs-connector

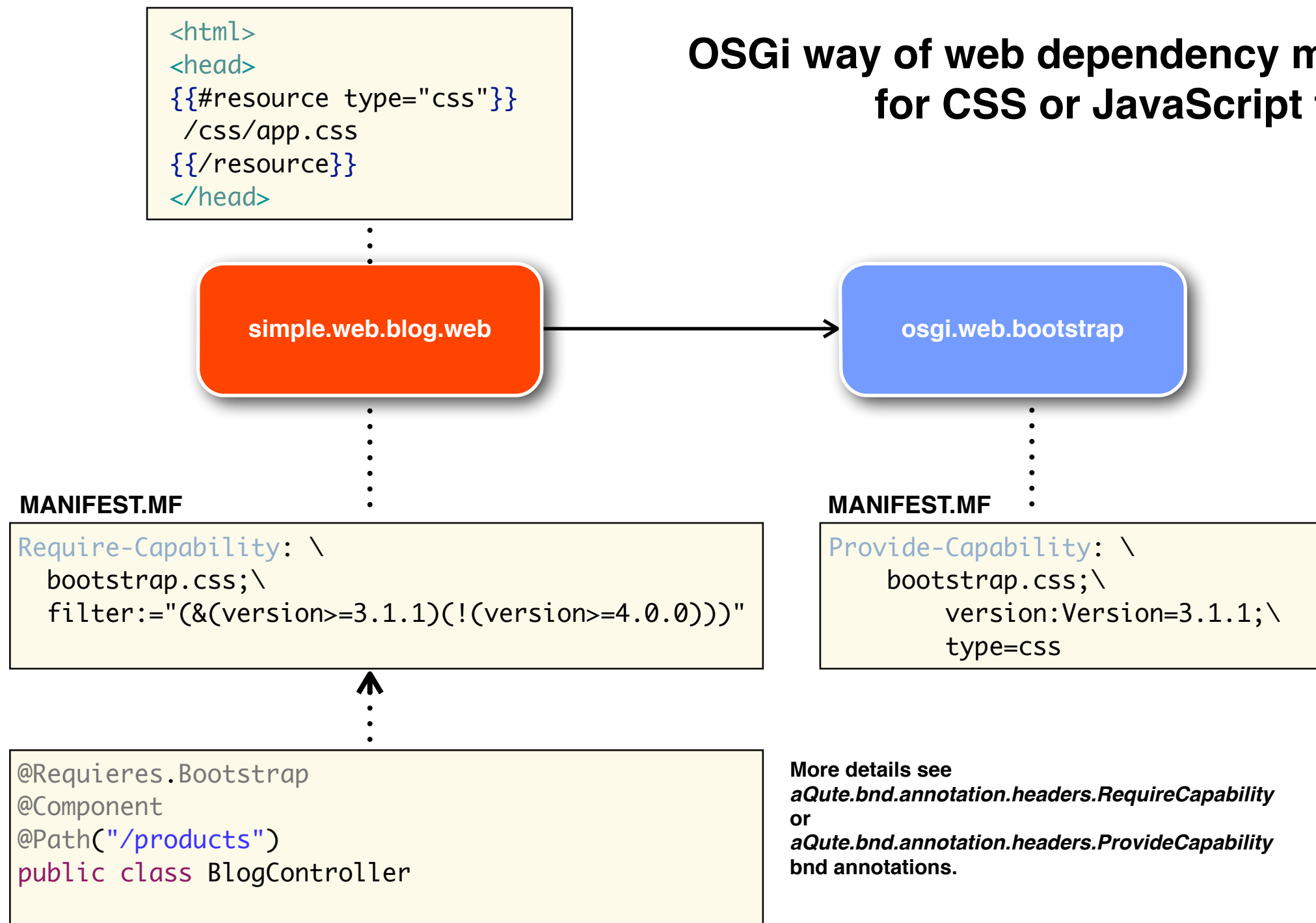
Handlebars Helpers

```
<thead>
  <th>#id</th><th>Title</th><th>Content</th><th></th>
</thead>
{{#html-table-content columns="id, title, content" resource="blog"}}
{{/html-table-content}}
```



Static Web Bundles

OSGi way of web dependency management
for CSS or JavaScript frameworks



Web Bundle build with Yeoman Grunt Bower



yeoman

- Scaffolding tool for webapps
- Yeoman helps kickstart new projects
- provide a generator ecosystem

<http://yeoman.io/>



Grunt

- JavaScript Task Runner
- Grunt ecosystem is huge
- minification, compilation, unit testing, linting, ...

<http://gruntjs.com/>



bower

- package manager for the web
- solution to the problem of front-end package management
- ecosystem is huge

<http://bower.io/>

Thats the tool chain web developers love...

REST Consumer based on AngularJS

\$blogResource (REST Consumer)

```
angular.module('blogApp')
  .factory('$blogResource', ['$resource', function($resource) {
    return $resource( '/rest/blog/:postId', { postId: '@postId' }, { });
  }]);
```

MainCtrl (the controller is using the REST resource to delete a blog entry)

```
angular.module('blogApp')
  .controller('MainCtrl', ['$scope', '$blogResource', function($scope, $blogResource) {

    $scope.posts = $blogResource.query();

    $scope.deletePost = function(post) {
      $blogResource.delete({postId: post.id}).$promise.then(function() {
        $scope.posts = $blogResource.query();
      });
    };

  }]);
```

Building REST Resource (JAX-RS)

A REST Resource build in standard and flexible way based on JAX-RS

```
@Component
@Path("/rest/blog")
public class BlogResource implements Resource {

    BlogRepository blogRepository;

    @Reference(target = isTransactionalService)
    public void setBlogRepository(BlogRepository blogRepository) { ... }

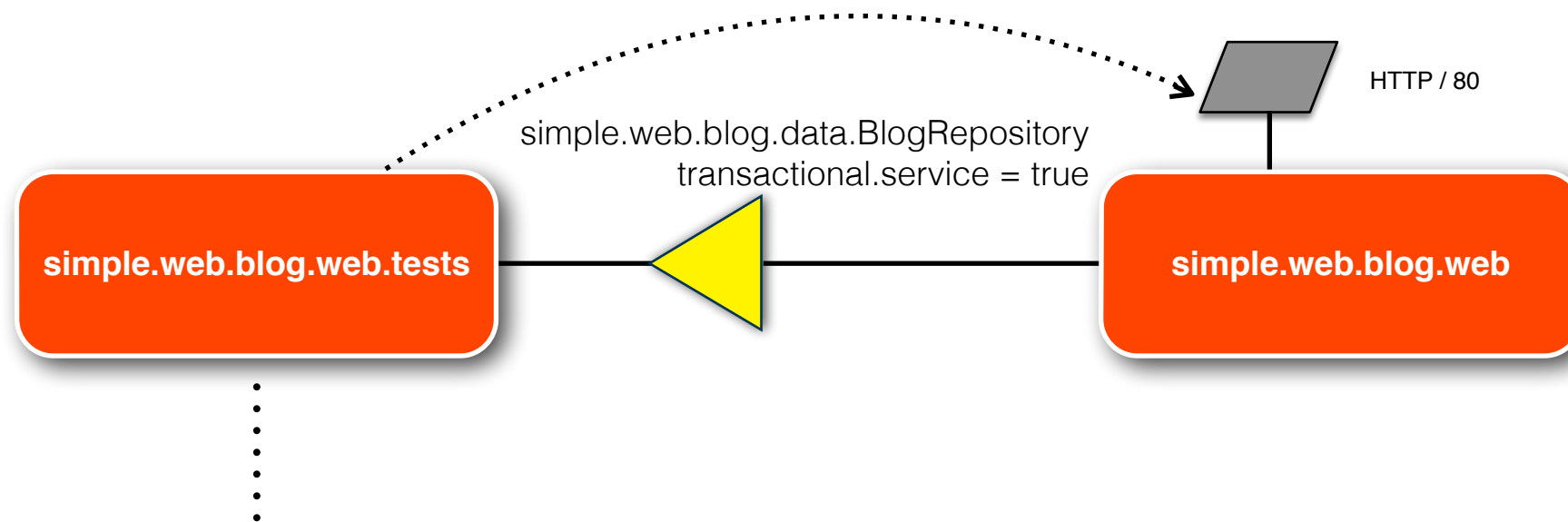
    @GET
    public List<Blog> query() { return blogRepository.findAll(); }

    @GET
    @Path("/{id}")
    public Blog get(@PathParam("id") Long id) { return blogRepository.getOne(id); }

    @POST
    public void post(Blog blog) { blogRepository.save(blog); }

    @DELETE
    @Path("/{id}")
    public void delete(@PathParam("id") Long id) { blogRepository.delete(id); }
}
```


Integration Testing REST Resources



```
class BlogResourceSpec extends Specification {  
  
    @OSGiServiceRegistration(properties=["transactional.service = true"])  
    BlogRepository mockBlogRepository = Mock(BlogRepository)  
  
    def getProductsByExistingId() {  
        given:  
        mockBlogRepository.findOne(42) >> new Blog(title: 'OSGi in Action', content: '-')  
        when:  
        Client client = ClientBuilder.newClient();  
        Response response = client  
            .target("http://localhost:8080")  
            .path("halres").path("blog").path("42").request().get();  
        then:  
        response.status == Status.OK.statusCode  
    }  
}
```

Testing AngularJS Controllers

```
// Initialize the controller and a mock scope
beforeEach(inject(function ($controller, $rootScope, $injector) {
    scope = $rootScope.$new();
    $httpBackend = $injector.get('$httpBackend');
    $httpBackend.expect('GET', '/rest/blog').respond([{id: 1}, {id: 42}]);
    MainCtrl = $controller('MainCtrl', {
        $scope: scope
    });
}));

it('should send a request to delete a blog post', function () {
    $httpBackend.expect('DELETE', '/rest/blog/42').respond(200, 'success');
    $httpBackend.expect('GET', '/rest/blog').respond([{id: 1}]);
    scope.deletePost({id: 42});
    $httpBackend.flush();
    expect(scope.posts.toString()).toBe([{id: 1}].toString());
});
```

Technologie Stack

- **Modern Web-Application OSGi Stack**
 - AngularJS (Superheroic JavaScript Framework)
<https://angularjs.org/>
 - Jetty (Web Server)
<https://www.eclipse.org/jetty/>
 - osgi-jax-rs-connector (Jersey)
<https://github.com/hstaudacher/osgi-jax-rs-connector>
 - Spring Data JPA (for simple JPA Services)
<http://projects.spring.io/spring-data-jpa/>
 - Spock (testing and specification framework)
<https://code.google.com/p/spock/>
 - Eclipse Equinox or Apache Felix as powerful OSGi Framework



Feedback

„**Erik Meijer:**

*Are you saying you cannot write large programs in **Java**?*

Anders Hejlsberg:

*No, you can write large programs in **Java**.*

You just can't maintain them. „

Quelle - <http://t.co/Uw2iglqf>

Compose small “**applications**” (modules)
in **to large systems.**

Quelle - <http://t.co/Uw2iglqf>

Resources

- OSGi Simple Blog App
<https://github.com/tux2323/simple.web.blog>