Introduction to JAX-RS

Developing RESTful Web services with JAX-RS

What is JAX-RS?

- JAX-RS: Java API for RESTful Services
  - An annotation driven API to help developers build RESTful Web services and clients in Java.
    - resources are described by POJOs + annotations
    - uses an http centric programming model
    - is included in Java EE

What are RESTful Services?

- The way a RESTful application works:
    - resources are identified by URIs
        - (JAX-RS maps these resources to Java classes)
    - clients read/write/modify/delete resources only via standard http requests (GET, PUT, POST, DELETE)
    - requests and responses contain resource representations in formats identified by media types (MIME types)
    - responses contain URIs that link to further resources

Pricipals of RESTful Web Services

- give every resource (information unit) an ID
- use only the standard set of http methods
- use http links (URIs) to define relations between resources
- you may use multiple representations (e.g. XML, JSON, text)
- use stateless communications
    - the server does not keep session information

Resources are always identified by URIs

- JAX-RS maps a resource to a Java class
  - a POJO (Plain Old Java Object)
- the resource ID is provided by the @PATH annotation
  - the value is a relative URI; the base URI is provides by the deployment context (deployment descriptor) or parent resource
  - use embeded parameters for a variable part of the URI

Example of Resource URIs

- retrieving information about the customer of a purchase order with known ID:

variable part of URI

```
@Path("orders/{order_id}")
public class OrderResource {
  @GET
  @Path("customer")
  CustomerResource getCustomer(...) {...}
}
```

The Standard Set of Methods

- in contrast to SOAP-based Web services REST uses a standard set of methods
  - JAX-RS routes the request to the appropriate resource class and method
  - the method's return value is mapped to the response

Method	Purpose	Annotation
GET	read (possible to cache it)	@GET
POST	update	@POST
PUT	create	@PUT
DELETE	remove	@DELETE

Mapping of Parameters

- parameter annotations specify the mapping of request parameters to Java parameters

```
@Path("properties/{name}")
public class SystemProperty {
  @GET
  Property get(@PathParam("name") String name)
    {...}

  @PUT
  Property set(@PathParam("name") String name,
    String value) {...}
}
```

MIME type determines Resource Representation

- the data may be offered in a variety of formats
  - XML, JSON, XHTML, text…
    - for different kinds of clients
- content negotiation is supported
  - by specifying it in the accept header:
    - e.g.: GET /myResource
      Accept: application/json
  - or URI based:
    - e.g.: GET /myResource.json

Response to Content Negotiation With Accept Header

- specify resonse to different accept headers:

```
@GET
@Produces({"application/xml","application/json"})
Order getOrder(@PathParam("order_id") String id){
  ...
}

@GET
@Produces("text/plain")
String getOrder2(@PathParam("order_id") String id){
  ...
}
```

Response to URL-based Content Negotiation

- specify resonse to different URL endings:

```
@Path("/orders")
public class OrderResource {
  @Path("{orderId}.xml")
  @GET
  public Order getOrderInXML(@PathParam("orderId")
                                   String orderId) {
    . . .
  }

  @Path("{orderId}.json")
  @GET
  public Order getOrderInJSON(@PathParam("orderId")
                                   String orderId) {
    . . .
  }
}
```

Use http Links to Define Relations Between Resources

- example where the response contains links
- link customer and product information to purchase order:

```
<order self="http://example.com/orders/101230">
  <customer ref="http://example.com/customers/bar">
  <product ref="http://example.com/products/21034"/>
  <amount value="1"/>
</order>
```

JAX-RS Implementations

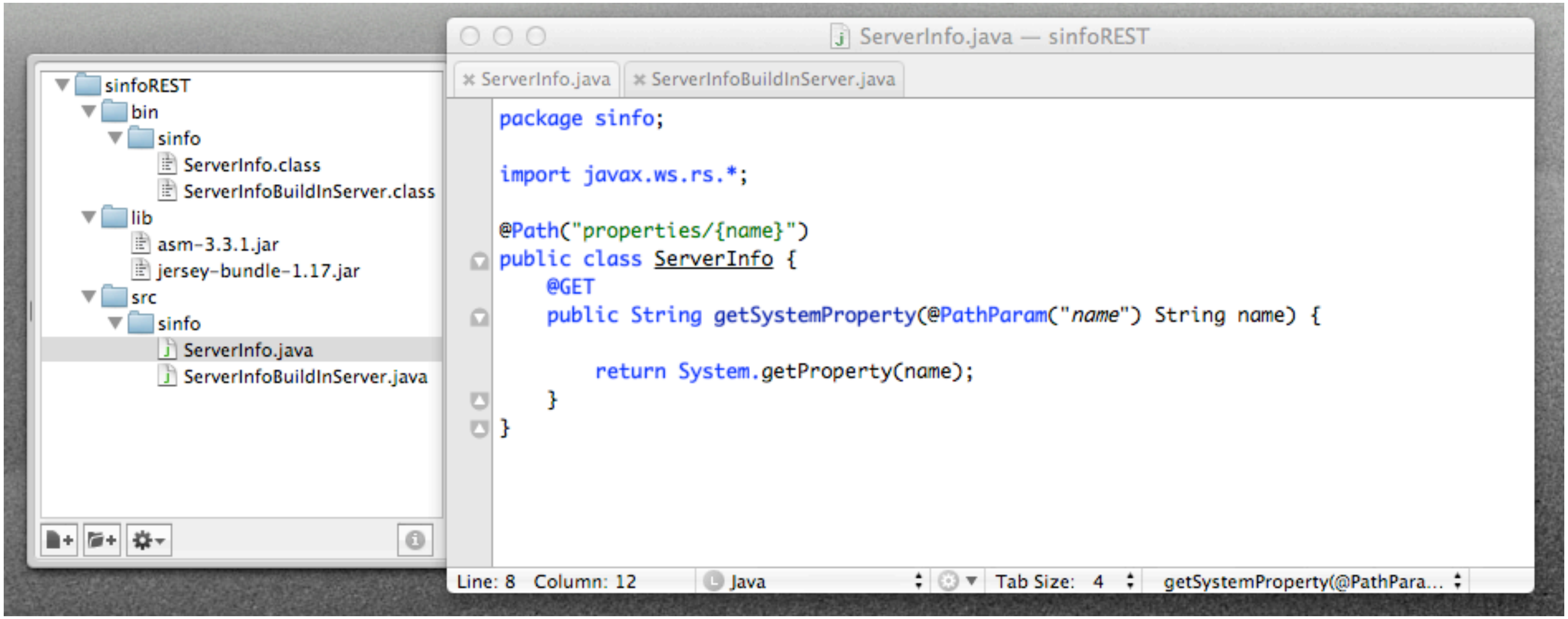
- every Java EE 6 application server implements it
- Java SE 6 and 7 do not implement it
- **Jersey** is the open source, production quality, JAX-RS (JSR 311) Reference Implementation
  - http://jersey.java.net
    - may be used with a servlet container (e.g. Tomcat) or with the mini-http-server build into Java SE 6 and 7

Jersey

- download the latest versions of the following files and put them in your classpath:
  - jersey-bundle.jar
  - jsr311-api.jar
  - asm.jar

Almost the simplest example possible:

- defining a RESTful service to return Java system properties



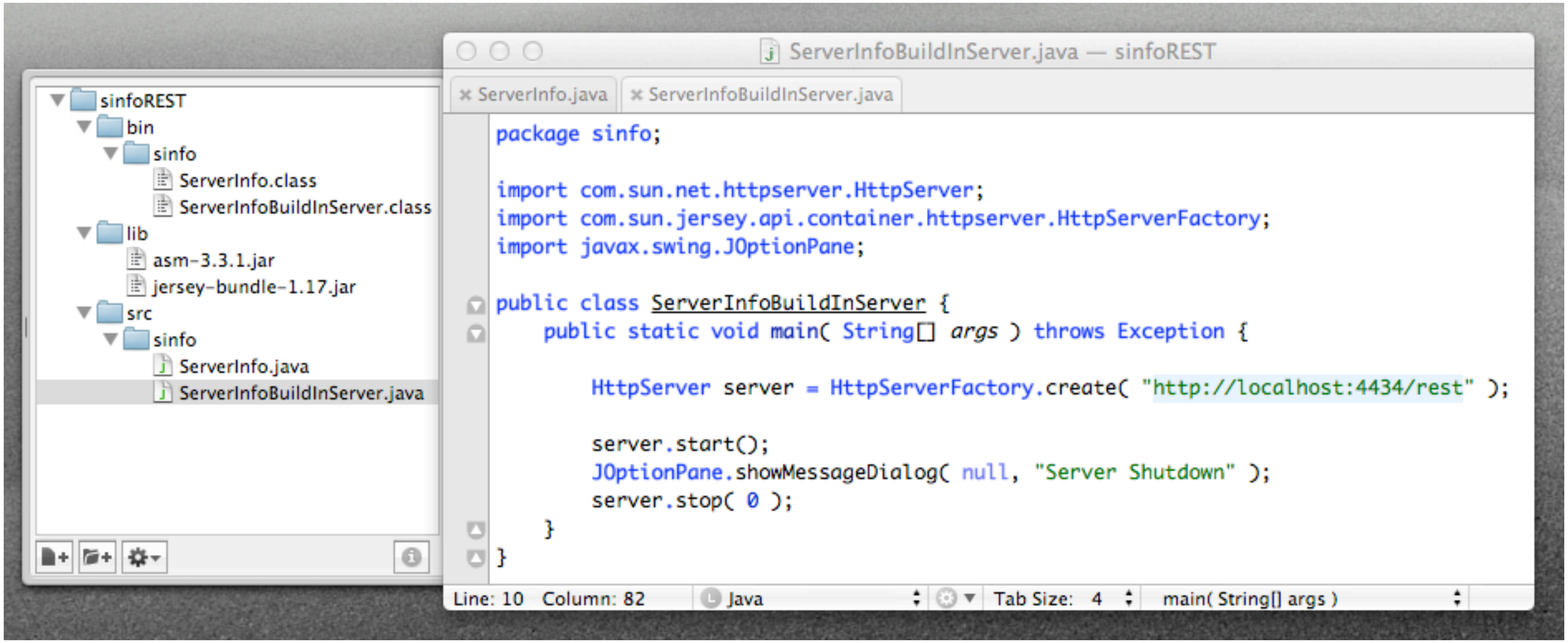
sinfoREST
bin
sinfo
ServerInfo.class
ServerInfoBuildInServer.class
lib
asm-3.3.1.jar
jersey-bundle-1.17.jar
src
sinfo
ServerInfo.java
ServerInfoBuildInServer.java
ServerInfo.java — sinfoREST
ServerInfo.java
ServerInfoBuildInServer.java

```java
package sinfo;

import javax.ws.rs.*;

@Path("properties/{name}")
public class ServerInfo {
    @GET
    public String getSystemProperty(@PathParam("name") String name) {

        return System.getProperty(name);
    }
}
```

Line: 8 Column: 12 Java Tab Size: 4 getSystemProperty(@PathPara...

Deployment with the build-in Server

- Jersey integrates itself into the Java build-in http-server
  - when starting, it will scan the classpath for JAX-RS annotated classes…



```java
package sinfo;

import com.sun.net.httpserver.HttpServer;
import com.sun.jersey.api.container.httpserver.HttpServerFactory;
import javax.swing.JOptionPane;

public class ServerInfoBuildInServer {
    public static void main( String[] args ) throws Exception {

        HttpServer server = HttpServerFactory.create( "http://localhost:4434/rest" );

        server.start();
        JOptionPane.showMessageDialog( null, "Server Shutdown" );
        server.stop( 0 );
    }
}
```

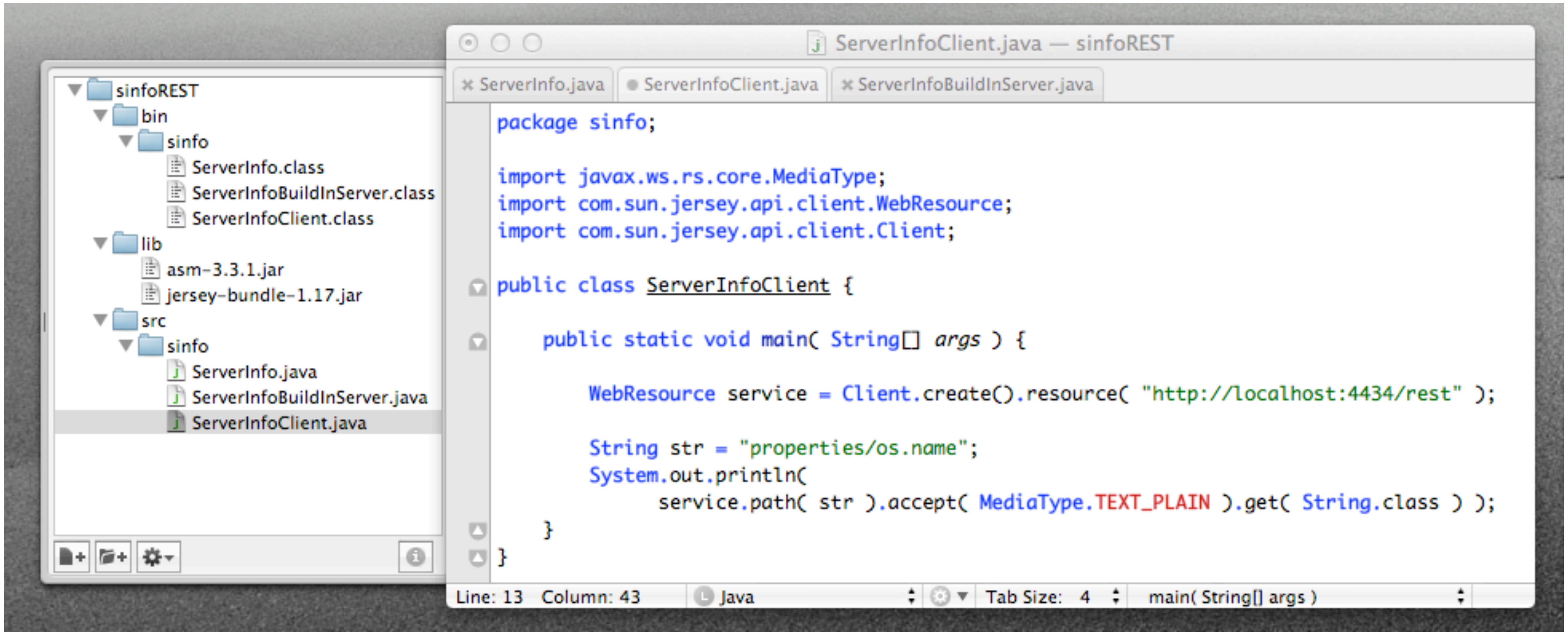
To Get It Going…

- `javac -cp bin:lib/* -d bin src/sinfo/*.java`
- `java -cp bin:lib/* sinfo/ServerInfoBuildInServer`

```
Apr 17, 2013 6:15:55 PM com.sun.jersey.api.core.ClasspathResourceConfig init
INFO: Scanning for root resource and provider classes in the paths:
  bin
  lib/asm-3.3.1.jar
  lib/jersey-bundle-1.17.jar
Apr 17, 2013 6:15:55 PM com.sun.jersey.api.core.ScanningResourceConfig logClasses
INFO: Root resource classes found:
  class sinfo.ServerInfo
Apr 17, 2013 6:15:55 PM com.sun.jersey.api.core.ScanningResourceConfig init
INFO: No provider classes found.
Apr 17, 2013 6:15:55 PM com.sun.jersey.server.impl.application.WebApplicationImp
```

A Simple Client

- the client may use Jersey as well to simplify the coding:



```java
package sinfo;

import javax.ws.rs.core.MediaType;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.Client;

public class ServerInfoClient {

    public static void main( String[] args ) {

        WebResource service = Client.create().resource( "http://localhost:4434/rest" );

        String str = "properties/os.name";
        System.out.println(
                service.path( str ).accept( MediaType.TEXT_PLAIN ).get( String.class ) );
    }
}
```

Line: 13   Column: 43   Java   Tab Size: 4   main(String[] args)

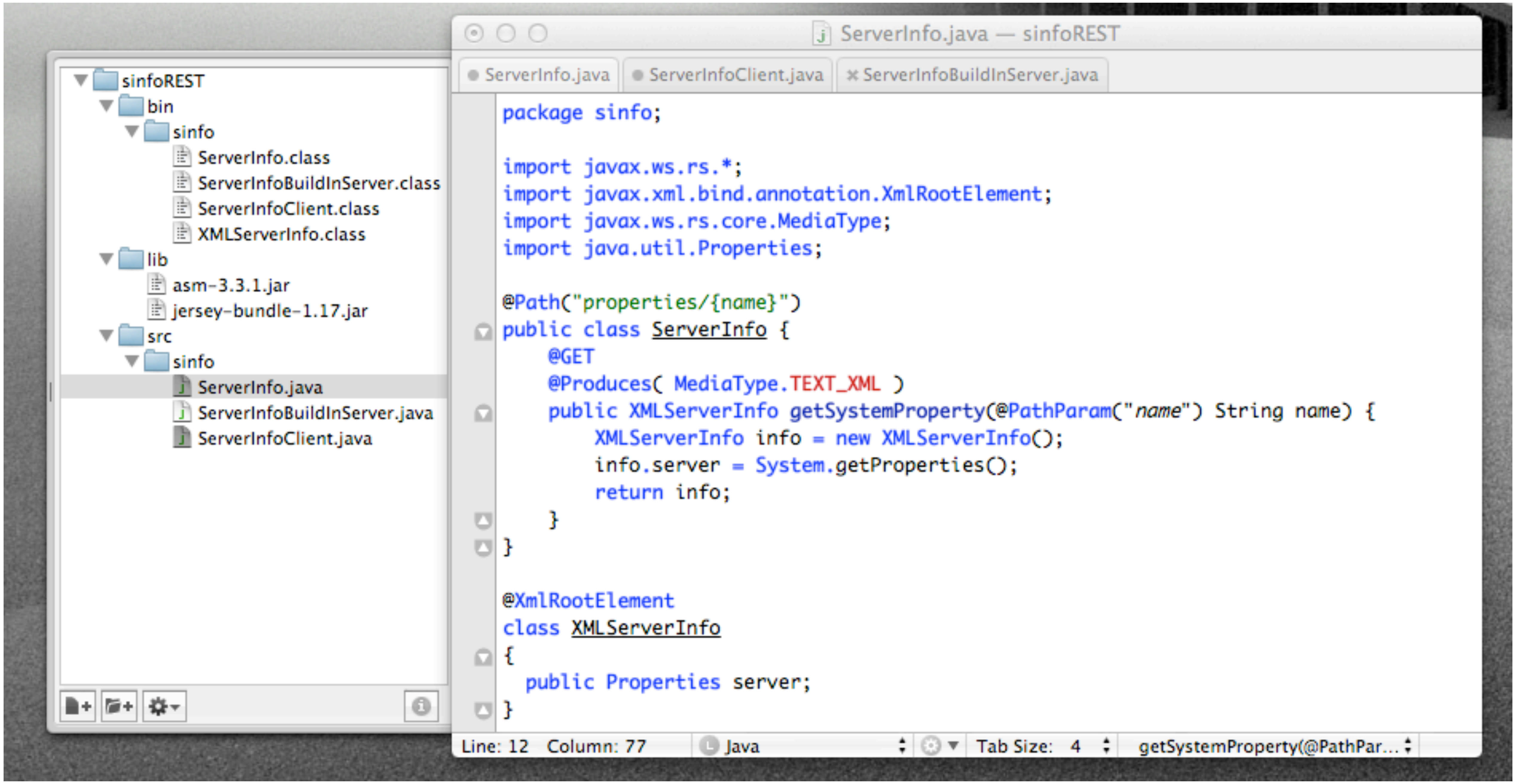
To Get It Going…

- `javac -cp bin:lib/* -d bin src/sinfo/*.java`
- `java -cp bin:lib/* sinfo/ServerInfoClient`
  `Mac OS X`

Returning a Java Object as XML

- the Java API for XML Binding JAXB may be used to let our service return all Java system properties as XML
  - JAXB converts a Java Object to XML
- we use the **@XMLRootElement** annotation to translate a **java.util.Properties** object into XML

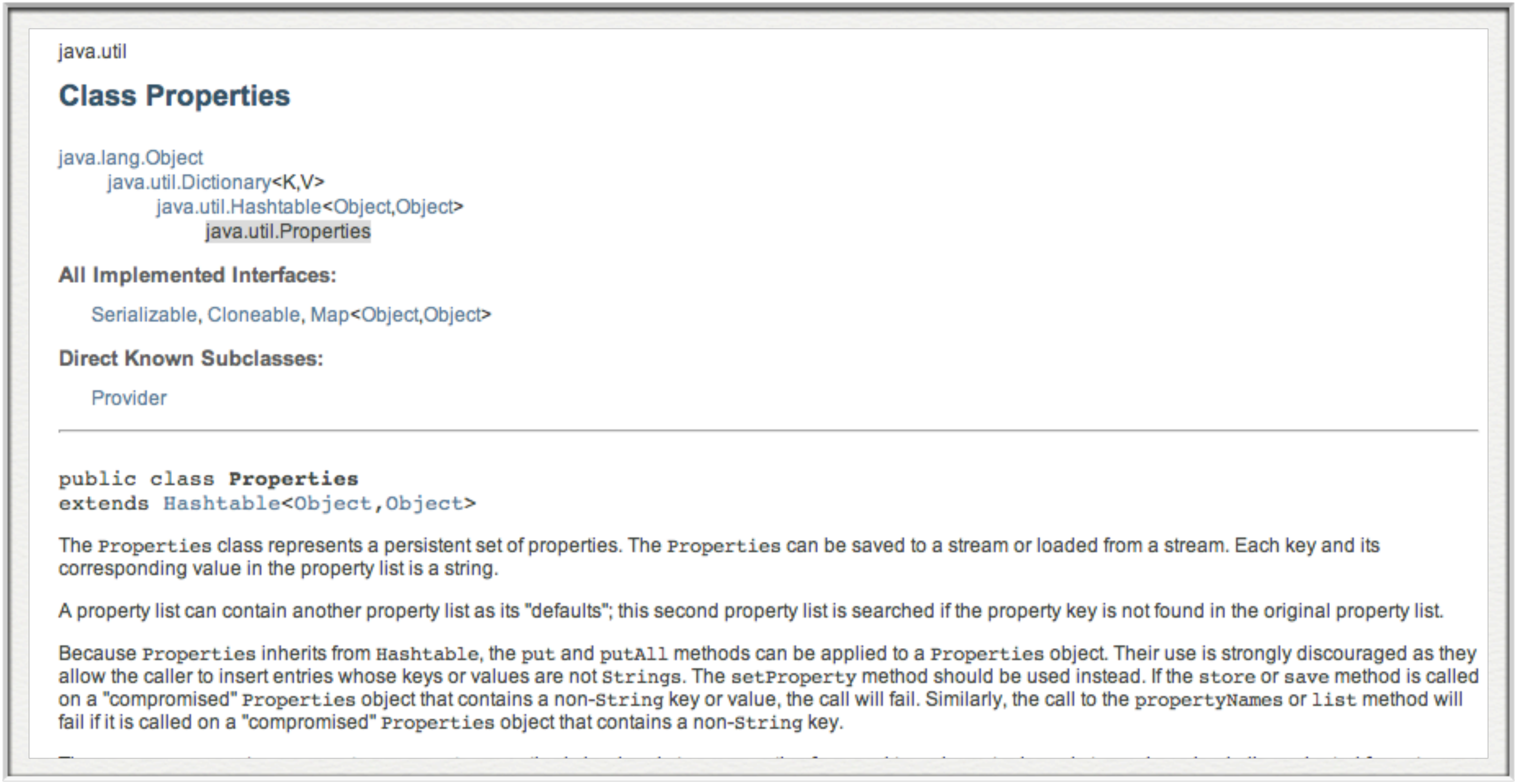
Modified ServerInfo.java



```java
package sinfo;

import javax.ws.rs.*;
import javax.xml.bind.annotation.XmlRootElement;
import javax.ws.rs.core.MediaType;
import java.util.Properties;

@Path("properties/{name}")
public class ServerInfo {
    @GET
    @Produces( MediaType.TEXT_XML )
    public XMLServerInfo getSystemProperty(@PathParam("name") String name) {
        XMLServerInfo info = new XMLServerInfo();
        info.server = System.getProperties();
        return info;
    }
}

@XmlRootElement
class XMLServerInfo
{
   public Properties server;
}
```

Line: 12   Column: 77   Java   Tab Size: 4   getSystemProperty(@PathPar…

The Object Containing the System Properties



java.util

Class Properties

java.lang.Object
    java.util.Dictionary<K,V>
        java.util.Hashtable<Object,Object>
            java.util.Properties

All Implemented Interfaces:

Serializable, Cloneable, Map<Object,Object>

Direct Known Subclasses:

Provider

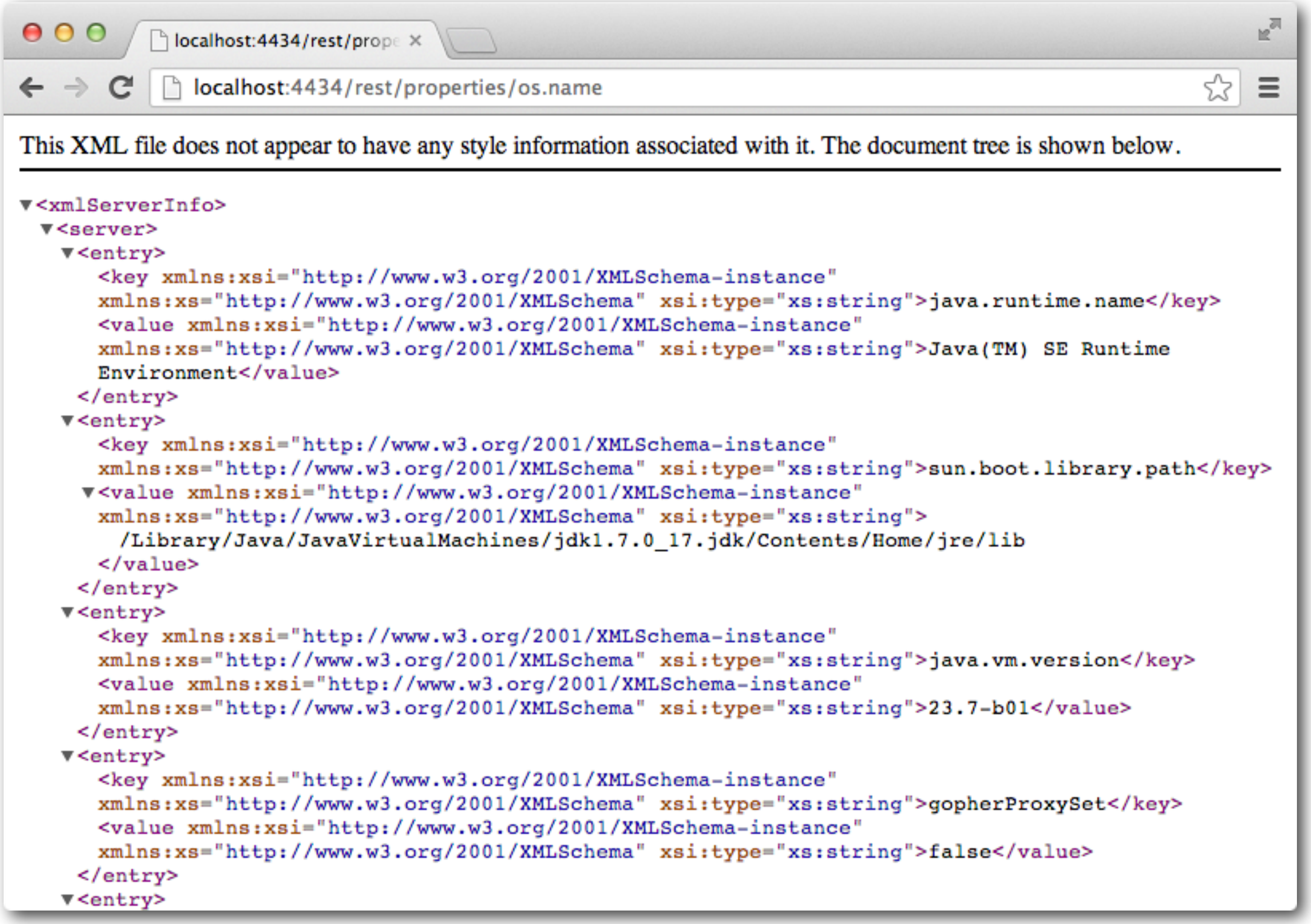
```
public class Properties
extends Hashtable<Object,Object>
```

The `Properties` class represents a persistent set of properties. The `Properties` can be saved to a stream or loaded from a stream. Each key and its corresponding value in the property list is a string.

A property list can contain another property list as its "defaults"; this second property list is searched if the property key is not found in the original property list.

Because `Properties` inherits from `Hashtable`, the `put` and `putAll` methods can be applied to a `Properties` object. Their use is strongly discouraged as they allow the caller to insert entries whose keys or values are not `Strings`. The `setProperty` method should be used instead. If the `store` or `save` method is called on a "compromised" `Properties` object that contains a non-`String` key or value, the call will fail. Similarly, the call to the `propertyNames` or `list` method will fail if it is called on a "compromised" `Properties` object that contains a non-`String` key.

Testing the Modified Service With a Browser


localhost:4434/rest/prope ×
localhost:4434/rest/properties/os.name

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<xmlServerInfo>
  ▼<server>
    ▼<entry>
        <key xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">java.runtime.name</key>
        <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">Java(TM) SE Runtime
        Environment</value>
      </entry>
    ▼<entry>
        <key xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">sun.boot.library.path</key>
      ▼<value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">
          /Library/Java/JavaVirtualMachines/jdk1.7.0_17.jdk/Contents/Home/jre/lib
        </value>
      </entry>
    ▼<entry>
        <key xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">java.vm.version</key>
        <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">23.7-b01</value>
      </entry>
    ▼<entry>
        <key xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">gopherProxySet</key>
        <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">false</value>
      </entry>
    ▼<entry>
```