

 免費電子書

學習

# playframework

Free unaffiliated eBook created from  
**Stack Overflow contributors.**

#playframe

work

.....	1
<b>1: playframework</b> .....	<b>2</b>
.....	2
Examples.....	2
1.....	2
.....	<b>2</b>
.....	<b>2</b>
.....	2
Mac OS X.....	2
Linux.....	2
.....	2
`sbt`.....	2
Play 2.4.x / 2.5.x - WindowsJava.....	3
.....	<b>3</b>
2.5.....	4
<b>CLI</b> .....	<b>4</b>
.....	5
<b>2: Java - Hello World</b> .....	<b>6</b>
.....	6
Examples.....	6
.....	6
Activator.....	6
.....	6
Hello World“Hello World”.....	7
<b>3: Java - JSON</b> .....	<b>9</b>
.....	9
Examples.....	9
JSON.....	9
/json.....	9
.....	9
.....	9

JSON.....	9
.....	10
.....	10
/.....	10
.....	10
“”.....	10
JSONJava.....	10
JSONJava.....	10
JavaJSON.....	11
JSONJSON.....	11
JSON.....	11
<b>4: WebWSCClient.....</b>	<b>12</b>
.....	12
Examples.....	12
Scala.....	12
<b>5: JSON - Scala.....</b>	<b>13</b>
.....	13
Examples.....	13
JSON.....	13
JavaJSON.....	13
JavaBodyParserJSON.....	14
ScalaJSON.....	14
.....	14
/.....	15
<b>Json.....</b>	<b>15</b>
<b>Json.....</b>	<b>15</b>
<b>6: - Java.....</b>	<b>17</b>
Examples.....	17
Guice - Play 2.4,2.5.....	17
<b>Play API-s.....</b>	<b>17</b>
.....	17

@ImplementedBy.....	17
Play.....	18
.....	19
.....	19
<b>7: - Scala.....</b>	<b>20</b>
.....	20
Examples.....	20
.....	20
Play.....	20
.....	21
<b>8:.....</b>	<b>22</b>
Examples.....	22
- JavaPlay 2.4,2.5.....	22
.....	22
.....	22
.....	22
<b>PowerMock.....</b>	<b>23</b>
.....	23
JSON.....	23
.....	24
.....	24
<b>9:.....</b>	<b>25</b>
.....	25
Examples.....	25
.....	25
<b>10:.....</b>	<b>26</b>
Examples.....	26
.....	26
DDL.....	27
<b>11: IDE.....</b>	<b>28</b>
Examples.....	28

IntelliJ IDEA.....	28
.....	28
.....	28
<b>IntelliJ</b> .....	<b>28</b>
.....	28
EclipsePlay IDE - JavaPlay 2.4,2.5.....	28
.....	28
<b>eclipse IDE</b> .....	<b>28</b>
Playeclipse.....	29
<b>eclipse IDE</b> .....	<b>29</b>
<b>eclipse</b> .....	<b>30</b>
Eclipse IDE.....	30
.....	30
EclipseScala.....	30
sbteclipse.....	30
.....	31
.....	<b>32</b>

---

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [playframework](#)

It is an unofficial and free playframework ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official playframework.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to [info@zzzprojects.com](mailto:info@zzzprojects.com)

---

# 1: playframework

playframework。

playframework。 playframework。

## Examples

1

---

PlayJava 6。 Play[GitAnt](#)。

Java`java --version`

PlayJava\$ **JAVA\_HOME**Java。

**play**Python。 UNIXPython 2.5。

---

。

1. Java。
2. [Play](#)。
3. “play”。

## Mac OS X.

Java。

1. [Play/Applications](#) 。
2. `/etc/paths/Applications/play-1.2.5` 。

OS X

1. [HomeBrew](#)
2. `brew install play`

## Linux

JavaSun-JDKOpenJDKgcjLinuxJava

JavaJDK。 PythonPython。

``sbt``

sbtactivator **Play** ◦ ◦

```
# create a new folder
mkdir myNewProject
# launch sbt
sbt
```

build.sbt

```
name := ""myProjectName""

version := "1.0-SNAPSHOT"

offline := true

lazy val root = (project in file(".")).enablePlugins(PlayScala)
scalaVersion := "2.11.6"
# add required dependencies here .. below a list of dependencies I use
libraryDependencies ++= Seq(
  jdbc,
  cache,
  ws,
  filters,
  specs2 % Test,
  "com.github.nscala-time" %% "nscala-time" % "2.0.0",
  "javax.ws.rs" % "jsr311-api" % "1.0",
  "commons-io" % "commons-io" % "2.3",
  "org.asynchttpclient" % "async-http-client" % "2.0.4",
  cache
)

resolvers += "scalaz-bintray" at "http://dl.bintray.com/scalaz/releases"

resolvers ++= Seq("snapshots", "releases").map(Resolver.sonatypeRepo)

resolvers += "Typesafe Releases" at "http://repo.typesafe.com/typesafe/maven-releases/"
```

projectbuild.properties **Play**

```
addSbtPlugin("com.typesafe.play" % "sbt-plugin" % "2.4.3")
```

◦ sbt ◦ sbtactivator ◦

## Play 2.4.x / 2.5.x - WindowsJava

---

1. Java 8 - [Oracle](#) ◦
2. Activator - [www.playframework.com/downloadzipPlay](http://www.playframework.com/downloadzipPlay)

```
c:\Play-2.4.2\activator-dist-1.3.5
```

3. sbt - [www.scala-sbt.org](http://www.scala-sbt.org) ◦



## 1. JAVA\_HOME

```
c:\Program Files\Java\jdk1.8.0_45
```

## 2. PLAY\_HOME

```
c:\Play-2.4.2\activator-dist-1.3.5;
```

## 3. SBT\_HOME

```
c:\Program Files (x86)\sbt\bin;
```

```
%JAVA_HOME%\bin;%PLAY_HOME%;%SBT_HOME%;
```

## 2.5

Play 2.5.32.5。

1. `activator-dist-1.3.10 \ bin \ activator.bat`。 `set SBT_HOME =BIN_DIRECTORY`
2. `activator-dist-1.3.10conf`。
3. `confsbtconfig.txt`。

## CLI

`cmd`。 CLICLI

```
activator new my-play-app play-java
```

```
activator new
```

。

Play 2.4 `project / plugins.sbt`

```
// Use the Play sbt plugin for Play projects
addSbtPlugin("com.typesafe.play" % "sbt-plugin" % "2.4.x")
```

2.4.x. Play 2.5。

`project / build.properties` `sbt`。 `sbt`。 Play 2.4.x

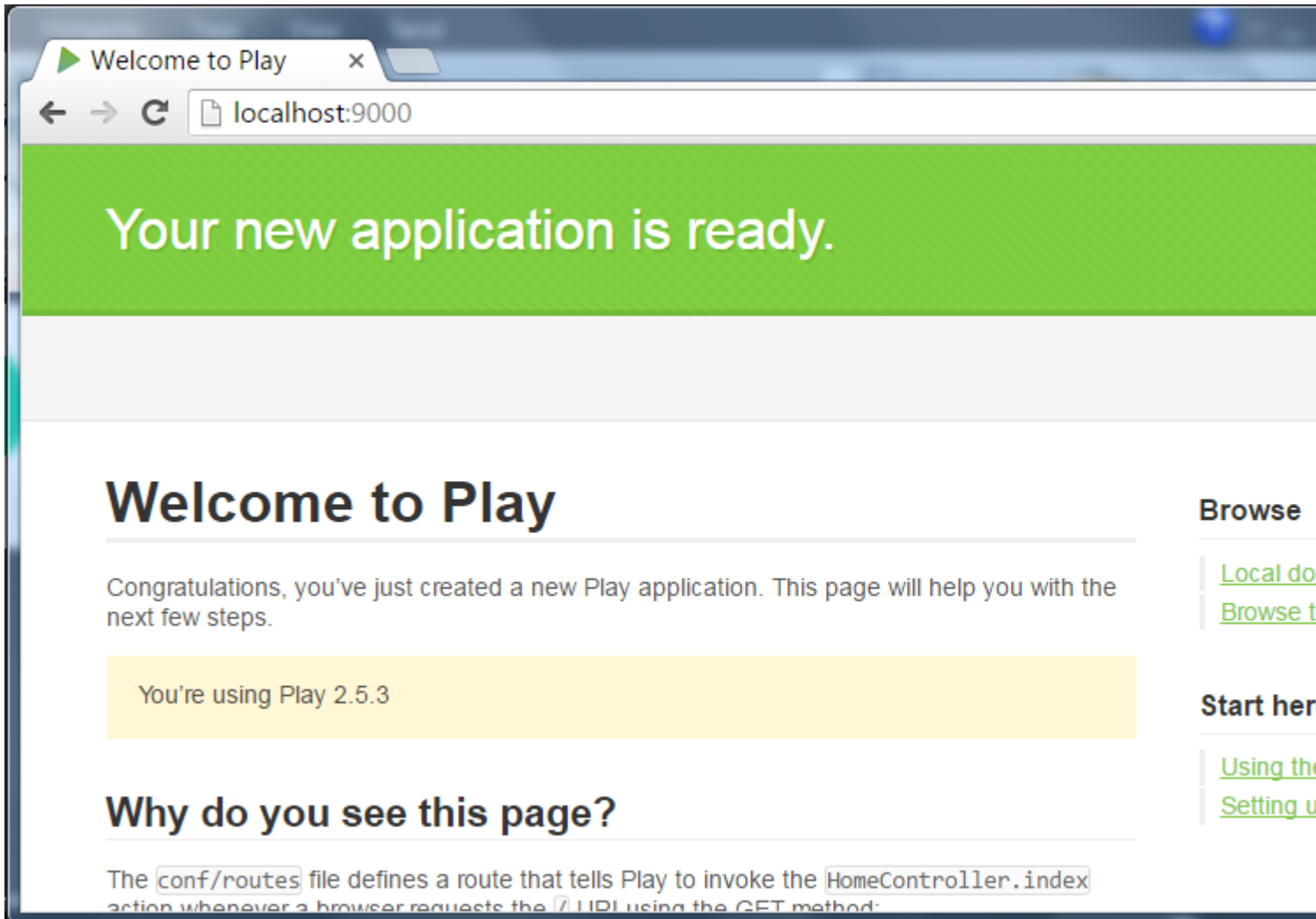
```
sbt.version=0.13.8
```

```
cd my-play-app
```

```
activator run
```

```
[info] p.c.s.NettyServer - Listening for HTTP on /0:0:0:0:0:0:0:9000  
(Server started, use Ctrl+D to stop and go back to the console...)
```

9000.URL [http// localhost9000](http://localhost:9000).



9000http443https. http

```
activator "run 9005"
```

playframework <https://riptutorial.com/zh-TW/playframework/topic/1052/playframework>

## 2: Java - Hello World

- Linux / MacOSPlay

### Examples

HelloWorld play-java

```
$ ~/activator-1.3.10-minimal/bin/activator new HelloWorld play-java
```

```
Fetching the latest list of templates...
```

```
OK, application "HelloWorld" is being created using the "play-java" template.
```

```
To run "HelloWorld" from the command line, "cd HelloWorld" then:
```

```
/home/YourUserName/HelloWorld/activator run
```

```
To run the test for "HelloWorld" from the command line, "cd HelloWorld" then:
```

```
/home/YourUserName/HelloWorld/activator test
```

```
To run the Activator UI for "HelloWorld" from the command line, "cd HelloWorld" then:
```

```
/home/YourUserName/HelloWorld/activator ui
```

### Activator

Play FrameworkActivator。 ActivatorPlay Framework。

ActivatorPlay 1.3.10

tutorialActivator

### Activator

```
To run "HelloWorld" from the command line, "cd HelloWorld" then:
```

```
/home/YourUserName/HelloWorld/activator run
```

activatorbin/activator。

```
bin/activator
```

Activator。。

```
[HelloWorld] $
```

~runActivator。。

Ctrl + DActivator shellCtrl + DOS shell

```
[HelloWorld] $ ~run
```

## Play

```
-- (Running the application, auto-reloading is enabled) ---  
[info] p.c.s.NettyServer - Listening for HTTP on /0:0:0:0:0:0:0:0:9000  
(Server started, use Ctrl+D to stop and go back to the console...)
```

## localhost9000 Play

localhost:9000

Your new application is ready.

# Welcome to Play

Congratulations, you've just created a new Play application. This page will help you with the next few steps.

You're using Play 2.5.4

## Hello World“Hello World”

“Hello World”Hello World. ◦

app/controllers/HomeController.java

```
public Result hello() {  
    return ok("Hello world!");  
}
```

conf/routes

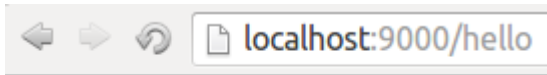
```
GET    /hello                controllers.HomeController.hello
```

## Play

```
[info] Compiling 4 Scala sources and 1 Java source to  
/home/YourUserName/HelloWorld/target/scala-2.11/classes...
```

```
[success] Compiled in 4s
```

localhost9000 / hello hello world



**Hello world!**

Java - Hello World <https://riptutorial.com/zh-TW/playframework/topic/5887/java-----hello-world>

# 3: Java - JSON

<https://www.playframework.com/documentation/2.5.x/JavaJsonActions>

## Examples

### JSON

```
import play.libs.Json;

public JsonNode createJson() {
    // {"id": 33, "values": [3, 4, 5]}
    ObjectNode rootNode = Json.newObject();
    ArrayNode listNode = Json.newArray();

    long values[] = {3, 4, 5};
    for (long val: values) {
        listNode.add(val);
    }

    rootNode.put("id", 33);
    rootNode.set("values", listNode);
    return rootNode;
}
```

### /json

```
import play.libs.Json;
// (...)
```

```
// Note: "app" is an play.Application instance
JsonNode node = Json.parse(app.resourceAsStream("public/myjson.json"));
```

```
String myStr = "{\"name\": \"John Doe\"}";
JsonNode node = Json.parse(myStr);
```

### JSON

jsonJSON

```
[
  {
    "name": "John Doe",
    "work": {
      "company": {
        "name": "ASDF INC",
        "country": "USA"
      },
      "cargo": "Programmer"
    }
  }
]
```

```

    },
    "tags": ["java", "jvm", "play"]
  },
  {
    "name": "Bob Doe",
    "work": {
      "company": {
        "name": "NOPE INC",
        "country": "AUSTRALIA"
      },
      "cargo": "SysAdmin"
    },
    "tags": ["puppet", "ssh", "networking"],
    "active": true
  }
]

```

```

JsonNode node = json.get(0).get("name"); // --> "John Doe"
// This will throw a NullPointerException, because there is only two elements
JsonNode node = json.get(2).get("name"); // --> *crash*

```

```

JsonNode node1 = json.at("/0/name"); // --> TextNode("John Doe")
JsonNode node2 = json.at("/2/name"); // --> MissingNode instance
if (! node2.isMissingNode()) {
    String name = node2.asText();
}

```

/

```

JsonNode node2 = json.at("/0/work/company/country"); // TextNode("USA")

```

```

List<JsonNode> d = json.findValues("country"); // List(TextNode("USA"), TextNode("AUSTRALIA"))

```

“”

```

List<JsonNode> e = json.findParents("active"); // List(ObjectNode("Bob Doe"))

```

## JSONJava

JacksonPlay JSONjason。 getter / setter。 Book ISBN getISBN/setISBN get / setJackson

- JavaJSON“ISBN”JSON
- setISBNJava setISBN isbnJSON“ISBN”。

## JSONJava

```

public class Person {
    String id, name;
}

```

```
}  
  
JsonNode node = Json.parse("{\"id\": \"3S2F\", \"name\", \"Salem\"}");  
Person person = Json.fromJson(node, Person.class);  
System.out.println("Hi " + person.name); // Hi Salem
```

## JavaJSON

```
// "person" is the object from the previous example  
JsonNode personNode = Json.toJson(person)
```

## JSONJSON

```
// personNode comes from the previous example  
String json = personNode.toString();  
// or  
String json = Json.stringify(json);
```

## JSON

```
System.out.println(personNode.toString());  
/* Prints:  
{ "id": "3S2F", "name": "Salem" }  
*/  
  
System.out.println(Json.prettyPrint(personNode));  
/* Prints:  
{  

```

Java - JSON <https://riptutorial.com/zh-TW/playframework/topic/6318/java---json>



---

## 4: WebWSClient

<https://www.playframework.com/documentation/2.5.x/ScalaWS>

### Examples

#### Scala

##### HTTPWSClient

```
import javax.inject.Inject

import play.api.libs.ws.WSClient

import scala.concurrent.{ExecutionContext, Future}

class MyClass @Inject() (
  wsClient: WSClient
)(implicit ec: ExecutionContext){

  def doGetRequest(): Future[String] = {
    wsClient
      .url("http://www.google.com")
      .get()
      .map { response =>
        // Play won't check the response status,
        // you have to do it manually
        if ((200 to 299).contains(response.status)) {
          println("We got a good response")
          // response.body returns the raw string
          // response.json could be used if you know the response is JSON
          response.body
        } else
          throw new IllegalStateException(s"We received status ${response.status}")
        }
      }
  }
}
```

**WebWSClient** <https://riptutorial.com/zh-TW/playframework/topic/2981/webwsclient>

# 5: JSON - Scala

## Playplay json

```
"com.typesafe.play" % "play-json_2.11" % "2.5.3"build.sbt
```

- [https://mvnrepository.com/artifact/com.typesafe.play/play-json\\_2.11](https://mvnrepository.com/artifact/com.typesafe.play/play-json_2.11)
- [Play JSON Librarysbt](#)

## Examples

### JSON

#### JSON JsValue

```
import play.api.libs.json._

val json = JsObject (Map(
  "name" -> JsString("Jsony McJsonface"),
  "age" -> JsNumber(18),
  "hobbies" -> JsArray(Seq(
    JsString("Fishing"),
    JsString("Hunting"),
    JsString("Camping")
  ))
))
```

```
import play.api.libs.json._

val json = Json.obj(
  "name" -> "Jsony McJsonface",
  "age" -> 18,
  "hobbies" -> Seq(
    "Fishing",
    "Hunting",
    "Camping"
  )
)
```

### JSON

```
json.toString
// {"name":"Jsony McJsonface","age":18,"hobbies":["Fishing","Hunting","Camping"]}
Json.prettyPrint(json)
// {
//   "name" : "Jsony McJsonface",
//   "age" : 18,
//   "hobbies" : [ "Fishing", "Hunting", "Camping" ]
// }
```

### JavaJSON

```

public Result sayHello() {
    JsonNode json = request().body().asJson();
    if(json == null) {
        return badRequest("Expecting Json data");
    } else {
        String name = json.findPath("name").textValue();
        if(name == null) {
            return badRequest("Missing parameter [name]");
        } else {
            return ok("Hello " + name);
        }
    }
}

```

## JavaBodyParserJSON

```

@BodyParser.Of(BodyParser.Json.class)
public Result sayHello() {
    JsonNode json = request().body().asJson();
    String name = json.findPath("name").textValue();
    if(name == null) {
        return badRequest("Missing parameter [name]");
    } else {
        return ok("Hello " + name);
    }
}

```

*PlayHTTP400Content-type* `application/json` JSON

## ScalaJSON

### JSON

```

val str =
  """{
    |   "name" : "Jsony McJsonface",
    |   "age" : 18,
    |   "hobbies" : [ "Fishing", "Hunting", "Camping" ],
    |   "pet" : {
    |     "name" : "Doggy",
    |     "type" : "dog"
    |   }
  |}""".stripMargin

```

### JSONJsValue

```
val json = Json.parse(str)
```

```
(json \ "name").as[String] // "Jsony McJsonface"
```

- \JSON
- \\JSON

- `.apply(idx) (idx)`
- `.as[T]`
- `.asOpt[T] None`
- `.validate[T] JSONJsSuccessJsError`

```
(json \ "name").as[String] // "Jsony McJsonface"
(json \ "pet" \ "name").as[String] // "Doggy"
(json \\ "name").map(_.as[String]) // List("Jsony McJsonface", "Doggy")
(json \\ "type")(0).as[String] // "dog"
(json \ "wrongkey").as[String] // throws JsResultException
(json \ "age").as[Int] // 18
(json \ "hobbies").as[Seq[String]] // List("Fishing", "Hunting", "Camping")
(json \ "hobbies")(2).as[String] // "Camping"
(json \ "age").asOpt[String] // None
(json \ "age").validate[String] // JsError containing some error detail
```

/

## JSONJSON

```
case class Person(
  name: String,
  age: Int,
  hobbies: Seq[String],
  pet: Pet
)

case class Pet(
  name: String,
  `type`: String
)

// these macros will define automatically the conversion to/from JSON
// based on the cases classes definition
implicit val petFormat = Json.format[Pet]
implicit val personFormat = Json.format[Person]
```

## Json

```
val person = Person(
  "Jsony McJsonface",
  18,
  Seq("Fishing", "Hunting", "Camping"),
  Pet("Doggy", "dog")
)

Json.toJson(person).toString
// {"name":"Jsony
McJsonface","age":18,"hobbies":["Fishing","Hunting","Camping"],"pet":{"name":"Doggy","type":"dog"}}
```

## Json

```
val str =
  """{
  |   "name" : "Jsony McJsonface",
  |   "age" : 18,
  |   "hobbies" : [ "Fishing", "Hunting", "Camping" ],
  |   "pet" : {
  |     "name" : "Doggy",
  |     "type" : "dog"
  |   }
  |}""".stripMargin

Json.parse(str).as[Person]
// Person(Jsony McJsonface,18,List(Fishing, Hunting, Camping),Pet(Doggy,dog))
```

**JSON - Scala** <https://riptutorial.com/zh-TW/playframework/topic/2983/json-----scala>

---

## 6: - Java

### Examples

#### Guice - Play 2.4,2.5

GuicePlayDI ◦ GuicePlay◦

---

## Play API-s

Play 2.5DIAPI-sAPI◦ *Configuration JPAApi CacheApi*

Play APIPlay◦ *@Inject Configuration*

```
@Inject
private Configuration configuration;
```

```
private Configuration configuration;
@Inject
public MyController(Configuration configuration) {
    this.configuration = configuration;
}
```

DI - *@Inject*

DI *Play.current*◦ *injector*◦ *Configuration*

```
private Configuration configuration =
Play.current().injector().instanceOf(Configuration.class);
```

---

**@ImplementedByGuice**◦

## @ImplementedBy

@ImplementedBy◦ ◦

### 1. *CacheProvider*

```
@ImplementedBy(RuntimeCacheProvider.class)
public interface CacheProvider {
    CacheApi getCache();
}
```

### 2. *RunTimeCacheProvider*

```

public class RunTimeCacheProvider implements CacheProvider {
    @Inject
    private CacheApi appCache;
    @Override
    public CacheApi getCache() {
        return appCache;
    }
}

```

*RunTimeCacheProvider* *appCache*.

### 3. @Inject

```

public class HomeController extends Controller {
    @Inject
    private CacheProvider cacheProvider;
    ...
    public Result getCacheData() {
        Object cacheData = cacheProvider.getCache().get("DEMO-KEY");
        return ok(String.format("Cache content:%s", cacheData));
    }
}

```

*@ImplementedBy* *CacheProvider* *RunTimeCacheProvider*. . . *@ ImplementedBy*. **Guice**.

## Play

### PlayModule

```

import com.google.inject.AbstractModule;
public class Module extends AbstractModule {
    @Override
    protected void configure() {
        // bindings are here
    }
}

```

*configure*.

*CacheProvider* *RunTimeCacheProvider*

### 1. CacheProvider @ImplementedBy

```

public interface CacheProvider {
    CacheApi getCache();
}

```

### 2.

```

public class Module extends AbstractModule {
    @Override
    protected void configure() {
        bind(CacheProvider.class).to(RunTimeCacheProvider.class);
    }
}

```

```
}
```

*RuntimeCacheProviderJUnit* ◦ *CacheProvider* ◦ ◦

◦

1. *FakeCacheCacheApi* - ◦
2. *FakeCacheProviderCacheProvider*

```
public class FakeCacheProvider implements CacheProvider {
    private final CacheApi fakeCache = new FakeCache();
    @Override
    public CacheApi getCache() {
        return fakeCache;
    }
}
```

2.

```
public class Module extends AbstractModule {
    private final Environment environment;
    public Module(Environment environment, Configuration configuration) {
        this.environment = environment;
    }
    @Override
    protected void configure() {
        if (environment.isTest() ) {
            bind(CacheProvider.class).to(FakeCacheProvider.class);
        }
        else {
            bind(CacheProvider.class).to(RuntimeCacheProvider.class);
        }
    }
}
```

◦ ◦ ◦ ...◦

Play ◦ ◦ *OnStartupModule* ◦

```
package modules;
import com.google.inject.AbstractModule;
public class OnStartupModule extends AbstractModule {
    @Override
    protected void configure() {
        ...
    }
}
```

Play ◦ *OnStartupModule application.conf*

```
play.modules.enabled += "modules.OnStartupModule"
```

- Java <https://riptutorial.com/zh-TW/playframework/topic/6060/----java>



## 7: - Scala

- class MyClassUsingAnother @InjectmyOtherClassInjectedMyOtherClass{...}
- @SingletonMyClassThatShouldBeASingleton...

### Examples

```
import javax.inject._
@Singleton
class BurgersRepository {
    // implementation goes here
}
```

◦

```
import javax.inject._
class FastFoodService @Inject() (burgersRepository: BurgersRepository) {
    // implementation goes here
    // burgersRepository can be used
}
```

◦ **FastFoodService** ◦

```
import javax.inject._
import play.api.mvc._
@Singleton
class EatingController @Inject() (fastFoodService: FastFoodService) extends Controller {
    // implementation goes here
    // fastFoodService can be used
}
```

### Play

WSClientConfiguration ◦

```
class ComplexService @Inject() (
    configuration: Configuration,
    wsClient: WSClient,
    applicationLifecycle: ApplicationLifecycle,
    cacheApi: CacheApi,
    actorSystem: ActorSystem,
    executionContext: ExecutionContext
) {
    // Implementation goes here
    // you can use all the injected classes :
    //
    // configuration to read your .conf files
    // wsClient to make HTTP requests
    // applicationLifecycle to register stuff to do when the app shutdowns
    // cacheApi to use a cache system
    // actorSystem to use AKKA
    // executionContext to work with Futures
}
```

```
}
```

## ExecutionContext

```
class ComplexService @Inject() (
  configuration: Configuration,
  wsClient: WSClient
)(implicit executionContext: ExecutionContext) {
  // Implementation goes here
  // you can still use the injected classes
  // and executionContext is imported as an implicit argument for the whole class
}
```

◦ ◦ ◦

```
import com.google.inject.AbstractModule
// Play will automatically use any class called `Module` that is in the root package
class Module extends AbstractModule {

  override def configure() = {
    // Here you can put your customisation code.
    // The annotations are still used, but you can override or complete them.

    // Bind a class to a manual instantiation of it
    // i.e. the FunkService needs not to have any annotation, but can still
    // be injected in other classes
    bind(classOf[FunkService]).toInstance(new FunkService)

    // Bind an interface to a class implementing it
    // i.e. the DiscoService interface can be injected into another class
    // the DiscoServiceImplementation is the concrete class that will
    // be actually injected.
    bind(classOf[DiscoService]).to(classOf[DiscoServiceImplementation])

    // Bind a class to itself, but instantiates it when the application starts
    // Useful to executes code on startup
    bind(classOf[HouseMusicService]).asEagerSingleton()
  }
}
```

- Scala <https://riptutorial.com/zh-TW/playframework/topic/3020/----scala>

---

# 8:

## Examples

### - JavaPlay 2.4,2.5

---

Class *Helpers*. *PlayHTTPCookie* - *Play*. *HelpersfakeApplication*. *HelpersfakeApplication*  
*WithApplication* .

#### *Helpers* API

```
Helpers.running(Application application, final Runnable block);  
Helpers.fakeApplication();
```

```
public class TestController extends WithApplication {  
    @Test  
    public void testSomething() {  
        Helpers.running(Helpers.fakeApplication(), () -> {  
            // put test stuff  
            // put asserts  
        });  
    }  
}
```

#### Helpersimport

```
import static play.test.Helpers.fakeApplication;  
import static play.test.Helpers.running;  
...  
@Test  
public void testSomething() {  
    running(fakeApplication(), () -> {  
        // put test stuff  
        // put asserts  
    });  
}
```

```
}
```

---

URL. Java . *Play*. *Call* .

#### test *Helpers* API

```
Result result = Helpers.route(Helpers.fakeRequest(Call action));
```

1.

```
GET /conference/:confId    controllers.ConferenceController.getConfId(confId: String)
POST /conference/:confId/participant
controllers.ConferenceController.addParticipant (confId:String)
```

```
2. controllers.routes.ConferenceController.getConfId (conferenceId)
   controllers.routes.ConferenceController.addParticipant (conferenceId)
```

### 3. *getConfId*GET ◦

```
Result result =
Helpers.route (Helpers.fakeRequest (controllers.routes.ConferenceController.getConfId (conferenceId))
```

### 4. *addParticipant*POST ◦ ◦

```
ParticipantDetails inputData = DataSimulator.createParticipantDetails ();
Call action = controllers.routes.ConferenceController.addParticipant (conferenceId);
Result result = route (Helpers.fakeRequest (action).bodyJson (Json.toJson (inputData)) );
```

---

## PowerMock

```
@RunWith (PowerMockRunner.class)
@PowerMockIgnore ({"javax.management.*", "javax.crypto.*"})
public class TestController extends WithApplication {
....
```

### *RequestBuilder*

```
RequestBuilder fakeRequest = Helpers.fakeRequest (action);
```

### *addParticipant*

```
RequestBuilder mockActionRequest =
Helpers.fakeRequest (controllers.routes.ConferenceController.addParticipant (conferenceId));
```

```
Result result = Helpers.route (mockActionRequest);
```

```
@Test
public void testLoginOK() {
    running(fakeApplication(), () -> {
        /**whatever mocking*/Mockito.when(...).thenReturn(...);
        RequestBuilder mockActionRequest = Helpers.fakeRequest (
            controllers.routes.LoginController.loginAdmin());
        Result result = route(mockActionRequest);
        assertEquals (OK, result.status());
    });
}
```

# JSON

T. ◦

1

```
public static <T> RequestBuilder fakeRequestWithJson(T input, String method, String url) {
    JsonNode jsonNode = Json.toJson(input);
    RequestBuilder fakeRequest = Helpers.fakeRequest(method, url).bodyJson(jsonNode);
    System.out.println("Created fakeRequest="+fakeRequest +",
body="+fakeRequest.body().asJson());
    return fakeRequest;
}
```

2

```
public static <T> RequestBuilder fakeActionRequestWithJson(Call action, T input) {
    JsonNode jsonNode = Json.toJson(input);
    RequestBuilder fakeRequest = Helpers.fakeRequest(action).bodyJson(jsonNode);
    System.out.println("Created fakeRequest="+fakeRequest +",
body="+fakeRequest.body().asJson());
    return fakeRequest;
}
```

```
public static final String BASIC_AUTH_VALUE = "dummy@com.com:12345";
public static RequestBuilder fakeActionRequestWithBaseAuthHeader(Call action) {
    String encoded = Base64.getEncoder().encodeToString(BASIC_AUTH_VALUE.getBytes());
    RequestBuilder fakeRequest =
Helpers.fakeRequest(action).header(Http.HeaderNames.AUTHORIZATION,
                                "Basic " + encoded);
    System.out.println("Created fakeRequest="+fakeRequest.toString() );
    return fakeRequest;
}
```

```
public static final String FAKE_SESSION_ID = "12345";
public static RequestBuilder fakeActionRequestWithSession(Call action) {
    RequestBuilder fakeRequest = RequestBuilder fakeRequest =
Helpers.fakeRequest(action).session("sessionId", FAKE_SESSION_ID);
    System.out.println("Created fakeRequest="+fakeRequest.toString() );
    return fakeRequest;
}
```

Play *SessionHashMap* <StringString>◦

```
public static Http.Session fakeSession() {
    return new Http.Session(new HashMap<String, String>());
}
```

<https://riptutorial.com/zh-TW/playframework/topic/6192/>

---

# 9:

•

## Examples

scripts

1.

2. build.sbt

```
import NativePackagerHelper._
```

3. build.sbt

```
mappings in Universal += directory("scripts")
```

4. **activator dist** target/universal/

<https://riptutorial.com/zh-TW/playframework/topic/6642/>

# 10:

## Examples

build.sbt **MysqlPostGreSQL**

```
"mysql" % "mysql-connector-java" % "5.1.20",
"org.postgresql" % "postgresql" % "9.3-1100-jdbc4",
"com.typesafe.slick" %% "slick" % "3.1.1",
"com.typesafe.play" %% "play-slick" % "1.1.1"
```

application.conf

```
mydb.driverjava="slick.driver.MySQLDriver$"
mydb.driver="com.mysql.jdbc.Driver"
mydb.url="jdbc:mysql://hostaddress:3306/dbname?zeroDateTimeBehavior=convertToNull"
mydb.user="username"
mydb.password="password"
```

## RDBMS

```
package mypackage

import slick.driver.MySQLDriver
import slick.driver.PostgresDriver

object SlickDBDriver{
  val env = "something here"
  val driver = env match{
    case "postGreCondition" => PostgresDriver
    case _                   => MySQLDriver
  }
}
```

```
import mypackage.SlickDBDriver.driver.api._
import slick.lifted.{TableQuery, Tag}
import slick.model.ForeignKeyAction

case class MyModel(
  id: Option[Long],
  name: String
) extends Unique

class MyModelDB(tag: Tag) extends IndexedTable[MyModel](tag, "my_table"){
  def id          = column[Long]("id", O.PrimaryKey, O.AutoInc)
  def name        = column[String]("name")

  def * = (id.? , name) <> ((MyModel.apply _).tupled, MyModel.unapply _)
}

class MyModelCrud{
  import play.api.Play.current
```

```
val dbConfig = DatabaseConfigProvider.get[JdbcProfile](Play.current)
val db = dbConfig.db

val query = TableQuery[MyModelDB]

// SELECT * FROM my_table;
def list = db.run{query.result}
}
```

## DDL

SQL。。

```
val table = TableQuery[MyModel] SQL - DDL
```

```
import mypackage.SlickDBDriver.driver.api._
table.schema.createStatement
```

<https://riptutorial.com/zh-TW/playframework/topic/4604/>



---

# 11: IDE

## Examples

### IntelliJ IDEA

---

1. IntelliJ IDEA
2. IntelliJScala
3. PlayActivator `activator new [nameoftheproject] play-scala` ◦

- 
1. IntelliJ IDEA
  2. File > Open ... >[nameoftheproject]> OK
  3. ◦ ◦ OK
  4. IntelliJ IDEA ◦ `rootroot-build` ◦ “OK ◦
  5. IntelliJ ◦ IntelliJ ◦

---

## IntelliJ

IDE/`sbt//` ◦ IntelliJ ◦ ◦

1. Run > Edit configurations...
2. + >Play 2 App
3. [nameofyourproject] ◦ “OK ◦
4. “RunUIRunDebug ◦ `Runsbt run` ◦ Debug ◦

---

IntelliJ IDEA > Preferences > Build, Execution, Deployment > Build tools > SBT > Project-level settings > Use auto-import ◦

Scalaimport ◦ IntelliJ IDEA`build.sbt` ◦ IntelliJ IDEA ◦ IntelliJ ◦ IntelliJ`build.sbt` ◦ ◦ UISBT ◦

### EclipsePlay IDE - JavaPlay 2.4,2.5

---

PlayIDE ◦ **eclipseactivator** `eclipsePlayeclipse` ◦ Eclipse**sbt** ◦ ◦ eclipse IDE ◦ <http://scala-ide.org/download/sdk.html>ScalaJava 8 lunamars eclipse ◦

---

## eclipse IDE

## Playeclipse

### 1. *eclipseproject / plugins.sbt*

```
//Support Play in Eclipse
addSbtPlugin("com.typesafe.sbteclipse" % "sbteclipse-plugin" % "4.0.0")
```

### 2. *build.sbteclipse*

```
EclipseKeys.preTasks := Seq(compile in Compile)
```

### 3. {user root} .sbt \ repositories. ◦ *activator-launcher-localactivator-local*

```
activator-local: file:///${activator.local.repository-C:/Play-2.5.3/activator-dist-
1.3.10//repository},
[organization]/[module]/(scala_[scalaVersion]/)(sbt_[sbtVersion]/)[revision]/[type]s/[artifact](-
[classifier]).[ext]
activator-launcher-local: file:///${activator.local.repository-${activator.home-
${user.home}/.activator}/repository},
[organization]/[module]/(scala_[scalaVersion]/)(sbt_[sbtVersion]/)[revision]/[type]s/[artifact](-
[classifier]).[ext]
```

### 4.

```
activator compile
```

### 5. eclipse

```
activator eclipse
```

## Existing Projects into espaceclipse. ◦

## Playeclipse

### 1. *build.sbt*

```
EclipseKeys.withSource := true
```

### 2.

## eclipse IDE

### sbt

### 1. *.sbt \ 0.13 \ pluginsplugins.sbt* ◦ *Windowsasch*

```
c:\asch\.sbt\0.13\plugins\plugins.sbt
```

## 2. *eclipseplugins.sbt*

```
//Support Play in Eclipse  
addSbtPlugin("com.typesafe.sbteclipse" % "sbteclipse-plugin" % "4.0.0")
```

## 3. *.sbtsbteclipse.sbt* ◦ Windowsasch

```
c:\asch\.sbt\0.13\sbteclipse.sbt
```

## 4. *sbteclipse.sbtactivator eclipse*

```
import com.typesafe.sbteclipse.plugin.EclipsePlugin.EclipseKeys  
EclipseKeys.preTasks := Seq(compile in Compile)
```

## 5. EclipseKeys◦

# eclipse

9999

```
activator -jvm-debug run
```

```
activator -jvm-debug [port] run
```

1. **Debug As Debug Configurations** ◦
2. **Debug ConfigurationsRemote Java ApplicationNew** ◦
3. 9999“ ” ◦

**Debug**◦ ◦

## Eclipse IDE

1. Java81.8.0\_91
2. Eclipse neonJavaScriptWeb Developer
3. Play Framework 2.5.4

## EclipseScala

1. Eclipse
2. “ Help >“ Eclipse Marketplace
3. FindScala
4. Scala IDE

## sbteclipse

1. `.\project\ plugins.sbt`

2. `plugins.sbt` **eclipse**

`addSbtPlugin“com.typesafe.sbteclipse”“sbteclipse-plugin”“4.0.0”`

3. `cd C:\play\play-scala` ◦

1. **Eclipse**File > Import

2. Existing Projects into Workspace

3.

Eclipse IDE ◦

IDE <https://riptutorial.com/zh-TW/playframework/topic/4437/ide>

S. No		Contributors
1	playframework	<a href="#">Abhinab Kanrar</a> , <a href="#">Anton</a> , <a href="#">asch</a> , <a href="#">Community</a> , <a href="#">implicitdef</a> , <a href="#">James</a> , <a href="#">John</a> , <a href="#">robguinness</a>
2	Java - Hello World	<a href="#">Salem</a>
3	Java - JSON	<a href="#">Salem</a>
4	WebWSClient	<a href="#">implicitdef</a> , <a href="#">John</a> , <a href="#">Salem</a>
5	JSON - Scala	<a href="#">Anton</a> , <a href="#">asch</a> , <a href="#">implicitdef</a> , <a href="#">John</a> , <a href="#">Salem</a>
6	- Java	<a href="#">asch</a>
7	- Scala	<a href="#">asch</a> , <a href="#">implicitdef</a>
8		<a href="#">asch</a>
9		<a href="#">Juliend</a>
10		<a href="#">John</a>
11	IDE	<a href="#">Alice</a> , <a href="#">asch</a> , <a href="#">implicitdef</a>