digitalis

DiF v2.x

Digitalis Framework
Comercial Presentation

Development Department

Digitalis Informática ©2008

Pedro Viegas (pviegas@digitalis.pt)

Outline

Presentation outline

- What’s DiF?
- Why DiF?
- What’s new in v2.0?
- Architecture and it’s benefits
- DiF’ Features
- Built on top of Best-of-breed
- Tools
- Resources
- Who’s using DiF?
- What’s next?
- Summary

What’s DiF?

DiF in a nutshell

- Open-source J2EE framework
- Component-driven development
- Pluggable architecture, enables integrations/customizations
- Adaptable Framework
- No configuration files, 100% java code!
- Multi-channel, not just Web
- RAD – Rapid Application Development
- Focus on performance
- Web 2.0 UI
- Focus on Quality (QAG)
- Convention vs Configuration vs Programming
- Development Platform created and used by Digitalis Informática lda

Why DiF?

Reinvent the wheel?

- Do we place ourselves above these Gurus and others?
  - Craig McClanahan (Struts)
  - Rod Johnson (Spring)
  - Howard Lewis Ship (Tapestry)
  - Ed Burns, Craig McClanahan "and a whole bunch of brain power" (JSF – JSR-127)
- What's the DiF's vision that we could not find in other solutions?
  - Java does not have to be complex
  - No need to be a mechanic to drive a car
  - Productivity in Java MUST be as high as in any other RAD
  - Repetitions are bad...
    - If it's common, the framework must be ready to help
  - I should no have to use a new framework for each kind of project, due to the channel I'm using. I should be able to switch plug-ins and use mainly the same framework
  - If it is too much trouble...
    - Something is missing in the framework!
  - This is not how I want this to be done...
    - Switch framework plug-ins...
    - Or create your own!

Why DiF?

Reasons for choosing DiF for a MANAGER

- Open-source framework
- Standards
- Uses the best technological solutions available
- Certified for several Application, RDBMS and other servers
- Maximizes performance
- Minimizes development time
- Minimizes need for technical skills of developers
- Pluggable platform, highly customizable and extensible
- Build with integration needs in mind
- Out-of-the-box
  - Integration with LDAP, RDBMS, Identity/Authorization management and other servers.
  - Administration interfaces
  - Component library that contributes for a common user experience in DiF based products
  - Usability and accessibility
  - Internationalization
  - Etc.
- The Software-house that created DiF offers support development services

Why DiF?

Reasons for choosing DiF for a <u>DEVELOPER</u>

- J2EE framework
- Feature rich
  - Several UI Web 2.0 components
  - Management of Session, user, group, ACL, exception, logging and others
  - Parameter management and conversion
  - Configurations
- Adapts to the developer code
- Abundant code generation
- Integrates several tools and APIs out-of-the-box
- Favors component-driven development
- Maximizes performance
- IoC centric, completely pluggable and extensible
- Multi-channel, abstracts the developer from channel specifics
- Development environment (light and simple!)
- 100% java code (+Annotations). No need for XML, BD, or any other meta-data repository

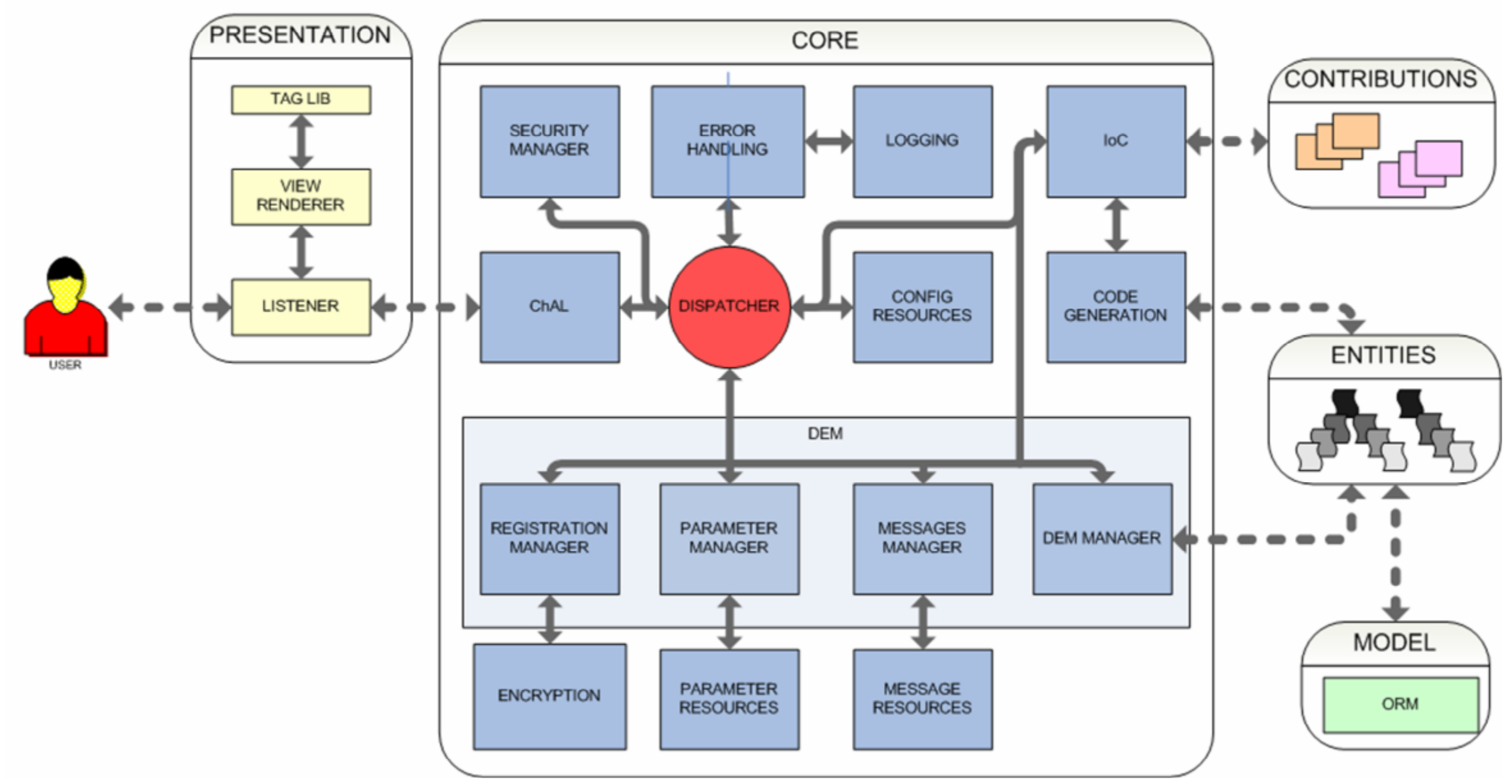
What's new in DiF v2.x

What's new from the previous version

- Multi-channel
- No more external meta-data repository. Annotation-centric Model
- Component-driven development
- Fully pluggable, even inside it's Core
- IoC as the integration central piece for all modules/plugins
- Code generation for several needs
- Web UI 2.0 (several components, accessibility, usability)
- Hibernate integrated for ORM, with code generation (+/- 80%)
- Automated project management: Maven
- Several application servers supported
- More...
  - Productivity
  - Performance
  - Simplicity
  - Flexibility

DiF2 Architecture

New platform's architecture description



DiF2 Architecture

New platform's architecture description

- JDK Java 5.0
  - Extensive use of annotations
- MVC Architecture (multi-channel)
- Servlet 2.1 for HTTP Channel (Web Applications)
- Guice IoC container
  - Integration with external modules, and, for all internal framework modules
  - Fully pluggable architecture
- Certified with:
  - JBoss
  - Jetty
  - Tomcat

DiF2 Architecture

Model View Controller –Common implementations

- Model View Controller in Web frameworks
  - The controller is a servlet. Stuck to the Servlet API
  - The flow and request processing if often hard-coupled
  - Can't use the Model easily with other channels, least of all, the Views
- MVC + Front Controller
  - Separates the flow from request/response processing. Does not solve the other issues though.
- Several Multi-channel MVN implementations
  - Replicate some controller logic in many channel implementations
  - Do not fully abstract the Model from the Channel, allowing the use of specific APIs like Servlet API, SOAP and others

DiF2 Architecture

DiF2 MVC, a Multi-Channel MVC that simplifies development

- Model
  - Simple annotated POJOs. Total freedom, no need to use any specific channel API
- View
  - Several possibilities
  - For the moment JSPViewRenderer for HTTP. Other are coming, like PDF, Freemarker, etc.
  - Add your own!
- Controller
  - Listener
  - ChAL
  - Dispatcher

DiF2 Architecture

ChAL & Dispatcher

- Listener
  - The listener receives the request and send it to the ChAL. These are channel dependent naturally.

They are the entrances. Web requests, SMS gateways connections, email clients, etc.

DiF2 Architecture

ChAL & Dispatcher

- Listener
  - The listener receives the request and send it to the ChAL. These are channel dependent naturally.
- ChAL
  - Converts requests from the channel format to the standard DiF format and vice-versa.
  - Provides the DiF developers abstraction from the Channel. It's specific needs are here processed and restructured in a simple and common format. Client info are represented in a standard client object, attributes and parameters are placed in the functional objects of DiF.

Translators to the DiF standard. Enable that the same development process can be used for any channel

DiF2 Architecture

ChAL & Dispatcher

- Listener
  - The listener receives the request and send it to the ChAL. These are channel dependent naturally.
- ChAL
  - Converts requests from the channel format to the standard DiF format and vice-versa.
  - Provides the DiF developers abstraction from the Channel. It's specific needs are here processed and restructured in a simple and common format. Client info are represented in a standard client object, attributes and parameters are placed in the functional objects of DiF.
- Dispatcher
  - **Each channel can have different authentication/authorization/publishing/execution steps. We can, if necessary, provide a different implementation for each step for each channel.**

The flow and steps to process a request. Can be customized for each channel, even externally to the framework's core.

DiF2 Architecture

ChAL & Dispatcher

- Listener
  - The listener receives the request and send it to the ChAL. These are channel dependent naturally.
- ChAL
  - Converts requests from the channel format to the standard DiF format and vice-versa.
  - Provides the DiF developers abstraction from the Channel. It's specific needs are here processed and restructured in a simple and common format. Client info are represented in a standard client object, atributes and parameters are placed in the functional objects of DiF.
- Dispatcher
  - Each channel can have different authentication/authorization/publishing/execution steps. We can, if necessary, provide a different implementation for each step for each channel.
- New implementations
  - DiF can receive new implementations of these modules and, without any additional configuration, handles requests in these new channels. In theory a service developed for a channel can be executed in another one, given the special attention that the view can render for the new channel.

DiF's features?

Main features of DiF

DiF's features?

DEM: DiF Entity Model

- SOA Hierarchy
  - All services are structured in a set hierarchy:
    - Provider – Application – Service - Stage.
  - This hierarchy allows the definition and packaging of services, and separate them in executions steps, stages.
  - This implementation is now as simple as to annotate a java class with: @ProviderDefinition, @ApplicationDefinition, @ServiceDefinition, @StageDefinition
- The DEM is available programmatically through the API
- Possible automations
  - UDDI e WSDL: We believe these will be generated directly from the DEM
  - WSRP/JSR-168: Available portlets and how to invoke them

```
@StageDefinition(name = "Sample Service", service="sampleservice")
@View(target="sample.jsp")
public class SampleStage {

   @Execute
   public void myMethod() {
      // Do something...
   }
}
```

DiF's features?

Dynamic Code Generation

- DiF generates code for conventioned on configurable features
  - Annotation driven. Dozens of easy to use annotations are available
  - Abstracts the developer from the majority of the DiF's internal API
  - Adapts to the developer code and not the other way around
  - Shields the developer code from future API changes/upgrades
  - Enables existing client code to be enriched with new functionalities, performance improvements or new best practices, after the code creation.
  - All implementation follow the model: "convention vs configuration vs programming", that is, conventioned behavior is by default, we can adjust them with configurations and as a last resort we can programmatically change them.

DiF's glue! Allows to connect all features and implement the majority of it's concepts.

DiF's features?

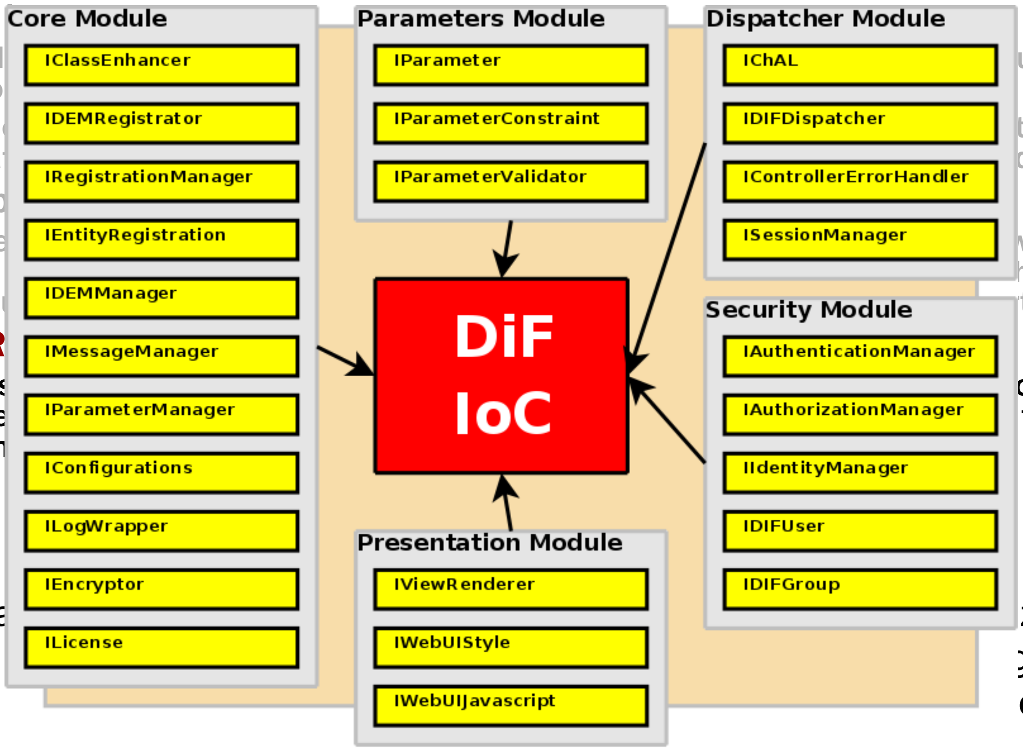
IoC – Inversion of Control container

- Integration layer
  - Central piece of integration for DiF. In itself DiF favors SoC integrating all it's modules through IoC, making it easy to integrate it with other technologies. Very similar to Spring.
  - New modules will be available in the future that will enable DiF to be extended with functionalities and integrations with several technologies/products. Anyone can do this is a simplified way.
- Pluggable platform
  - In DiF everything is a module/plug-in. We can replace existing plug-ins with our own (i.e.: our own authentication engine that manages and authenticates users in a proprietary database). The architecture enables collection of contributions to, for instance, build a list of existent parameter validators of export formats to the developers.

DiF itself is a collection of plug-ins. This makes it easy to replace a default plug-in with a new one to integrate a given external platform. The same concept applies to user applications that decide to follow this philosophy.

DiF's features?

IoC – Inversion of Control container



- Integration
  - Central ules through IoC, making it easy to
  - New m th functionalities and integra d way.
- Pluggable p
  - In DiF e wn (i.e.: our own authentication engine hitecture enables collection of contribu t formats to the developers.
- **Drop-in JAR**
  - **There is ckaged in a JAR file with a module the App. Server the IoC picks it up an**

To insta ze and ones. edited!

DiF's features?

AOP – Aspect Oriented Programming

- AOP Integration
  - DiF includes the integration of AspectJ as an AOP API for cross-cutting concerns
- Existing implementations
  - Logging.
  - Exceptions

DiF's features?

Model ORM Generator

- Native Hibernate integration
    - DiF includes Hibernate integration and code generation for the Model java classes (through the Maven ORM plug-in)

Hibernate is the most widely used ORM API. It offers:
•BD abstraction (virtually supports any DB that has a JDBC driver)
•Cache management
•Query optimizer
•Relations/Constraints management, PKs management, etc.
•Transactional management, etc.

DiF’s features?

Model ORM Generator

- Native Hibernate integration
  - DiF includes Hibernate integration and code generation for the Model java classes (through the Maven ORM plug-in)
- ORM Model generation by reverse engineering (reveng) of the Database Entity Model
  - Includes the following layers: Data objects, Mapping files, DAOs, Services
  - The generated classes can be extended, not being overwritten in future “reveng” processes.
- IoC integration
  - The ORM generated is binded by auto generated IoC contributions in the “reveng” process.

```
@StageDefinition(name = "Sample Service", service="sampleservice")
@View(target="sample.jsp")
public class SampleStage {

    @Inject
    protected IService cseService;

    @Execute
    public void myMethod() {
        int total = cseService.getCursosDAO().findById(1000).getAlunos().size();
    }
}
```

DiF's features?

Model ORM Generator

- Native Hibernate integration
  - DiF includes Hibernate integration and code generation for the Model java classes (through the Maven ORM plug-in)
- ORM Model generation by reverse engineering (reveng) of the Database Entity Model
  - Includes the following layers: Data objects, Mapping files, DAOs, Services
  - The generated classes can be extended, not being overwritten in future "reveng" processes.
- IoC integration
  - The ORM generated is binded by auto generated IoC contributions in the "reveng" process.
- Customization of the whole process
  - Definition of desired service structured in an XML file
  - Freemarker templates for all generated files
  - Several configuration possibilities provided by the Hibernate Tools Reverse Engineering process

DiF’s features?

Security

- Different needs in different modules
  - IdentityManager: Users, Grups, their relationships and attributes
  - AuthenticationManager: Authentication management
  - AuthorizationManager: Access control management for users and groups to services

Separation of concerns: Identity (who are you?), authentication (are you really who you say you are?) e authorization (what can you do/access?). Facilitates integrations.

DiF’s features?

Security

- Different needs in different modules
  - IdentityManager: Users, Grups, their relationships and attributes
  - AuthenticationManager: Authentication management
  - AuthorizationManager: Access control management for users and groups to services
- Several flavors possible
  - Actual: Static, LDAP, Database
  - Future: More LDAPs, Kerberos, others
- External contributions
  - Any developer can easily contribute with specific implementations of any of these modules. Be it for integrating a currently unsupported standard of to integrate with a proprietary system.
- Default environment
  - Automatic creation of default users or groups in any integrated systems

DiF’s features?

Parameter management

- Parameter scope
  - Request: Request parameters that are discarded after the current request is served
  - Static: static parameters, shared by all instances of the same entity
  - Session: Parameters that are kept thought the current session
  - User: User parameters. Are shared even through all session of this same user

All web pages interact through the submission of parameters.
A necessary and recurrent task.

I.e.: http://www.server.com?**num=10**&**date=11-11-2010**&**active=true**

DiF’s features?

Parameter management

- Parameter scope
  - Request: Request parameters that are discarded after the current request is served
  - Static: static parameters, shared by all instances of the same entity
  - Session: Parameters that are kept thought the current session
  - User: User parameters. Are shared even through all session of this same user
- Advantages of DiF’s parameter management
  - Assignment of class attributes
  - Automatic conversion to Java types
  - Default values
  - Flexible and extendible parameter validation system, based on constraints and validators
  - Automatic scope management
  - Parameter inheritance through the DEM
  - Parameter validation reflected in the View through specific UI components
  - Persistence
  - All this with an annotation in a class attribute

Automation is the word of the day.
Ask for a behavior, don’t implement it!

DiF's features?

Parameter

- Parameter
  - Reque
  - Static:
  - Sessio
  - User: U
- Advantage
  - Assign
  - Autom
  - Defau
  - Flexib
  - Autom
  - Param
  - Param
  - Persis
  - All this

s

```
@StageDefinition(name = "Sample Service", service="sampleservice")
@View(target="sample.jsp")
public class SampleStage {

    @Parameter
    String someStringParameter;

    @Parameter(defaultValue="20", constraints="required,max=100,min=10")
    @Persist(scope=ParameterScope.SESSION)
    Long someLongParameter;

    @Parameter(defaultValue="true")
    @Persist(scope=ParameterScope.USER)
    Boolean someBooleanParameter;

    @Parameter(constraints="required")
    @Persist(scope=ParameterScope.SESSION)
    Date someDateParameter;

    @Execute
    public void myMethod() {
        // Do something...
    }
}
```

Automation is the word of the day.
Ask for a behavior, don't implement it!

DiF's features?

Messages

- Simple message definition
  - Defined in simple ".properties" files. One for each entity.
  - DEM inherited messages
  - Reusable message packages
  - Automatic internationalization (i.e. SomeStage.messages.en)

DiF's features?

Messages

- Simple message definition
  - Defined in simple ".properties" files. One for each entity.
  - DEM inherited messages
  - Reusable message packages
  - Automatic internationalization (i.e. SomeStage.messages.en)
- Features & Automations
  - It will be possible to custmize messages in a deployment envirionament
  - These customizations will be kept in an automatically managed external repository that, through IoC, can be any format desired. (default will be a database).

DiF’s features?

Messages

- Simple message definition
  - Defined in simple “.properties” files. One for each entity.
  - DEM inherited messages
  - Reusable message packages
  - Automatic internationalization (i.e. SomeStage.messages.en)
- Features & Automations
  - It will be possible to custmize messages in a deployment envirionament
  - These customizations will be kept in an automatically managed external repository that, through IoC, can be any format desired. (default will be a database).
- Gestão & Performance
  - Memory management
  - Lazy loading

DiF’s features?

Configurations

- Configuration repository
  - No more endless XML/properties files in several directories, or database backends
  - DiF’s managed parameters are kept in Registry (in windows) or a structured file system structure (in Linux). All is accessible since the format is a hierarchical set of XML files
  - Any new implementations of this API can use a different repository
- How will a user edit these configurations?
  - A specific interface will be made available by default in DiF for this purpose.
- How will the developer use it?
  - Simple POJOs, or annotated ones for non-default configurations

Complexity reduction for developers, centralizations for administrators and more easiness and data organization for both.

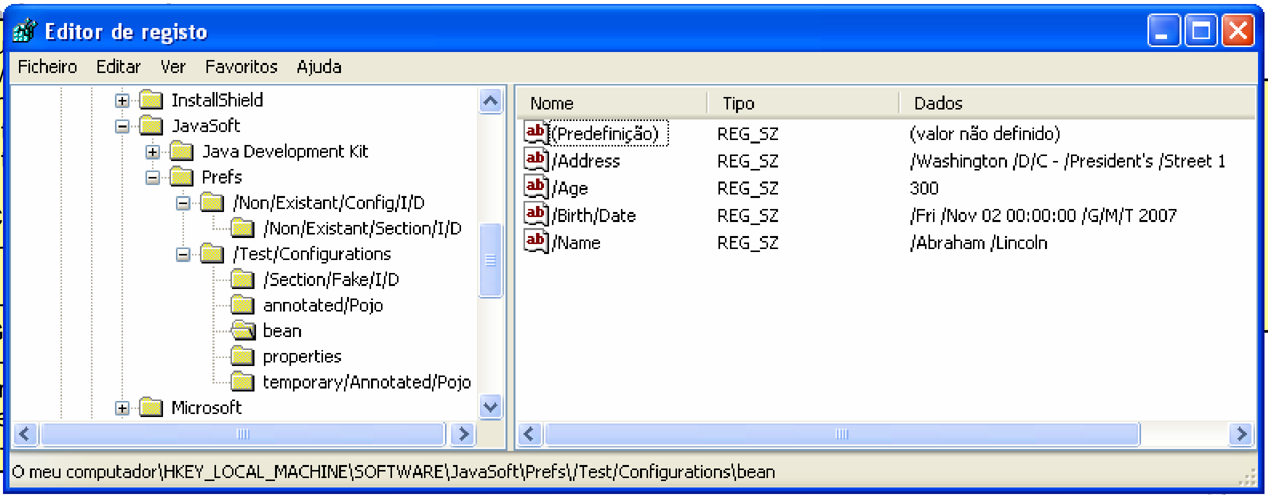
DiF’s features?

Configurations

- Configura
- H
- H

```
root@linu
/etc/.java/
`-- .syster
    |-- .sys
    |-- .sys
    `-- dif2
        |-- C
        |   |--
        |   |
        |   `--
        |-- G
        |   `--
        |-- pr
        `-- re
            `--
```

All



```
InstallShield
JavaSoft
    Java Development Kit
    Prefs
        /Non/Existant/Config/I/D
            /Non/Existant/Section/I/D
        /Test/Configurations
            /Section/Fake/I/D
            annotated/Pojo
            bean
            properties
            temporary/Annotated/Pojo
Microsoft
```

Nome	Tipo	Dados
(Predefinição)	REG_SZ	(valor não definido)
/Address	REG_SZ	/Washington /D/C - /President's /Street 1
/Age	REG_SZ	300
/Birth/Date	REG_SZ	/Fri /Nov 02 00:00:00 /G/M/T 2007
/Name	REG_SZ	/Abraham /Lincoln

O meu computador\HKEY_LOCAL_MACHINE\SOFTWARE\JavaSoft\Prefs\/Test/Configurations\bean

Complexity reduction for developers, centralizations for administrators and more easiness and data organization for both.

DiF’s features?

Web 2.0: Components & Component Library

- Component driven
  - Tags are components
  - Views (Pages) are components

Component driven development natively supported by the framework. In components (Tags) and stages (Pages)

DiF’s features?

Web 2.0: Components & Component Library

- Component driven
  - Tags are components
  - Views (Pages) are components
- Component Library
  - Extensive UI component library
  - JavaScript library integrated
  - Accessible/Rich UI versions selectable by the user
  - Dynamic component that generate rich interfaces with little developer effort
    - Grid
    - BeanEditForm

Component driven development natively supported by the framework. In components (Tags) and stages (Pages)

DiF's features?

Maven: Project Automation

- Maven, what is it?
  - O Maven is a software project technical management tool
  - Created by Jakarta to standardize and integrate all it's projects. Just on it's central repository Maven has near 25.000 libraries integrated

Maven: Create project, access resources, package, deploy, test, install, document, distribute, etc!

DiF's features?

Maven: Project Automation

- Maven, what is it?
  - O Maven is a software project technical management tool
  - Created by Jakarta to standardize and integrate all it's projects. Just on it's central repository Maven has near 25.000 libraries integrated

I.e..: `user@linux:~$ mvn archetype:create…`

- Development stages integration
  - All process is managed by Maven, namely:
    - New project creation based on a template

DiF’s features?

Maven: Project Automation

- Maven, what is it?
  - O Maven is a software project technical management tool
  - Created by Jakarta to standardize and integrate all it’s projects. Just on it’s central repository Maven has near 25.000 libraries integra

I.e..: `user@linux:~$ mvn eclipse:eclipse…`

- Development stages integration
  - All process is managed by Maven, namely:
    - New project creation based on a template

DiF's features?

Maven: Project Automation

- Maven, what is it?
  - O Maven is a software project technical management tool
  - Created by Jakarta to standardize and integrate all it's projects. Just on it's central repository Maven has near 25.000 libraries integrated

Automatic execution every time we generate the packages. These condition the package generation on successful testing.

- Development stages integration
  - All process is managed by Maven, namely:
    - New project creation based on a template
    - Access to the framework package with all it's

DiF's features?

Maven: Project Automation

- Maven, what is it?
  - O Maven is a software project technical management tool
  - Created by Jakarta to standardize and integrate all it's projects. Just on it's central repository Maven has near 25.000 libraries integrated

I.e..: `user@linux:~$ mvn site site:deploy…`

- Development stages integration
  - All process is managed by Maven, namely:
    - New project creation based on a template
    - Access to the framework package with all it's

DiF’s features?

Maven: Project Automation

- Maven, what is it?
  - O Maven is a software project technical management tool
  - Created by Jakarta to standardize and integrate all it’s projects. Just on it’s central repository Maven has near 25.000 libraries integrated

Reports are included in the site. We can create automation that prevent some steps from being executed if the quality in bellow a given threshold.

- Development stages integration
  - All process is managed by Maven, namely:
    - New project creation based on a template
    - Access to the framework package with all it’s

DiF’s features?

Maven: Project Automation

- Maven, what is it?
  - O Maven is a software project technical management tool
  - Created by Jakarta to standardize and integrate all it’s projects. Just on it’s central repository Maven has near 25.000 libraries integrated

I.e..: `user@linux:~$ mvn cargo:start…`

- Development stages integration
  - All process is managed by Maven, namely:
    - New project creation based on a template
    - Access to the framework package with all it’s

DiF's features?

Maven: Project Automation

- Maven, what is it?
  - O Maven is a software project technical management tool
  - Created by Jakarta to standardize and integrate all it's projects. Just on it's central repository Maven has near 25.000 libraries integrated

I.e..: `user@linux:~$ mvn package deploy release…`

- Development stages integration
  - All process is managed by Maven, namely:
    - New project creation based on a template
    - Access to the framework package with all it's

Built on top of Best-of-breed

Base DiF2 technologies. Integration and usage.

JEE 5.0	Core
Servlet API	View Renderer
Tag libraries	View Renderer
AspectJ	AOP
Guice (Google)	IoC
Javassist (JBoss)	Bytecode enhancement
JUnit, HttpUnit, easymock…	Testing
Hibernate (JBoss)	ORM (PostgreSQL, mySQL, Oracle, SQL Server, DB2, etc…)
Maven (Apache)	Project Management
Ext JS	JavaScript library
JNDI	LDAP

Satellite Projects

Projects that were driven/created for DiF's common needs

- IoC Utils
  - IoC container implementation. Abstracts and extends the internal IoC's functionalities
  - Presently provides a Google Guice based implementation
- Configuration Utils
  - Implementation of an API to manage all configuration need based on POJOs
  - Presently provides an implementation based on the Preferences API from Sun
- Bytecode Utils
  - Implementation of an API for bytecode enhancement. Simplifies the task
  - Presently provides an implementation based on Javassist from JBoss
- LDAP Utils
  - Implementations of an API to manage an LDAP server.
  - Extends the information possible to store on a LDAP server.
  - Normalizes the access to non-standard features of the supported existing LDAP servers
  - Based on JNDI
  - Presently supports : Active Directory (AD).
  - In the future: Oracle Internet Directory (OID) and OpenLDAP.

Satellite Projects

Projects that were driven/created for DiF's common needs

- Logger
  - Implementation of an API for logging with extended functionalities
  - Presently provides an implementation based on Log4J from Apache
- ISS
  - Asynchronous/synchronous process management engine
  - Automates the management os the server's resources to achieve the maximum load for the running process, without degrading the server's response
- Maven plug-ins

Archetypes	New DiF project templates
Packager	Automated packager of source code, libraries and executable libraries
ORM Generator	Reverse engineering process for a ORM Model from a database entity model

Tools

Productivity tools

- Maven
- ORM Generator
- DiF IDE
    - Based on Eclipse
    - Bundled with a set of plug-ins selected to facilitate the developer that uses DiF.

Resources

Resources available for DiF adopters

- Product sites
  - http://development.digitalis.pt
- Mailing list
  - http://development.digitalis.pt/apache2-default/dif2/dif-core/mail-lists.html
- Issue tracking
  - http://development.digitalis.pt/apache2-default/dif2/dif-core/issue-tracking.html
- Wiki
  - http://development.digitalis.pt/mediawiki/
- Proposed commercial services from Digitalis
  - DiF startup-kit – In progress
  - User Guide (book in digital format) – In progress
  - Development on DiF Course
  - Consulting on DiF development
  - Out-sourcing of DiF developers
  - Support for DiF developers

Who’s using DiF

DiF Adopters

- Digitalis Informática
- Grupo Lusófona – Largest non-public University in Portugal
- FCUL – The Faculty of Sciences of the University of Lisbon, Portugal
- Retail company (Prospect, to be confir
- Open-source project (Prospect, to be confirmed)

What’s next?

Future developments.

- WebServices as a Channel
- Cache, WebCache
- Native AJAX
- View Renderers: PDF, Freemarker
- Portlets (JSR-168, WSRP) as a Channel
- Administration services for all DiF plug-ins
- Workflow (BPEL probably)
- Reporting framework
- Service and data auditing (integrated with ORM generation)
- DiF IDE (Wizards e productivity editors) – Maximize RAD

Q&A

Thank you

Development Department

http://development.digitalis.pt

Digitalis Informática ©2008

http://www.digitalis.pt