NEOShield-2

Science and Technology for Near-Earth Object Impact Prevention

Grant agreement no:	640351	Project Start:	1 March 2015
Project Coordinator	Airbus Defence and Space DE	Project Duration:	31 Months

WP 11.3 **Deliverable D11.4**	**NEO Physical Properties Database User Manual**		
WP Leader	DLR	Task Leader	DMS
Due date	M4, 31.07.2015		
Delivery date	26.10.2015		
Issue	1.2		
Editor (authors)	Esther Parrilla-Endrino		
Contributors	Mª del Mar Nuñez-Campos		
Verified by	Ettore Perozzi		
Document Type	R		
Dissemination Level	PU		

The NEOShield-2 Consortium consists of:		
Airbus DS GmbH (Project Coordinator)	ADS-DE	Germany
Deutsches Zentrum für Luft- und Raumfahrt e.V.	DLR	Germany
Airbus Defence and Space SAS	ADS-FR	France
Airbus Defence and Space Ltd	ADS-UK	United Kingdom
Centre National de la Recherche Scientifique	CNRS	France
DEIMOS Space Sociedad Limitada Unipersonal	DMS	Spain
Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.	EMI	Germany
GMV Aerospace and Defence SA Unipersonal	GMV	Spain
Istituto Nazionale di Astrofisica	INAF	Italy
Observatoire de Paris	OBSPM	France
The Queen's University of Belfast	QUB	United Kingdom

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 640351.

HORIZON 2020

Change Record

Issue	Date	Section, Page	Description of Change
1.0	31.08.2015		Initial draft issue
1.1	01.10.2015		First stable version that includes all operations description and the full installation procedure. Also implemented comments provided by Airbus dated 27th July 2015.
1.2	26.10.2015	All	Implementation of comments into various sections, which are raised by consortium and during a dedicated telecon.

Table of Contents

1 Introduction

1.1 Scope

The scope of this document is to provide a complete view on the usage of the NEOShield-2 NEO Physical Properties Database, describing how to install, configure and operate the system. The manual is structured as follows:

- Section 2 provides an overview of the project scope.
- Section 3 describes the top-level architecture and the set of components of the system.
- Section 4 lists the third-party technologies used in the implementation of the system.
- Appendix A details the steps to install, deploy and test the system.

1.2 List of Abbreviations

AD	Applicable Document
API	Application Program Interface
DAO	Data Access Object
DMZ	Demilitarized Zone
EARN	European Asteroid Research Node
ESA	European Space Agency
GA	Grant Agreement
HMI	Human Machine Interface
HTTP	Hypertext Transfer Protocol
IPR	Intellectual Property Rights
MPC	Minor Planet Center
MVP	Model View Presenter
NEA	Near Earth Asteroid
NEO	Near Earth Object
NEOCC	NEO Coordination Center
NEODyS	Near-Earth Objects Dynamic Site
PL	Priority List
P3L	Physical Properties Priority List
RD	Reference Document
SCN	Spaceguard Central Node
SLA	Service Level Agreement
SOAP	Simple Object Access Protocol
SSA	Space Situational Awareness
SW	Software
TBC	To Be Completed
UI	User Interface
VM	Virtual Machine

WP	Work Package
XML	Extensible Markup Language

1.3 Applicable Documents

[AD1] NEOShield-2: “Science and Technology for Near-Earth Object Impact Prevention”, Grant Agreement no. 640351, 28.10.2014.

1.4 Reference Documents

[RD1] Perozzi E., Bassano E., Gloria M., Pagano F., Reboa L., Milani A., Bernardi F., Farnocchia D., Valsecchi G.B., D'Abramo G., Franco R., Drolshagen G., Koschny D., 2011. *Designing the Space Situational Awareness NEO Segment.* In proc. 2nd IAA Planetary Defense Conference.

[RD2] NEODyS: Near-Earth Objects Dynamic Site: http://newton.dm.unipi.it/neodys/

[RD3] European Asteroid Research Node (EARN): http://earn.dlr.de/

[RD4] The Spaceguard System: http://spaceguard.iasf-roma.inaf.it/SSystem/SSystem.html

[RD5] Minor Planet Centre: *the nerve centre of asteroid detection in the Solar System.* http://www.minorplanetcenter.net/iau/mpc.html

[RD6] ESA NEO Coordination Centre http://neo.ssa.esa.int

[RD7] Perozzi E.: *Observations Support Tools User Manual for NEOShield-2,* draft version from the 05th of May, 2015.

[RD8] NEOShield-2 D9.1, NEOShield-2 Dynamical Web Interface User Manual, v1.1 from the 1st of October 2015.

1.5 Standards

[STD1] Oracle Java Code Conventions (http://www.oracle.com/technetwork/java/codeconvtoc-136057.html), revised the 20th of April, 1999.

[STD2] Web Service Description Language, v2.0 (http://www.w3.org/ TR/ wsdl20/).

2 System Overview

2.1 Project background

Impacts of near-Earth objects (NEOs) have contributed to mass extinctions and evolution, and it is a proven fact that NEOs will continue to hit the Earth at irregular intervals in the future, with the potential for catastrophic damage to life and property.

In the context of the NEOShield-2 project, astronomical observations of NEOs will be carried out to improve our understanding of their physical properties, concentrating on the smaller sizes of most concern for mitigation purposes, and to identify further objects suitable for missions for physical characterisation, and NEO deflection demonstration. The aim of NEOShield-2 is to expand and enhance the space-mission NEO target database to include targets of interest for NEO exploratory missions in addition to mitigation demonstration missions. The physical properties and measurement uncertainties, or quality flags, relevant to reconnaissance, sample-return and mitigation demonstration missions will be included.

The specialized potential-target database will initially be developed separately, and then connected to the NEOShield-2 NEO physical properties open data repository thus creating a dynamical web interface to allow the public user to search on NEO physical properties relevant to exploration or mitigation demonstration missions. Appropriate settings of criteria will enable required prioritized lists of potential NEO mission targets to be obtained.

This will eventually allow the onset of a “two-way” operational interface with the ESA NEO Coordination Centre (see section 2.2.1) for focusing physical observations to potential mission targets and for providing the information needed to properly plan and execute them successfully (e.g. assign priorities based on accessibility considerations, send/receive astrometric alerts, provide astrometry of challenging objects, prompt newly discovered object observation opportunities, answer to requests from the observers, etc.).

The NEOShield-2 NEO physical properties database and the dynamical web interface will be publicly accessible through the NEOShield-2 NEO Properties Portal, as shown in the context diagram of Figure 1. In this diagram the solid arrows represent the flux of data either generated or disseminated through the NEO properties portal, whereas the dotted lines represent the links with functions developed outside WP 11.3 and WP 9.1 or accessible outside the NEOShield-2 project (e.g. ESA NEOCC, see section 2.2.1).

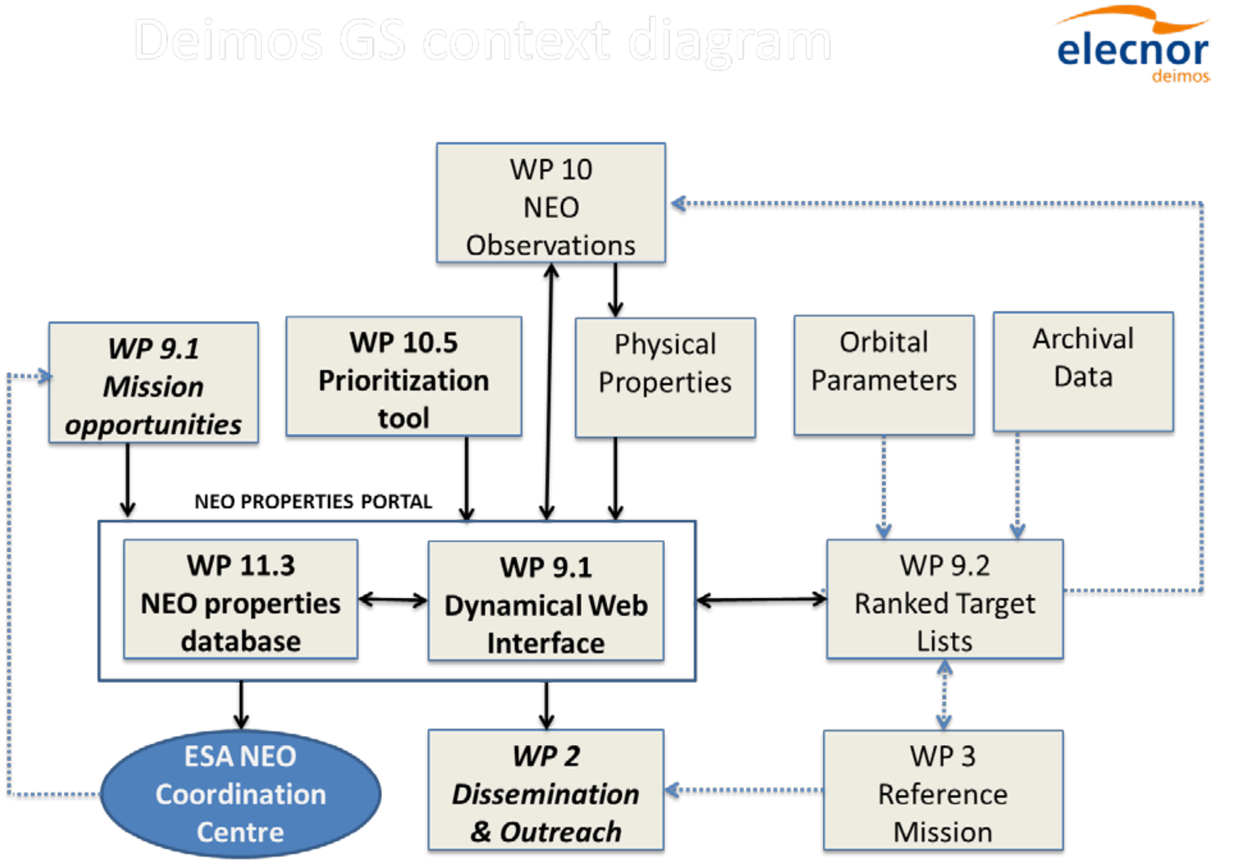

Deimos GS context diagram
elecnor deimos
WP 10 NEO Observations
WP 9.1 Mission opportunities
WP 10.5 Prioritization tool
Physical Properties
Orbital Parameters
Archival Data
NEO PROPERTIES PORTAL
WP 11.3 NEO properties database
WP 9.1 Dynamical Web Interface
WP 9.2 Ranked Target Lists
ESA NEO Coordination Centre
WP 2 Dissemination & Outreach
WP 3 Reference Mission

Figure 1: NEOShield-2 NEO properties database and dynamical web interface context diagram The public availability of the NEOCC physical properties database allows to avoid developing redundant functionalities within the NEOShield-2 project. The data produced by the NEOShield-2 observational activity, once published on the NEOShield-2 portal, will be integrated into the NEOCC database through the regular updating procedures provided to ESA by DLR under SLA.

2.2 NEO Physical Properties Database and Dynamical Web Interface

The aim of the NEOShield-2 NEO Physical Properties Database and Dynamical Web Interface implementation is to coordinate observations, ease the identification of suitable mitigation or exploration mission targets and to be able to compute and visualize example mission scenarios under different assumptions, allowing the retrieval and display of data useful for:

- Planning observations
- Performing off-line scientific analysis
- Update physical properties
- Summarize mission profiles

One of the main targets of the new Dynamical Web Interface is to contribute to increase the fraction of NEO population with known physical characteristics via dedicated observation campaign as well as, consolidating the NEO physical properties and their accessibility via tables and plots.

In order to do so the following data will be used:

- the orbital data of the whole NEO population as provided by the NEOCC continuously updated catalogue;
- the ephemeris of the individual objects as described in [RD7];

- the known NEO physical properties available from the NEOCC Physical Properties Database.

As discussed in section 2.1 in order to avoid redundancies the NEOShield-2 NEO Physical Properties Database (see Fig. 1) shall include only objects observed during NEOShield-2 campaigns. For each of them it will provide the data products at different processing levels (raw, intermediate and final) shall be stored.

The basic structure of a NEOShield-2 Dynamical Web Interface consists of a web page containing updated tables and plots of observed objects and potential targets from where shall be possible to locate and download related datafiles, the following data shall be available:

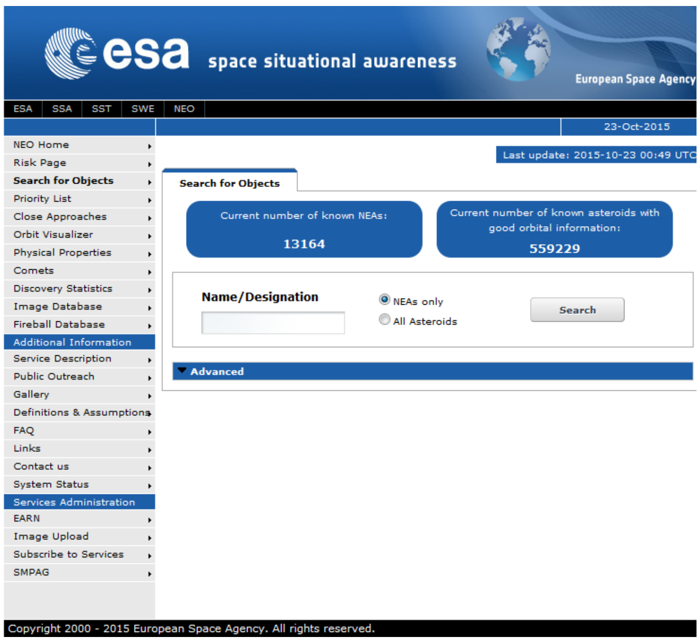
- For each object observed by NEOShield-2:
    - Images (spectra, light curves).
    - Data files (in two-column format: e.g. photometry vs time).
    - Final products (e.g. size, spectral type etc).
- The information produced by SW tools:
    - Mission scenarios (table).
    - Observational support priorities (table).
    - Accessibility diagrams (plot).
- Observational support priorities:
    - For each NEO computes visibility-related quantities.
    - Generates a table of observable objects prioritized in terms of importance and urgency.
    - The priority table will be run daily through an automatic procedure and displayed on the Dynamic Web Interface.
    - The priority table can be run off-line for an arbitrary date.
- Accessibility diagrams:
    - For each NEO computes the delta-V corresponding to a best case Hohmann-like transfer trajectory.
    - Generates an accessibility diagram (H-plot) of the NEO population.
    - The plot can be customized by highlighting subpopulations (e.g. objects belonging to a certain spectral type, objects observed by NEOShield-2 etc.).

2.2.1 The ESA NEO Coordination Centre Physical Properties database

When the whole NEO population is involved the Physical Properties database publicly available through the ESA SSA-NEO Coordination Centre web portal (see [RD6]) should be addressed.

Developed within the framework of the Space Situational Awareness programme of the European Space Agency, the NEOCC web portal (neo.ssa.esa.int) hosts the EARN data (European Asteroids Research Node, earn.dlr.de/) thus representing a primary source of NEO physical properties. EARN contains up-to-date published data on all know NEAs and the corresponding bibliographic references; it is maintained by DLR Institute of Planetary Research in Berlin and it is considered the most detailed open data repository for physical and dynamical properties of NEOs.

EARN data have been integrated into the NEOCC system as a fully searchable database: a single query interface allows to display both the dynamical and the physical properties of any given NEOs or to search for objects within certain parameters range for further investigation. The physical properties data can be browsed from the NEOCC web portal by selecting the “search for objects” main menu option. Once the corresponding page is displayed (Figure 2) one can either enter an object name and being led to the corresponding summary page (Figure 3) or perform an “advanced search” among the dynamical and physical data stored in the NEOCC database (Figure 4).



Last update: 2015-10-23 00:49 UTC

Search for Objects

Current number of known NEAs: 13164

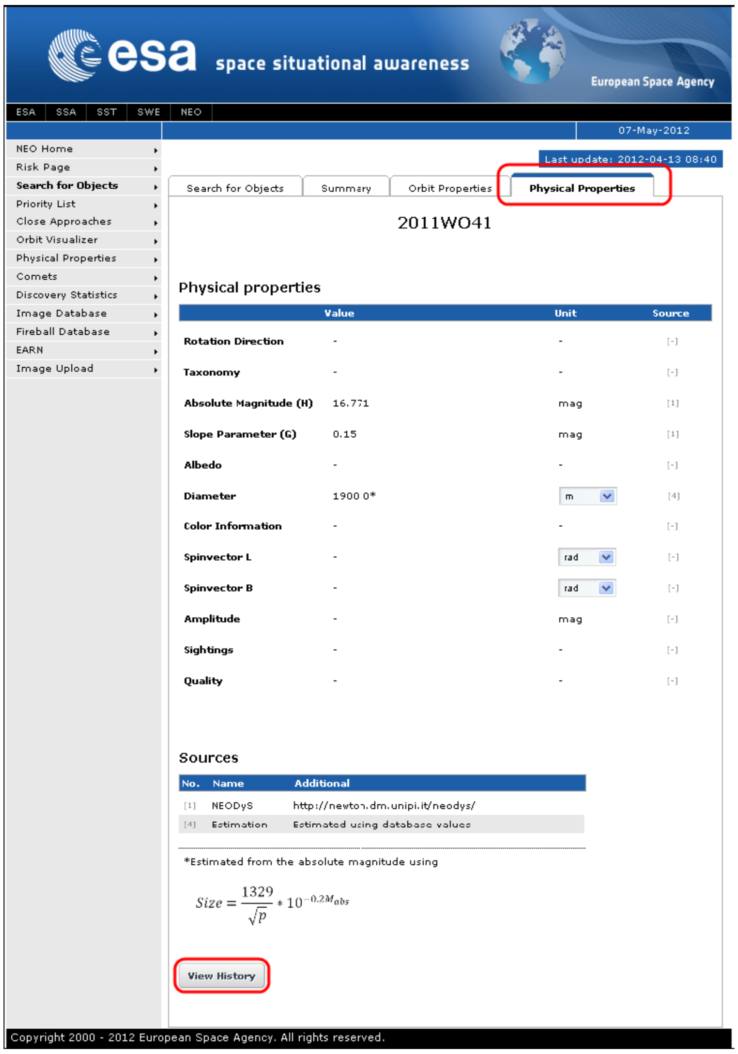
Current number of known asteroids with good orbital information: 559229

Name/Designation

[✓ NEAs only] All Asteroids

Copyright 2000 - 2015 European Space Agency. All rights reserved.

Figure 2: The "search for objects" page of the NEOCC web site



Last update: 2012-04-13 08:40

2011WO41

Physical properties

	Value	Unit	Source
Rotation Direction	-	-	[-]
Taxonomy	-	-	[-]
Absolute Magnitude (H)	16.771	mag	[1]
Slope Parameter (G)	0.15	mag	[1]
Albedo	-	-	[-]
Diameter	1900 0*	m	[4]
Color Information	-	-	[-]
Spinvector L	-	rad	[-]
Spinvector B	-	rad	[-]
Amplitude	-	mag	[-]
Sightings	-	-	[-]
Quality	-	-	[-]

Sources

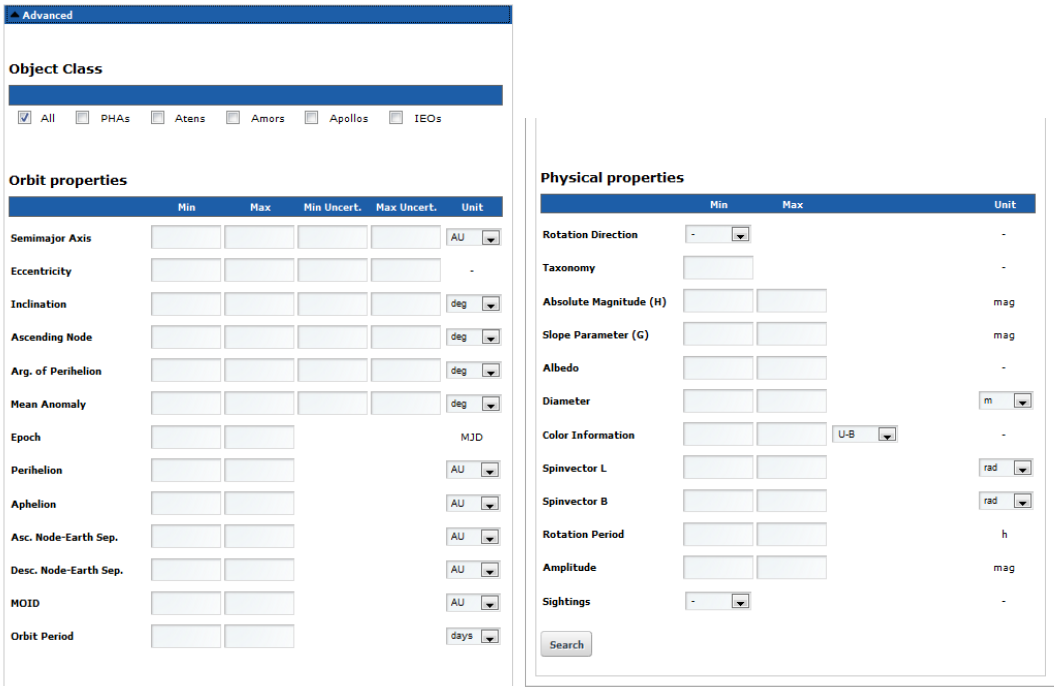
No.	Name	Additional
[1]	NEODyS	http://newton.dm.unipi.it/neodys/
[4]	Estimation	Estimated using database values

*Estimated from the absolute magnitude using

$$Size = \frac{1329}{\sqrt{p}} * 10^{-0.2M_{abs}}$$

Copyright 2000 - 2012 European Space Agency. All rights reserved.

Figure 3: The physical properties summary page of the NEOCC web site



▲ Advanced

Object Class

[✓ All] ☐ PHAs ☐ Atens ☐ Amors ☐ Apollos ☐ IEOs

Orbit properties

	Min	Max	Min Uncert.	Max Uncert.	Unit
Semimajor Axis					AU
Eccentricity					-
Inclination					deg
Ascending Node					deg
Arg. of Perihelion					deg
Mean Anomaly					deg
Epoch					MJD
Perihelion					AU
Aphelion					AU
Asc. Node-Earth Sep.					AU
Desc. Node-Earth Sep.					AU
MOID					AU
Orbit Period					days

Physical properties

	Min	Max		Unit
Rotation Direction	-			-
Taxonomy				-
Absolute Magnitude (H)				mag
Slope Parameter (G)				mag
Albedo				-
Diameter				m
Color Information			U-B	-
Spinvector L				rad
Spinvector B				rad
Rotation Period				h
Amplitude				mag
Sightings	-			-

Search

Figure 4: The advanced search option of the NEOCC web site

The possibility of ingesting additional data provided by the NEOShield-2 project and of providing a direct access to the NEOCC search engine from the NEOShield-2 Properties Portal will be proposed as part of the NEOCC evolutionary maintenance.

3 Overall Architecture

The objective of the NEO Physical Properties Database and Dynamical Web Interface architecture is the definition of a modular and scalable system composed by independent components that interact between each other using well-defined interfaces.

The Dynamical Web Interface will be publicly available, providing the necessary tools for searching and visualizing NEOShield-2 data. Thus in what follows the users of the NEO Properties Portal will be referred to as NEOShield-2 users.The only requirements for running the tool on any location are:

- To have installed a web browser.
- To have access to the Internet.

The Figure 3 below shows the context diagram of the NEO Physical Properties Database and Dynamical Web Interface, the default flow of events would be the following:

1. A NEOShield-2 user launches a web browser to connect remotely through the Internet to the Dynamical Web Interface home page.
2. The Dynamical Web Interface is installed at DEIMOS premises infrastructure outside and accessible from the Internet.
3. Once the Dynamical Web Interface is launched, the user can perform a set of operations like the NEO search, visualization of tabular data, plotting display etc…
4. When accessing the Dynamical Web Interface it interfaces with its backend infrastructure located within the DMZ. All requests between the frontend and backend parts of the system use the standard HTTP over SOAP protocol that ensures a well-defined communication based on XML messages.

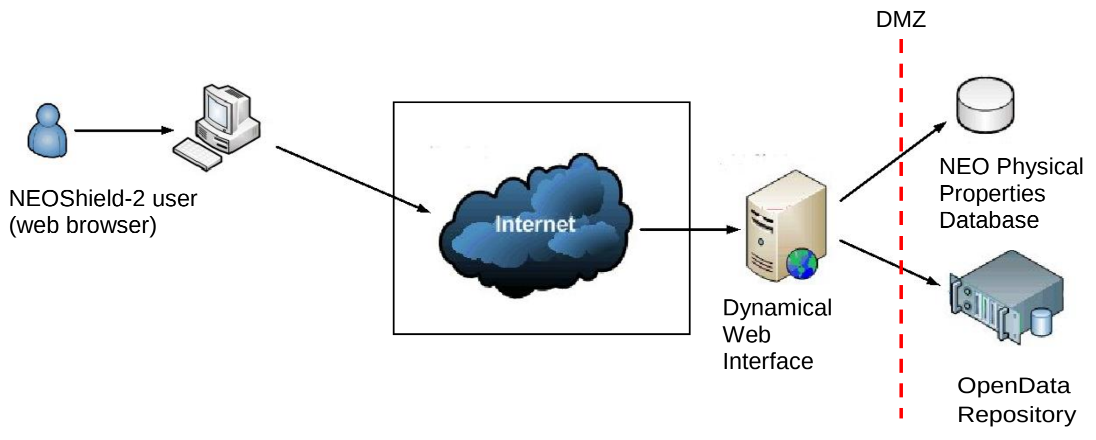


Figure 5: NEO Physical Properties Database and Dynamical Web Interface context diagram

From the design point of view, we can state the following high-level separation of capabilities:

- ❑ Logic related to the frontend presentation layer of the tool such as rendering of NEO data, tabular display, plotting etc....
- ❑ Logic related to the backend business logic layer of the tool that will interface with the OpenData Repository and the NEO Physical Properties Database.

Therefore a Three-Tier architecture shown in Figure 4 below has been selected as the best approach for the development of the Dynamical Web Interface:

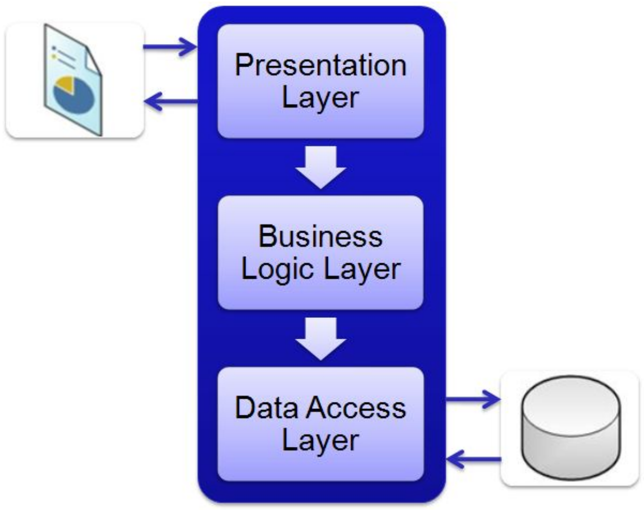


Figure 6: Dynamical Web Interface Three-Tier design

- The **Presentation** layer contains all client-side modules related with the query building forms and the results display of NEO data and related metadata in tabular or plot format.
- The **Business Logic** layer is the most important part in the whole architecture because it defines the real-world business rules that determine how data can be created, displayed, stored, and changed. It works as a bridge between the Presentation and Data Access layers, all the user values received from the input web forms are being passed to it and in turn, the results received from the storage infrastructure in row data format are returned as suitable objects that can be easily displayed to the user.
- The **Data Access** layer builds the necessary queries based on the received parameters from the Business Logic Layer and executes them over the NEO Physical Properties Database and the OpenData Repository, returning the results back.

The following diagram in Figure 5 displays the general structure of the components necessary to deliver data among the system; the dataflow among components is represented using arrows:

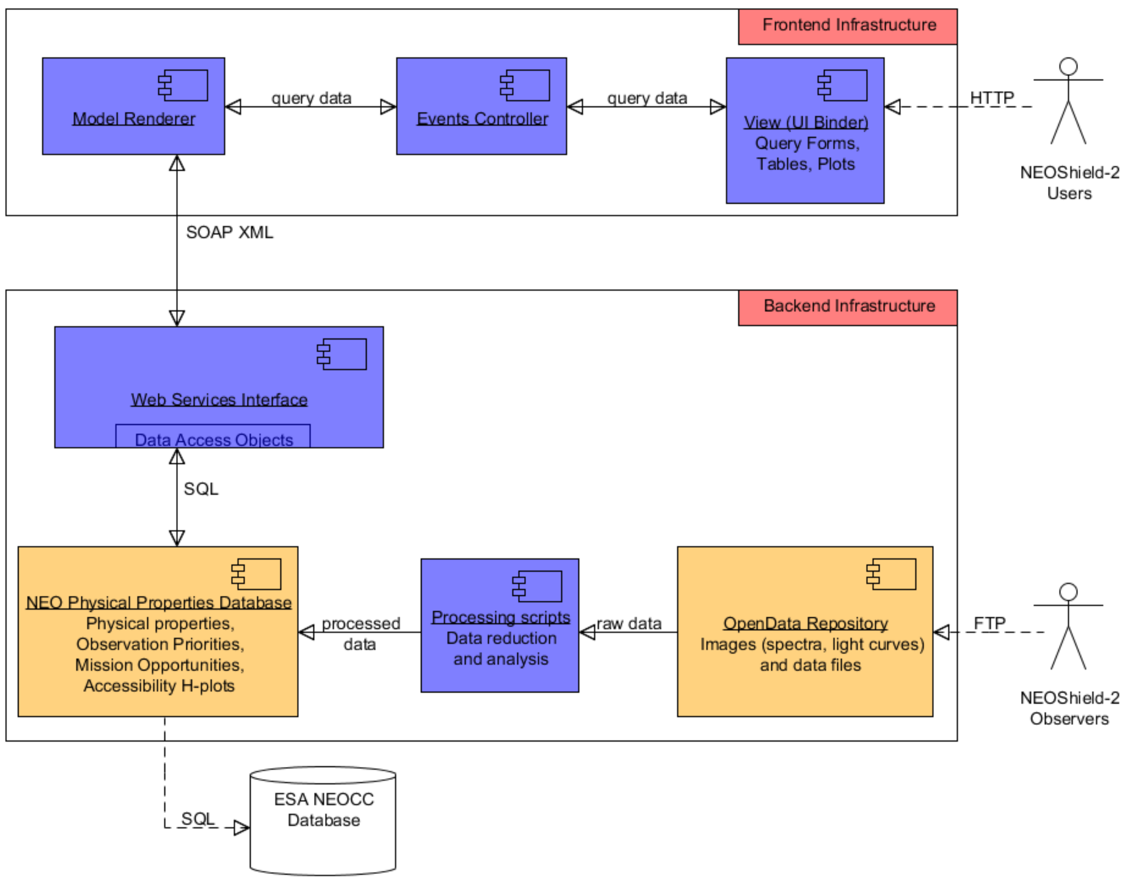


Figure 7: NEO Physical Properties Database and Dynamical Web Interface components diagram

The dataflow within the system is as follows:

1. Observers upload their raw data which is composed of Images (spectra and light curves) and data files in two-column format to the OpenData Repository using FTP.
2. This raw data is processed by a set of offline scripts that insert the corresponding physical properties and final products into the NEO Physical Properties Database.
3. Once the products are available, NEOShield-2 end users can access the Dynamical Web Interface via HTTP and perform queries using its graphical interface.
4. The queries are processed by the Model Renderer and sent via SOAP over HTTP to the backend interface that translates them into SQL statements which are executed in the NEO Physical Properties Database.
5. Query results are propagated back to the interfaces and the Model Renderer that using the UI Builder presents the user the proper tables and plots.

In the following sections it is detailed the design approach of the different system components.

3.1 Data Model

3.1.1 OpenData Repository

The OpenData Repository is an FTP server that provides public access to NEOShield-2 observers so they can upload raw data composed of images (spectra and light curves) and data files (in two-column format). The proposed structure is the one described in figure below:

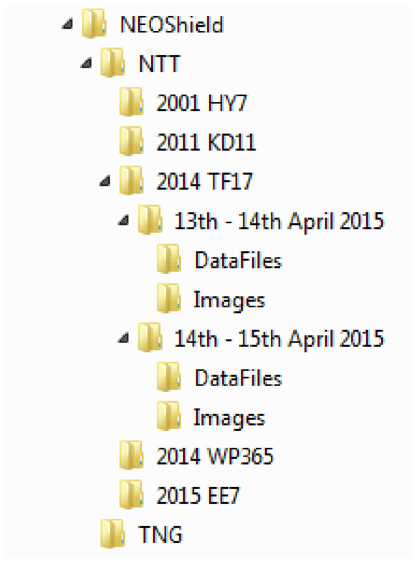


Figure 8: OpenData Repository proposed top-level structure

The logic behind this structure is the following one:

- There will be a root folder called ‘NEOShield’ that shall contain one folder per telescope, like for example “NTT”, “TNG” etc…
- Inside each telescope there will be the list of objects observed by that telescope, one folder per object.
- For each object observed there shall be one entry per observation date range and inside those observations campaigns folders the observers shall upload their images and ASCII data files in the dedicated folders.

3.1.1.1 Procedure to connect remotely to the OpenData Repository and upload files

The following steps must be executed to connect to the OpenData repository from any Internet location and upload images and data files associated to a given object:

1. Open a web browser and access DEIMOS VPN secure page:

   https://vpn.deimos-space.com

2. Once the page is loaded, use the NEOShield OpenData Repository credentials for public access:

   ```
   Name: [vpn_name]
   Password: [vpn_passwd]
   ```

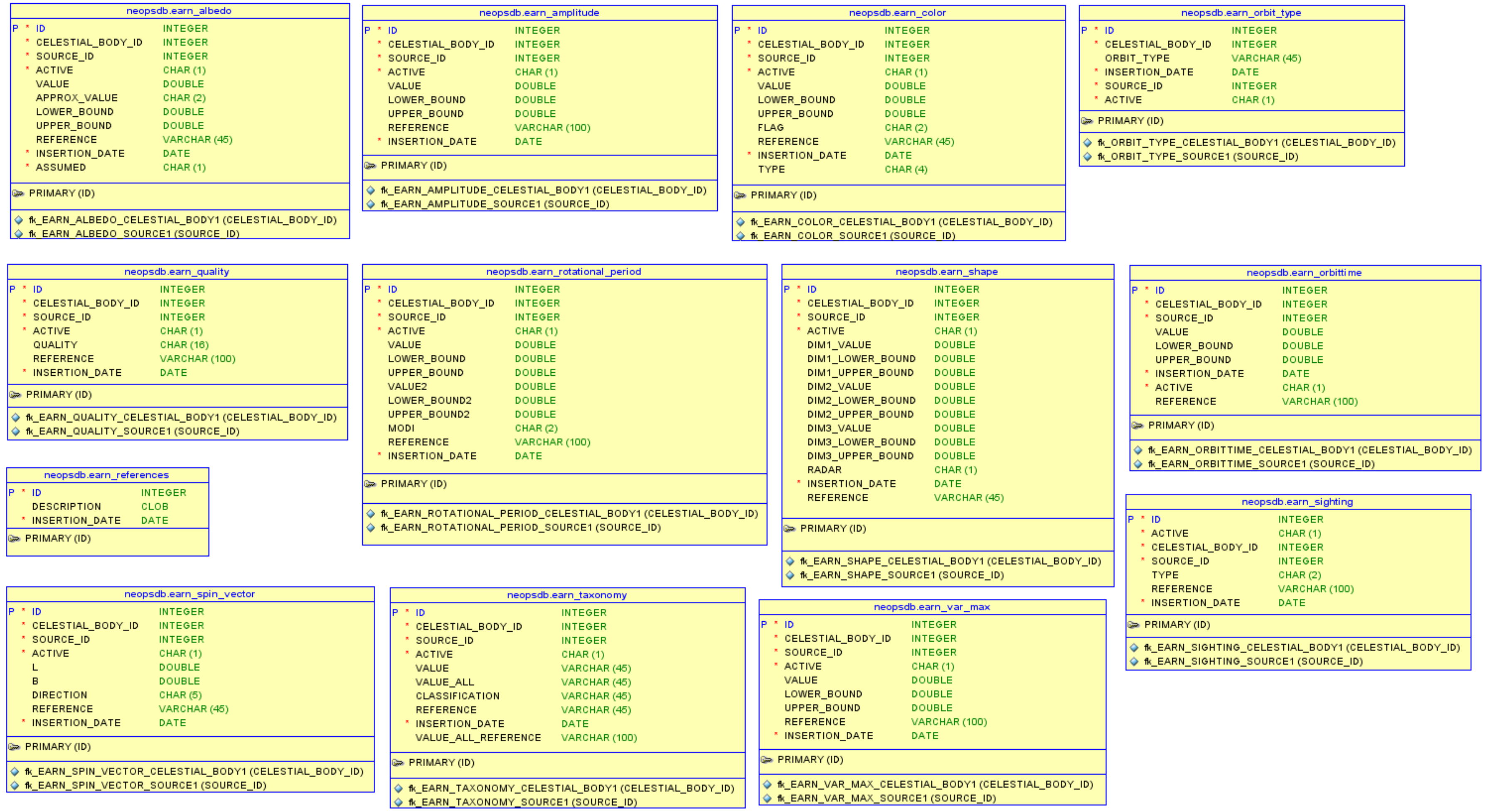
3. Once the user is logged in, he/she shall be prompted to install the VPN web plugin to access NEOShield-2 Backend secured infrastructure.
4. Re-start the web browser and repeat the login process, once in the user shall press the button “Connect”.
5. If the connection goes well then the secured access is granted and the user can access the OpenData Repository using a SFTP client tool with the following connection details:

   ```
   Host: [server-ip]
   Protocol: SFTP (port 22)
   User: [username]
   Pass: [passwd]
   ```

6. The structure of the repository shall be the same as described in Figure 6, if the user wants to upload data from a telescope which has no folder then he/she can create it and inside it create the proper object and date range folders for storing the files according to the convention defined in Figure 6.

3.1.2 NEO Physical Properties Database

The NEO Physical Properties Database shall store a consolidated version of the characteristics from objects observed during NEOShield-2 campaigns. Figure 7 shows the proposed table structure for the NEO Physical Properties Database in terms of fields and data types:



	neopsdb.earn_albedo	
P *	ID	INTEGER
*	CELESTIAL_BODY_ID	INTEGER
*	SOURCE_ID	INTEGER
*	ACTIVE	CHAR (1)
	VALUE	DOUBLE
	APPROX_VALUE	CHAR (2)
	LOWER_BOUND	DOUBLE
	UPPER_BOUND	DOUBLE
	REFERENCE	VARCHAR (45)
*	INSERTION_DATE	DATE
*	ASSUMED	CHAR (1)

PRIMARY (ID)

fk_EARN_ALBEDO_CELESTIAL_BODY1 (CELESTIAL_BODY_ID)
fk_EARN_ALBEDO_SOURCE1 (SOURCE_ID)

	neopsdb.earn_amplitude	
P *	ID	INTEGER
*	CELESTIAL_BODY_ID	INTEGER
*	SOURCE_ID	INTEGER
*	ACTIVE	CHAR (1)
	VALUE	DOUBLE
	LOWER_BOUND	DOUBLE
	UPPER_BOUND	DOUBLE
	REFERENCE	VARCHAR (100)
*	INSERTION_DATE	DATE

PRIMARY (ID)

fk_EARN_AMPLITUDE_CELESTIAL_BODY1 (CELESTIAL_BODY_ID)
fk_EARN_AMPLITUDE_SOURCE1 (SOURCE_ID)

	neopsdb.earn_color	
P *	ID	INTEGER
*	CELESTIAL_BODY_ID	INTEGER
*	SOURCE_ID	INTEGER
*	ACTIVE	CHAR (1)
	VALUE	DOUBLE
	LOWER_BOUND	DOUBLE
	UPPER_BOUND	DOUBLE
	FLAG	CHAR (2)
	REFERENCE	VARCHAR (45)
*	INSERTION_DATE	DATE
	TYPE	CHAR (4)

PRIMARY (ID)

fk_EARN_COLOR_CELESTIAL_BODY1 (CELESTIAL_BODY_ID)
fk_EARN_COLOR_SOURCE1 (SOURCE_ID)

	neopsdb.earn_orbit_type	
P *	ID	INTEGER
*	CELESTIAL_BODY_ID	INTEGER
	ORBIT_TYPE	VARCHAR (45)
*	INSERTION_DATE	DATE
*	SOURCE_ID	INTEGER
*	ACTIVE	CHAR (1)

PRIMARY (ID)

fk_ORBIT_TYPE_CELESTIAL_BODY1 (CELESTIAL_BODY_ID)
fk_ORBIT_TYPE_SOURCE1 (SOURCE_ID)

	neopsdb.earn_quality	
P *	ID	INTEGER
*	CELESTIAL_BODY_ID	INTEGER
*	SOURCE_ID	INTEGER
*	ACTIVE	CHAR (1)
	QUALITY	CHAR (16)
	REFERENCE	VARCHAR (100)
*	INSERTION_DATE	DATE

PRIMARY (ID)

fk_EARN_QUALITY_CELESTIAL_BODY1 (CELESTIAL_BODY_ID)
fk_EARN_QUALITY_SOURCE1 (SOURCE_ID)

	neopsdb.earn_rotational_period	
P *	ID	INTEGER
*	CELESTIAL_BODY_ID	INTEGER
*	SOURCE_ID	INTEGER
*	ACTIVE	CHAR (1)
	VALUE	DOUBLE
	LOWER_BOUND	DOUBLE
	UPPER_BOUND	DOUBLE
	VALUE2	DOUBLE
	LOWER_BOUND2	DOUBLE
	UPPER_BOUND2	DOUBLE
	MODI	CHAR (2)
	REFERENCE	VARCHAR (100)
*	INSERTION_DATE	DATE

PRIMARY (ID)

fk_EARN_ROTATIONAL_PERIOD_CELESTIAL_BODY1 (CELESTIAL_BODY_ID)
fk_EARN_ROTATIONAL_PERIOD_SOURCE1 (SOURCE_ID)

	neopsdb.earn_shape	
P *	ID	INTEGER
*	CELESTIAL_BODY_ID	INTEGER
*	SOURCE_ID	INTEGER
*	ACTIVE	CHAR (1)
	DIM1_VALUE	DOUBLE
	DIM1_LOWER_BOUND	DOUBLE
	DIM1_UPPER_BOUND	DOUBLE
	DIM2_VALUE	DOUBLE
	DIM2_LOWER_BOUND	DOUBLE
	DIM2_UPPER_BOUND	DOUBLE
	DIM3_VALUE	DOUBLE
	DIM3_LOWER_BOUND	DOUBLE
	DIM3_UPPER_BOUND	DOUBLE
	RADAR	CHAR (1)
*	INSERTION_DATE	DATE
	REFERENCE	VARCHAR (45)

PRIMARY (ID)

fk_EARN_SHAPE_CELESTIAL_BODY1 (CELESTIAL_BODY_ID)
fk_EARN_SHAPE_SOURCE1 (SOURCE_ID)

	neopsdb.earn_orbittime	
P *	ID	INTEGER
*	CELESTIAL_BODY_ID	INTEGER
*	SOURCE_ID	INTEGER
	VALUE	DOUBLE
	LOWER_BOUND	DOUBLE
	UPPER_BOUND	DOUBLE
*	INSERTION_DATE	DATE
*	ACTIVE	CHAR (1)
	REFERENCE	VARCHAR (100)

PRIMARY (ID)

fk_EARN_ORBITTIME_CELESTIAL_BODY1 (CELESTIAL_BODY_ID)
fk_EARN_ORBITTIME_SOURCE1 (SOURCE_ID)

	neopsdb.earn_references	
P *	ID	INTEGER
	DESCRIPTION	CLOB
*	INSERTION_DATE	DATE

PRIMARY (ID)

	neopsdb.earn_sighting	
P *	ID	INTEGER
*	ACTIVE	CHAR (1)
*	CELESTIAL_BODY_ID	INTEGER
*	SOURCE_ID	INTEGER
	TYPE	CHAR (2)
	REFERENCE	VARCHAR (100)
*	INSERTION_DATE	DATE

PRIMARY (ID)

fk_EARN_SIGHTING_CELESTIAL_BODY1 (CELESTIAL_BODY_ID)
fk_EARN_SIGHTING_SOURCE1 (SOURCE_ID)

	neopsdb.earn_spin_vector	
P *	ID	INTEGER
*	CELESTIAL_BODY_ID	INTEGER
*	SOURCE_ID	INTEGER
*	ACTIVE	CHAR (1)
	L	DOUBLE
	B	DOUBLE
	DIRECTION	CHAR (5)
	REFERENCE	VARCHAR (45)
*	INSERTION_DATE	DATE

PRIMARY (ID)

fk_EARN_SPIN_VECTOR_CELESTIAL_BODY1 (CELESTIAL_BODY_ID)
fk_EARN_SPIN_VECTOR_SOURCE1 (SOURCE_ID)

	neopsdb.earn_taxonomy	
P *	ID	INTEGER
*	CELESTIAL_BODY_ID	INTEGER
*	SOURCE_ID	INTEGER
	ACTIVE	CHAR (1)
	VALUE	VARCHAR (45)
	VALUE_ALL	VARCHAR (45)
	CLASSIFICATION	VARCHAR (45)
	REFERENCE	VARCHAR (45)
*	INSERTION_DATE	DATE
	VALUE_ALL_REFERENCE	VARCHAR (100)

PRIMARY (ID)

fk_EARN_TAXONOMY_CELESTIAL_BODY1 (CELESTIAL_BODY_ID)
fk_EARN_TAXONOMY_SOURCE1 (SOURCE_ID)

	neopsdb.earn_var_max	
P *	ID	INTEGER
*	CELESTIAL_BODY_ID	INTEGER
*	SOURCE_ID	INTEGER
*	ACTIVE	CHAR (1)
	VALUE	DOUBLE
	LOWER_BOUND	DOUBLE
	UPPER_BOUND	DOUBLE
	REFERENCE	VARCHAR (100)
*	INSERTION_DATE	DATE

PRIMARY (ID)

fk_EARN_VAR_MAX_CELESTIAL_BODY1 (CELESTIAL_BODY_ID)
fk_EARN_VAR_MAX_SOURCE1 (SOURCE_ID)

Figure 9: NEO Physical Properties Database proposed structure

3.2 SOAP Middleware Interface

3.2.1 JAX-WS SOAP API usage

The business layer providing the lower level access to the NEO Physical Properties database and generation of the information to be shown in the Dynamical Web Interface shall be composed of a set of Java web services that process input queries coming from the Dynamical Web Interface via SOAP over HTTP and translate them into SQL statements which are executed in the NEO Physical Properties Database.These web services are implemented using JAX-WS API which is a technology for building web services and clients that communicate using XML. JAX-WS allows developers to write message-oriented as well as RPC-oriented web services.

In JAX-WS, a web service operation invocation is represented by an XML-based protocol such as SOAP. The SOAP specification defines the envelope structure, encoding rules, and conventions for representing web service invocations and responses which are exposed through WSDL contracts (see [STD-2]). These calls and responses are transmitted as SOAP messages (XML files) over HTTP. Although SOAP messages are complex, the JAX-WS API hides this complexity from the application developer.

On the server side, the developer specifies the web service operations by defining methods in an interface written in the Java programming language. The developer also codes one or more classes that implement those methods. Client programs are also easy to code. A client creates a proxy (a local object representing the service) and then simply invokes methods on the proxy. With JAX-WS, the developer does not generate or parse SOAP messages. It is the JAX-WS runtime system that converts the API calls and responses to and from SOAP messages.

The corresponding WSDL for each service exposed by the SOAP Middleware Interface shall provide a machine-readable description of how the service can be called, what parameters it expects and what data structures it returns. Also WSDLs shall be validated against WS-I Basic Profiles.

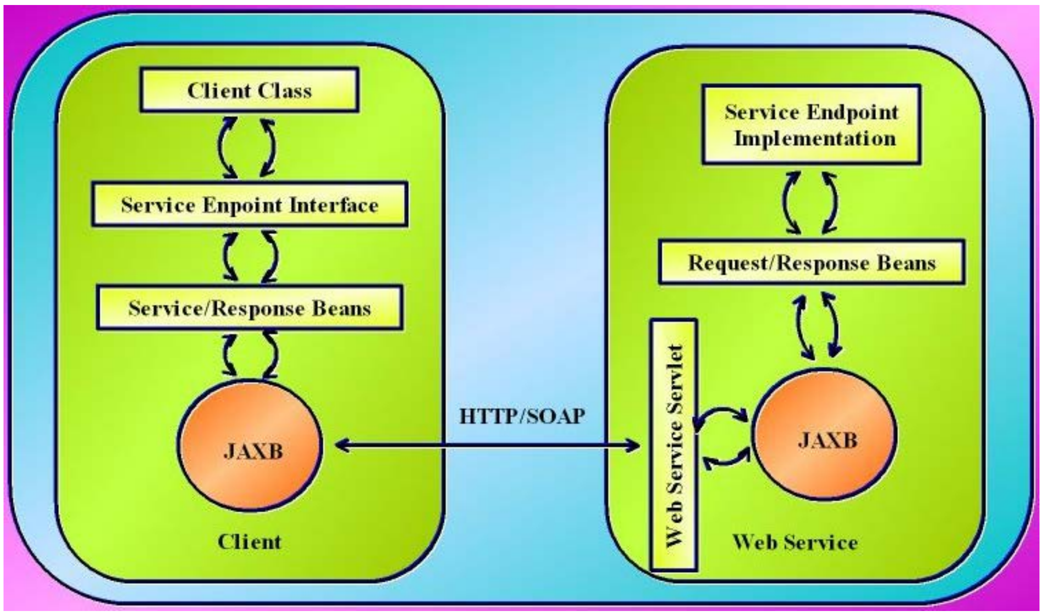


Figure 10: JAX-WS API work logic

3.2.2 Data Access Object usage

Access to persistent storage, such as to a database, varies greatly depending on the type of storage (relational databases, object-oriented databases, flat files, and so forth) and the vendor implementation, for that reason the SOAP Middleware Interface shall implement the Data Access Object (DAO) pattern for accessing to abstract and encapsulate all access to the the NEO Physical Properties Database, working as an adapter between the component and the data source.

The business component that relies on the DAO uses the simpler interface exposed by the DAO for its clients. The DAO completely hides the data source implementation details from its clients which communicate via the Transfer Object. Because the interface exposed by the DAO to clients does not change when the underlying data source implementation changes, this pattern allows the DAO to adapt to different storage schemes without affecting its clients or business components.

Figure below shows the implementation of this pattern for the use case of the Celestial Body entity which has been characterized just with the “designator” atribute for simplicity:

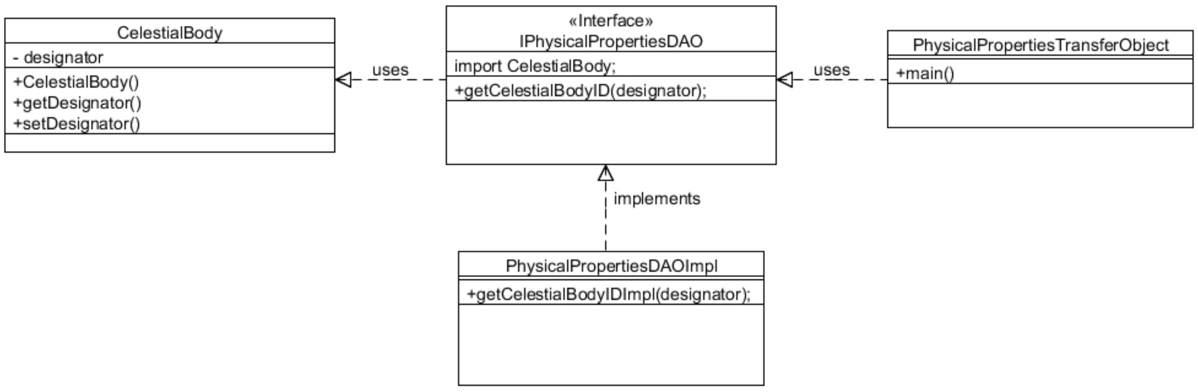


Figure 11: Data Access Object (DAO) pattern implementation

1. The “CelestialBody” JavaBean entity contains the definition of all celestial bodies’ attributes (like the “designator” field). These entity beans are transactional objects representing persistent data and expose the values of attributes by providing accessor methods (setters and getters) for each attribute it wishes to expose.
2. The interface class “IPhysicalPropertiesDAO” encapsulate the signature methods for accessing the different Physical Properties JavaBean classes, like the “CelestialBody” JavaBean.
3. The “PhysicalPropertiesDAOImpl” class implements the methods provided in the interface with the proper business logic inside.
4. The “PhysicalPropertiesTransferObject” class is used to encapsulate the business data coming from the client. A single method call is used to send and retrieve the Transfer Object. When the client requests the enterprise bean for the business data, the enterprise bean can construct the Transfer Object, populate it with its attribute values, and pass it by value to the client.

3.3 Dynamical Web Interface

For details about the Dynamical Web Interface architectural design sees [RD8].

3.4 Architectural Open Points

The following issues remain open in the design of the NEO Physical Properties Database:

Open Point ID	Description
[OP-1]	A clear data policy must still be established (e.g. before observations reduction and analysis no data can be made publicly available).

Open Point ID	Description
[OP-2]	The number of objects to be observed during the NEOShield-2 project must be addressed; We have made a rough estimation of 500 objects.
[OP-3]	It has to be clarified what are the different data types that you expect us to store (e.g. images, photometry files, spectroscopy, plots etc.) and the corresponding size for each object observed; We have estimated one image and a couple of ASCII files per object.
[OP-4]	It has to be clarified which facilities are likely to carry out most of the observations (e.g. NTT, TNG etc).
[OP-5]	Regarding the OpenData Repository, clarify if there are any preferences in organizing the data for subsequent retrieval; One single public user for all observers or different dedicated users?
[OP-6]	It has to be clarified if there is any baseline Operating System to be used across the project. SUSE Linux Enterprise Server 11 SP1 (x86_64) is the selected option currently installed @DEIMOS

Table 1: Architectural open points in the implementation of the NEO Physical Properties Database

4 Implementation Technologies

The following table shows the third party products used in the implementation of the NEO Physical Properties Database and Dynamical Web Interface and the corresponding IPR information.

Product	Vendor Name	Version and License
SUSE Linux Enterprise (operating system)	SUSE Linux (http://www.suse.com/)	SLES11 Service Pack 1 x86_64 (kernel release: 2.6.32.19-0.3) License at http://www.novell.com/company/legal/
Java JDK/JRE (J2EE VM)	Oracle Corporation (http://www.oracle.com/)	v1.7.0_60 SCSL (http://www.oracle.com/technetwork/java/scsl-1-1-149938.txt)
MySQL Database (realtional database)	Oracle Corporation (https://www.mysql.com/)	v5.5.24 SCSL (http://www.oracle.com/technetwork/java/scsl-1-1-149938.txt)
Apache Tomcat (J2EE container)	Apache Software Foundation (http://tomcat.apache.org/download-70.cgi)	v7.0.62 Apache License v2.0 (http://www.apache.org/licenses/LICENSE-2.0.html)
Apache Maven (automation tool)	Apache Software Foundation (http://ant.apache.org/)	v3.2.1 Apache License v2.0 (http://www.apache.org/licenses/LICENSE-2.0.html)
JAX-WS API (SOAP web services)	Java Net (http://jax-ws.java.net/)	v2.2.5 Apache License v2.0 (http://www.apache.org/licenses/LICENSE-2.0.html)
Hibernate (database interface API)	RedHat Corporation http://www.hibernate.org/	v4.0 GNU Lesser Public License v2.1 (https://www.gnu.org/licenses/old-licenses/lgpl-2.1.en.html)
SoapUI (SOAP web services testing tool)	Smart Bear Software (http://www.soapui.org/)	v5.2.0 Soap UI License (http://www.soapui.org/Developers-Corner/soapui-license.html)
Eclipse IDE for J2EE (development environment)	Eclipse Foundation (http://www.eclipse.org/downloads/packages/ecli	v4.4 SR2 (Luna) Eclipse Public License v1.0 (http://www.eclipse.org/legal/epl-v10.html)

Product	Vendor Name	Version and License
	pse-ide-java-ee-developers/lunasr2)	

Table 2: Third party products used in implementation of the NEO Physical Properties Database

Appendix A: Installation procedure

This appendix describes procedure to install the NEO Physical Properties Database and Dynamical Web Interface.

4.1 Installation Pre-requisites

This section details the Third-Party components that should be present prior to installing the NEO Physical Properties Database and Dynamical Web Interface in the target (development or operational) machine. The 'SRC_ROOT' path refers to the path in the NEOShield-2 DEIMOS 'trunk' code repository where the selected items are located.

4.1.1 Operating System

The reference Operating System for the project is SuSE Linux Enterprise Server v11 SP1. For checking the operating system, execute the following command on a terminal:

```
$ cat /etc/issue
Welcome to SUSE Linux Enterprise Server 11 SP1 (x86_64) - Kernel \r (\l).
```

4.1.2 Java VM

The Java reference version to be used is JRE 7; for installing the Java virtual machine execute the following commands:

1. Copy the installation bundle and unpack it in the desired location:

```
$ cp SRC_ROOT/COTS/Java/jdk-7u60-linux-x64.tar $HOME
$ cd
$ tar xfvz jdk-7u60-linux-x64.tar
```

2. To be able to run Java from any location in the build platform we must add the path to Java executable binary to the $PATH variable:

```
$ export JAVA_HOME=$HOME/jdk1.7.0_60/bin
$ export PATH=.:$JAVA_HOME/bin:$PATH
```

3. For checking the Java version installed run:

```
$ java -version
java version "1.7.0_60"
Java(TM) SE Runtime Environment (build 1.7.0_60-b19)
Java HotSpot(TM) 64-Bit Server VM (build 24.60-b09, mixed mode)
```

4.1.3 Apache Maven

For building the Dynamical Web Interface SOAP services we have used Maven automation tool, to install Maven follow these steps:

1. Copy the installation bundle and unpack it in the desired location:

```
$ cp SRC_ROOT/COTS/Maven/apache-maven-3.2.1-bin.tar.gz $HOME
$ cd
$ tar xfvz apache-maven-3.2.1-bin.tar.gz
```

2. To be able to run Maven from any location in the build platform we must add the path to Maven executable binary to the $PATH variable:

```
$ export MAVEN_HOME=$HOME/apache-maven-3.2.1
$ export PATH=.:$MAVEN_HOME/bin:$PATH
```

3. Maven repository dependencies are available in the '$HOME/.m2/repository' folder.
4. Notes on Maven execution flags:

- If we need to include debugging details during compilation then we can run the same Maven command but adding the '-X' flag. In this installation procedure we have not included this flag for the shake of simplicity.
- If we need to make an offline built of the services then we can run the same Maven command but adding the '-o' flag, Maven will not attempt to connect to any remote repository to retrieve artifacts and shall get the necessary dependencies from the local repository available in the '.m2/repository' folder. In this installation procedure we have included this flag to force the offline compilation mode.

4.1.4 MySQL Relational Database

This procedure installs the MySQL Server 5.6 using RPM binary bundles:

1. Copy the installation bundles to the deployment server:

```
$ cp SRC_ROOT/COTS/MySQL/MySQL-server-5.6.23-1.sles11.x86_64.rpm $HOME
$ cp SRC_ROOT/COTS/MySQL/MySQL-client-5.6.23-1.sles11.x86_64.rpm $HOME
```

2. As 'root' user install the packages:

```
$ sudo rpm -ivh MySQL-server-5.6.23-1.sles11.x86_64.rpm
$ rpm -ivh MySQL-client-5.6.23-1.sles11.x86_64.rpm
```

3. Modify MySQL root user password. You must set the password to the root user of MySQL environment. The password must be "neoroot":

```
$ mysqladmin -u root password [root_passwd]
```

4. Set MySQL as no case sensitive. By default, MySQL Server Linux version is case sensitive. We must change this property to improve server performance. Edit the MySQL config file:

```
$ vi /etc/my.cnf
```

And add the following entries:

```
[mysqld]
lower_case_table_names=1
```

5. Also allow remote connections to the Database by commenting the following line:

```
#bind-address     = 127.0.0.1
```

6. Restart MySQL Service. To do this type:

```
$ sudo /etc/init.d/mysql stop
$ sudo /etc/init.d/mysql start
```

7. Exit the root shell.
8. To connect to the database and display the available databases simply execute:

```
$ mysql -h localhost -u root –p (password=[root_passwd])
mysql> show databases;
mysql> exit;
```

4.1.5 Apache Tomcat

The Apache Tomcat reference version to be used is 7.0.62; for installing Tomcat engine execute the following commands:

1. Copy the installation bundle and unpack it in the desired location:

```
$ cp SRC_ROOT/COTS/Tomcat/apache-tomcat-7.0.62.tar.gz $HOME
$ cd
$ tar xfvz apache-tomcat-7.0.62.tar.gz
```

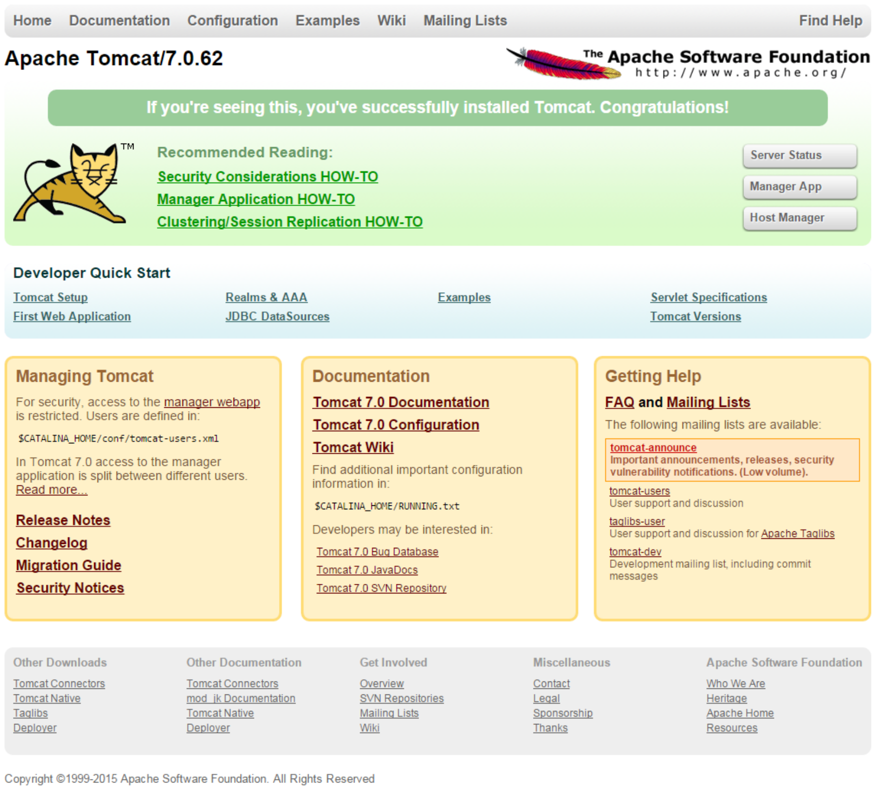
2. To be able to run Tomcat we must add the path to its installation root folder to the $PATH variable:

```
$ export CATALINA_HOME=$HOME/apache-tomcat-7.0.62
$ export PATH=.:$CATALINA_HOME/bin:$PATH
```

3. For starting Tomcat simply execute:

```
$ cd $CATALINA_HOME/bin
$ ./startup
```

4. Tomcat’s welcome page should be now available at the following url:
http://172.23.5.230:8080/


Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/7.0.62

The Apache Software Foundation
http://www.apache.org/

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:
Security Considerations HOW-TO
Manager Application HOW-TO
Clustering/Session Replication HOW-TO

Server Status
Manager App
Host Manager

Developer Quick Start

Tomcat Setup	Realms & AAA	Examples	Servlet Specifications
First Web Application	JDBC DataSources		Tomcat Versions

Managing Tomcat

For security, access to the manager webapp is restricted. Users are defined in:

```
$CATALINA_HOME/conf/tomcat-users.xml
```

In Tomcat 7.0 access to the manager application is split between different users. Read more...

Release Notes
Changelog
Migration Guide
Security Notices

Documentation

Tomcat 7.0 Documentation
Tomcat 7.0 Configuration
Tomcat Wiki

Find additional important configuration information in:

```
$CATALINA_HOME/RUNNING.txt
```

Developers may be interested in:

Tomcat 7.0 Bug Database
Tomcat 7.0 JavaDocs
Tomcat 7.0 SVN Repository

Getting Help

FAQ and Mailing Lists

The following mailing lists are available:

tomcat-announce
Important announcements, releases, security vulnerability notifications. (Low volume).

tomcat-users
User support and discussion

taglibs-user
User support and discussion for Apache Taglibs

tomcat-dev
Development mailing list, including commit messages

Other Downloads	Other Documentation	Get Involved	Miscellaneous	Apache Software Foundation
Tomcat Connectors	Tomcat Connectors	Overview	Contact	Who We Are
Tomcat Native	mod_jk Documentation	SVN Repositories	Legal	Heritage
Taglibs	Tomcat Native	Mailing Lists	Sponsorship	Apache Home
Deployer	Deployer	Wiki	Thanks	Resources

Copyright ©1999-2015 Apache Software Foundation. All Rights Reserved

Figure 12: Apache Tomcat Welcome Page

5. To stop Tomcat simply execute:

```
$ cd $CATALINA_HOME/bin
$ ./shutdown
```

4.2 Build procedure from sources

4.2.1 SOAP Web Services Interface

The SOAP backend services used by the Dynamical Web Interface to connect to the NEO Physical Properties Database can be built using Mavel tool by executing a single 'pom.xml' script following these steps:

```
$ cd SRC_ROOT/WebServices/src
$ mvn clean install –e -o -Dmaven.test.skip=true
```

This script will go and build each service one by one in the proper order and will place the generated binaries ready to be deployed in Weblogic in the following folder:

```
SRC_ROOT/WebServices/dist
```

The distribution folder shall include all the SOA services binaries (*.war or *.ear) ready to be deployed in Tocat J2EE container following steps detailed in section 0 below.

4.2.2 Dynamical Web Interface

4.2.2.1 Eclipse IDE Installation

For building all DPC HMI components we shall use the Eclipse IDE version "Luna" SR2 with SDK 4.4 bundle, the following steps detail the installation in the development server:

1. Get the Eclipse Luna SR2 with SDK 4.4 bundle from the source repository which is available in folder:

```
SRC_ROOT/COTS/Eclipse/eclipse-jee-luna-SR2-linux-gtk-x86_64.tar.gz
```

Note: For Windows platforms get instead the 'eclipse-jee-luna-SR2-win32-x86_64.zip' bundle.

2. Copy the zipped file to your local drive and unpack it in the desired location, in this installation procedure we shall deploy in the user's $HOME folder:

```
$ mkdir $HOME/eclipse_4.4
$ cd $HOME/eclipse_4.4
$ cp SRC_ROOT/COTS/Eclipse/eclipse-jee-luna-SR2-linux-gtk-x86_64.tar.gz .
$ tar xfvz eclipse-jee-luna-SR2-linux-gtk-x86_64.tar.gz
```

The following folder shall be created:

```
$ $HOME/eclipse_4.4/eclipse
```

3. Run the '$HOME/eclipse_4.4/eclipse/eclipse' executable file to launch the tool, if the file has no execution permissions then assign them by doing:

```
$ cd $HOME/eclipse_4.4/eclipse
$ chmod +x eclipse
$ ./eclipse &
```

4. The first time the application is launched it asks to setup the desired workspace location, by default we shall set this parameter to be:

```
$HOME/workspace_4.4
```

Once the application is loaded the "Welcome page" is displayed, we can close it and the tool it is ready to be used as coding editor.

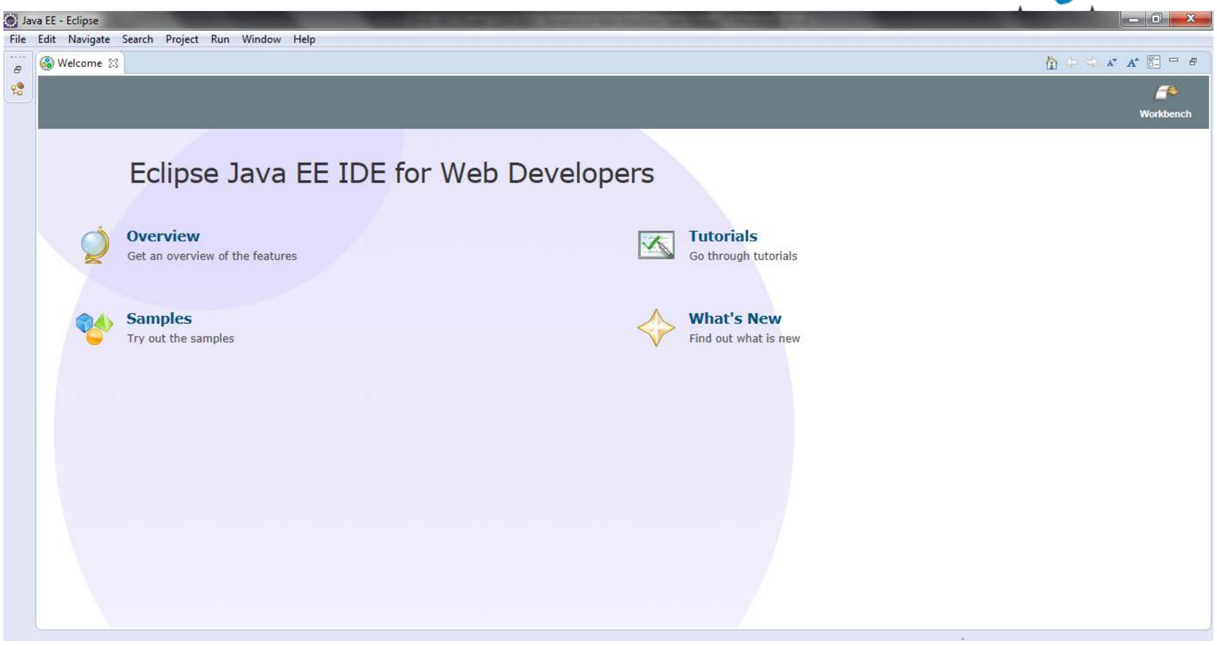


Figure 13: Eclipse Welcome Page

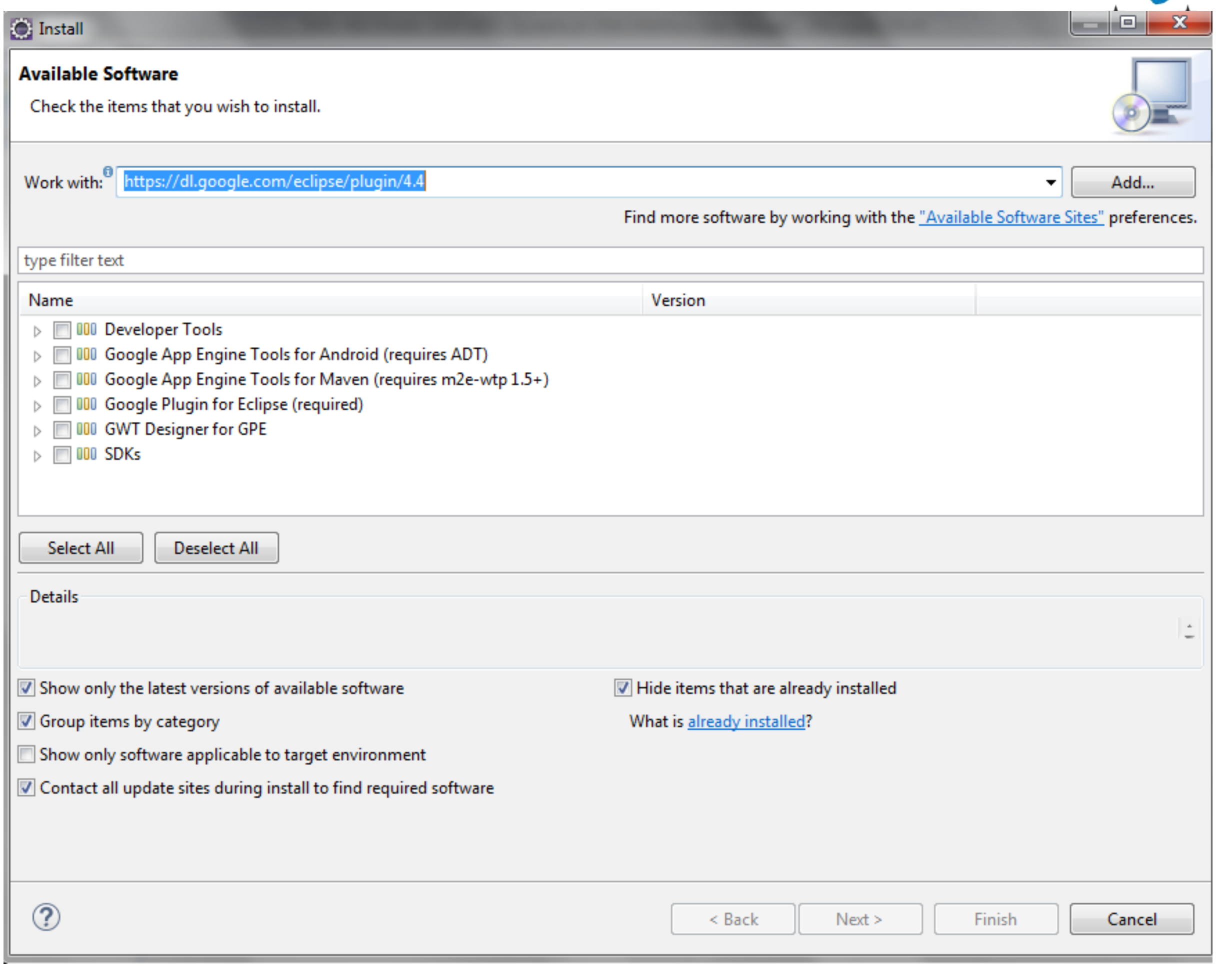
4.2.2.2 Google plugin for Eclipse Installation

To install the Google plugin for Eclipse follow these steps:

1. In Eclipse IDE, select the menu option "Help > Install New Software..." and in the dialog that appears, enter the following update site URL into the 'Work with' text box:

   https://dl.google.com/eclipse/plugin/4.4

2. Press the "Enter" key and then the list of available packages shall be displayed as shown below:


Install

Available Software

Check the items that you wish to install.

Work with: https://dl.google.com/eclipse/plugin/4.4 Add...

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
☐ Developer Tools	
☐ Google App Engine Tools for Android (requires ADT)	
☐ Google App Engine Tools for Maven (requires m2e-wtp 1.5+)	
☐ Google Plugin for Eclipse (required)	
☐ GWT Designer for GPE	
☐ SDKs	

Select All Deselect All

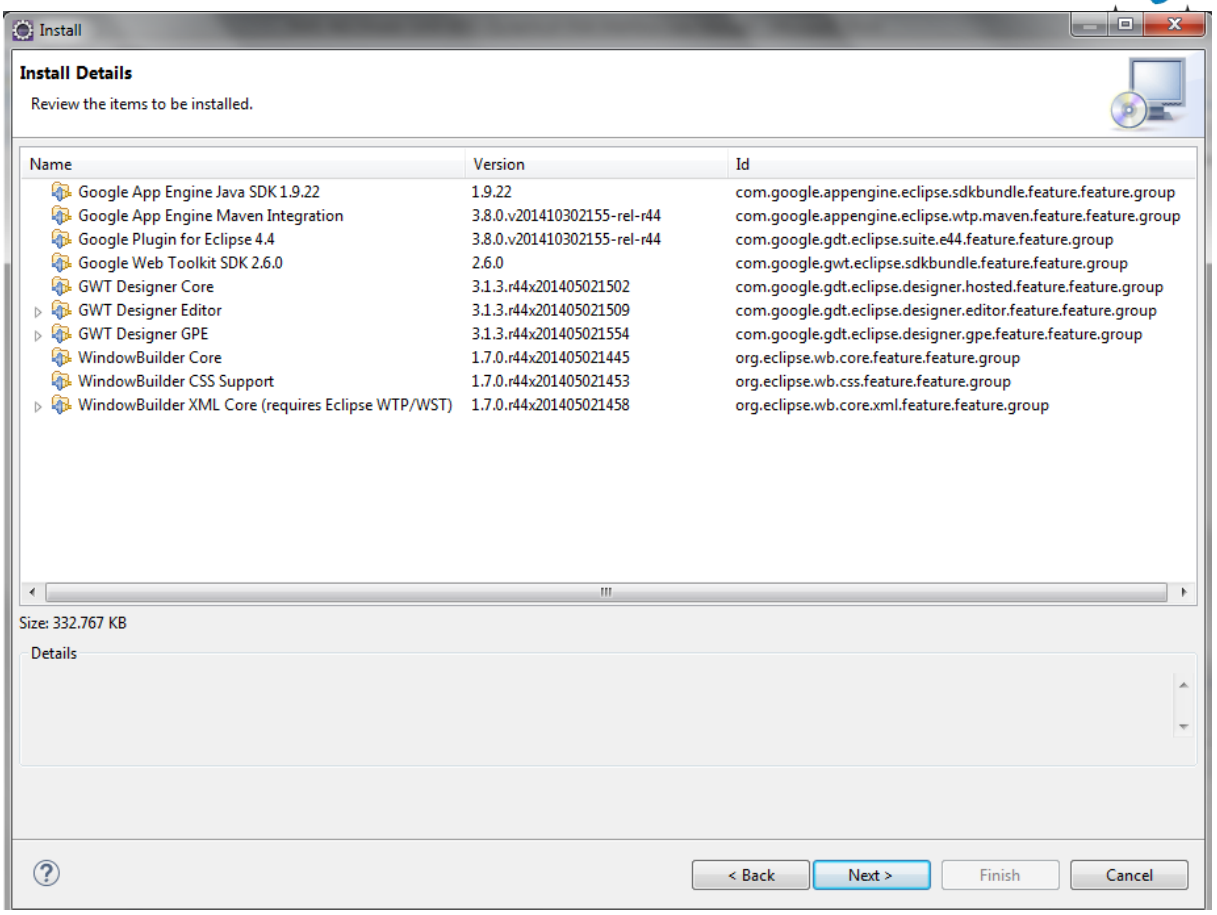
Details

☑ Show only the latest versions of available software
☑ Group items by category
☐ Show only software applicable to target environment
☑ Contact all update sites during install to find required software
☑ Hide items that are already installed
What is already installed?

< Back Next > Finish Cancel

Figure 14: Google plugin for Eclipse installation packages

3. From the list of available packages, we shall install the following ones and click “Next”:
    - Google App Engine Tools for Maven
    - Google Plugin for Eclipse
    - GWT Designer for the Google Plugin for Eclipse
    - Google Web Toolkit SDK
4. Review the features that you are about to install, if everything is OK then click “Next”.


Install

Install Details

Review the items to be installed.

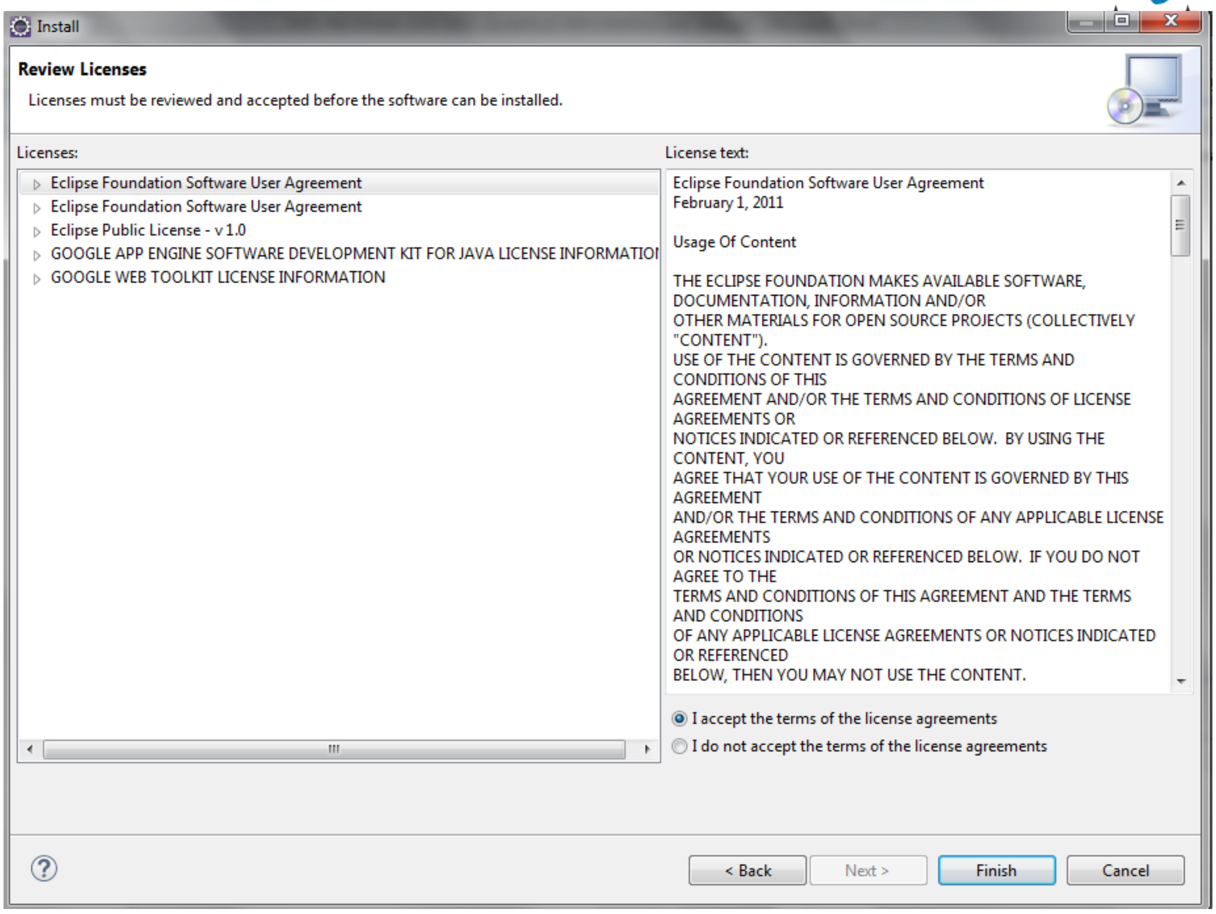
Name	Version	Id
Google App Engine Java SDK 1.9.22	1.9.22	com.google.appengine.eclipse.sdkbundle.feature.feature.group
Google App Engine Maven Integration	3.8.0.v201410302155-rel-r44	com.google.appengine.eclipse.wtp.maven.feature.feature.group
Google Plugin for Eclipse 4.4	3.8.0.v201410302155-rel-r44	com.google.gdt.eclipse.suite.e44.feature.feature.group
Google Web Toolkit SDK 2.6.0	2.6.0	com.google.gwt.eclipse.sdkbundle.feature.feature.group
GWT Designer Core	3.1.3.r44x201405021502	com.google.gdt.eclipse.designer.hosted.feature.feature.group
GWT Designer Editor	3.1.3.r44x201405021509	com.google.gdt.eclipse.designer.editor.feature.feature.group
GWT Designer GPE	3.1.3.r44x201405021554	com.google.gdt.eclipse.designer.gpe.feature.feature.group
WindowBuilder Core	1.7.0.r44x201405021445	org.eclipse.wb.core.feature.feature.group
WindowBuilder CSS Support	1.7.0.r44x201405021453	org.eclipse.wb.css.feature.feature.group
WindowBuilder XML Core (requires Eclipse WTP/WST)	1.7.0.r44x201405021458	org.eclipse.wb.core.xml.feature.feature.group

Size: 332.767 KB

Details

Figure 15: Google plugin for Eclipse installation packages summary

5. Read the license agreements and then select ‘I accept the terms of the license agreements’ and click ‘Finish’ to go on with the installation.


Install
Review Licenses
Licenses must be reviewed and accepted before the software can be installed.
Licenses:
Eclipse Foundation Software User Agreement
Eclipse Foundation Software User Agreement
Eclipse Public License - v 1.0
GOOGLE APP ENGINE SOFTWARE DEVELOPMENT KIT FOR JAVA LICENSE INFORMATIOI
GOOGLE WEB TOOLKIT LICENSE INFORMATION
License text:
< Back
Next >
Finish
Cancel

Eclipse Foundation Software User Agreement
February 1, 2011

Usage Of Content

THE ECLIPSE FOUNDATION MAKES AVAILABLE SOFTWARE, DOCUMENTATION, INFORMATION AND/OR OTHER MATERIALS FOR OPEN SOURCE PROJECTS (COLLECTIVELY "CONTENT"). USE OF THE CONTENT IS GOVERNED BY THE TERMS AND CONDITIONS OF THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. BY USING THE CONTENT, YOU AGREE THAT YOUR USE OF THE CONTENT IS GOVERNED BY THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT AND THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW, THEN YOU MAY NOT USE THE CONTENT.

[✓ I accept the terms of the license agreements] I do not accept the terms of the license agreements

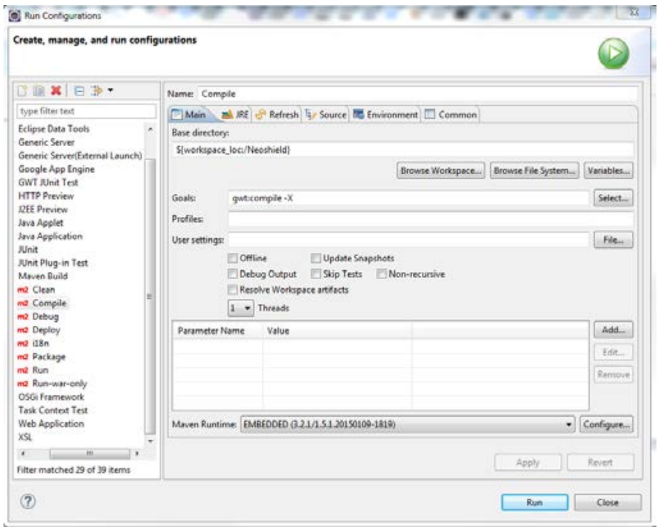
Figure 16: Google plugin for Eclipse installation 'License Agreement' approval

6. After the installation is completed, you will then be asked if you would like to restart Eclipse, click 'Restart Now' to complete the installaiton process.

4.2.2.3 Web Portal deployment

To build the Dynamical Web Interfaces and be able to get the proper *.war file ready to be deployed in Tomcat J2EE container follow these steps:

1. Import the project. In Eclipse IDE, select the menu option "File > Import... > SVN > Project from SVN" and create a new repository location with the url https://sputnik.deimos-space.com/svn/neoshield-2/trunk/WebPortal .
2. Update Project. Right click in project and select "Maven > Update Project…"
3. Clean the project. Select the menu option "Project > Clean...", the option "Project > Build Automatically" is checked.
4. GWT compile. Execute with maven gwt:compile:


Run Configurations

Create, manage, and run configurations

Name: Compile

Base directory: ${workspace_loc:/Neoshield}

Goals: gwt:compile -X

Profiles:

User settings:

Offline, Update Snapshots, Debug Output, Skip Tests, Non-recursive, Resolve Workspace artifacts: unchecked

Threads: 1

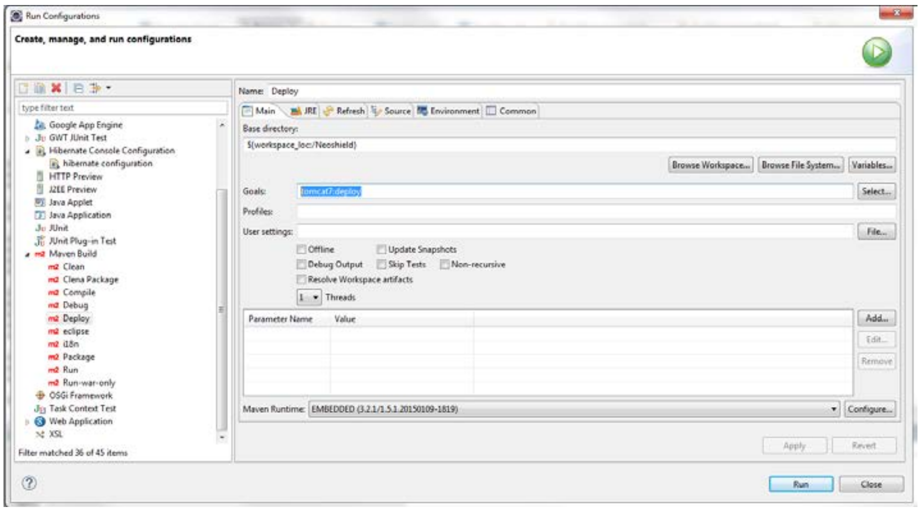
Parameter Name	Value

Maven Runtime: EMBEDDED (3.2.1/1.5.1.20150109-1819)

Filter matched 29 of 39 items

Figure 32: Maven compile

5. Deploy. Execute mvn tomcat7: deploy. A file with an extension *.war will be created in your project inside the target directory and it will deploy it automatically in the remote machine.


Run Configurations

Create, manage, and run configurations

Name: Deploy

Base directory: ${workspace_loc:/Neoshield}

Goals: tomcat7:deploy

Profiles:

User settings:

Offline, Update Snapshots, Debug Output, Skip Tests, Non-recursive, Resolve Workspace artifacts: unchecked

Threads: 1

Parameter Name	Value

Maven Runtime: EMBEDDED (3.2.1/1.5.1.20150109-1819)

Filter matched 36 of 45 items

Figure 33: Maven deploy

4.3 Installation from binaries

4.3.1 NEO Physical Properties Database

The NEO Physical Properties Database shall be stored in the MySQL server deployed within the NEOShield virtual machine described in section 4.1.4. Once the MySQL server is installed, it is necessary to create the 'neoshield' database and their associated tables using the following commands:

```
SRC_ROOT/Database/bin/neoshield_bbdd.sql
```

1. Create 'neoshield' database:

```
$ mysql --user=root --password=[root_passwd]  -e "create database neoshield
character set utf8;"
```

2. Create 'neoshield' tables:

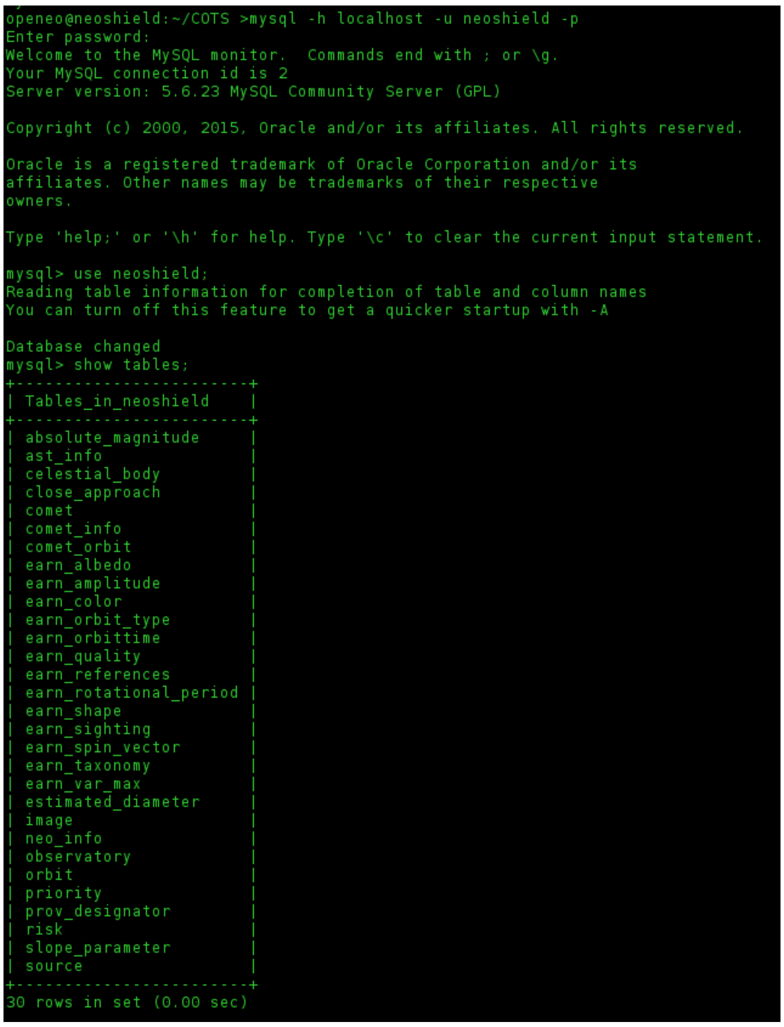
```
$   mysql   --user=root   --password=[root_passwd]   -A   -Dneoshield   <
neoshield_bbdd.sql
```

3. For managing NEOShield tables, we shall create a dedicated user and provide the necessary permissions:

```
$ CREATE USER 'neoshield'@'localhost' IDENTIFIED BY 'neoope';
$ GRANT ALL PRIVILEGES ON * . * TO 'neoshield'@'localhost';
$ FLUSH PRIVILEDGES;
```

4. Connect as 'neoshield' user to the database and display its tables:

```
$ mysql -h localhost -u neoshield –p (password=[neoope_passwd])
mysql> show databases;
mysql> use neoshield;
mysql> show tables;
```



```
openeo@neoshield:~/COTS >mysql -h localhost -u neoshield -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.23 MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use neoshield;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-------------------------+
| Tables_in_neoshield     |
+-------------------------+
| absolute_magnitude      |
| ast_info                |
| celestial_body          |
| close_approach          |
| comet                   |
| comet_info              |
| comet_orbit             |
| earn_albedo             |
| earn_amplitude          |
| earn_color              |
| earn_orbit_type         |
| earn_orbittime          |
| earn_quality            |
| earn_references         |
| earn_rotational_period  |
| earn_shape              |
| earn_sighting           |
| earn_spin_vector        |
| earn_taxonomy           |
| earn_var_max            |
| estimated_diameter      |
| image                   |
| neo_info                |
| observatory             |
| orbit                   |
| priority                |
| prov_designator         |
| risk                    |
| slope_parameter         |
| source                  |
+-------------------------+
30 rows in set (0.00 sec)
```

Figure 17: NEO Physical Properties Database tables' display

4.3.2 SOAP Web Services Interface

Describe here how we can deploy the SOAP web services into Tomcat J2EE container.

[TBC]

4.3.3 Dynamical Web Interface

To deploy the Dynamical Web Interface into Tomcat J2EE container follow these steps:

1. Check apache tomcat is started by executing in the command prompt:

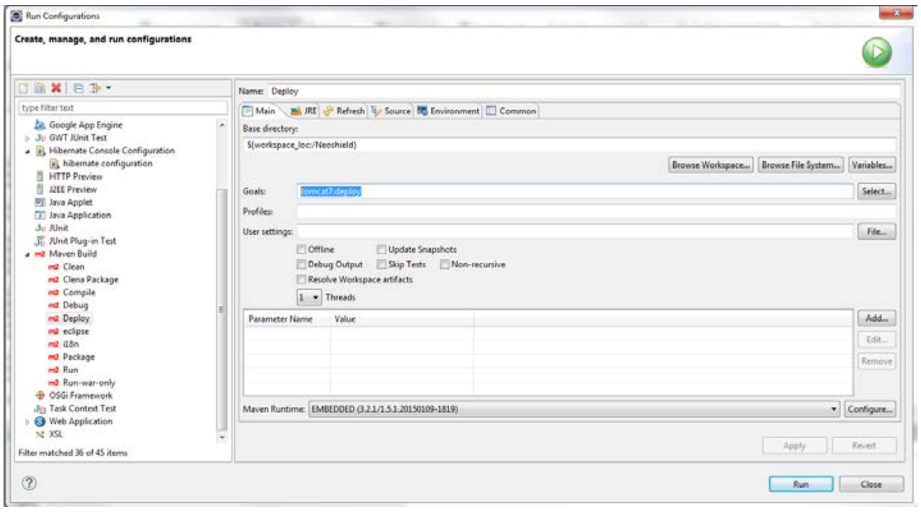
```
$ ps –ef | grep tomcat
```

If the output contains the whole path of the tomcat folder $CATALINA_HOME), then it is running.

2. Start apache tomcat if it is running:

```
$ cd $CATALINA_HOME/bin
$ ./start
```

3. Deploy. Execute mvn tomcat7: deploy. A file with an extension *.war will be created in your project inside the target directory and it will deploy it automatically in the remote machine.



Name: Deploy
Base directory: ${workspace_loc:/Neoshield}
Goals: tomcat7:deploy
Profiles:
User settings:
☐ Offline ☐ Update Snapshots
☐ Debug Output ☐ Skip Tests ☐ Non-recursive
☐ Resolve Workspace artifacts
1 Threads

Parameter Name	Value

Maven Runtime: EMBEDDED (3.2.1/1.5.1.20150109-1819)

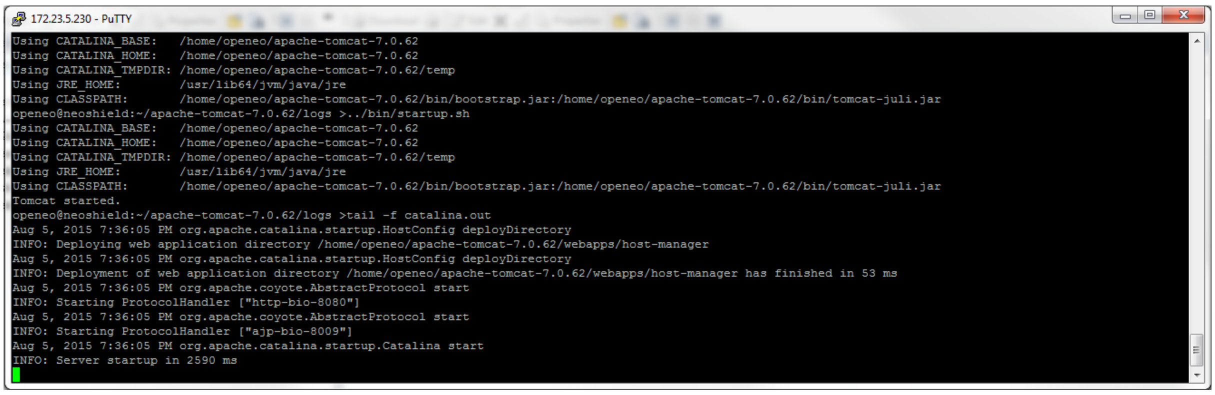
Figure 35: Maven deploy

4. Copy the Neoshield.properties. The Neoshield.properties is located inside the neshield project in the config folder. Copy it in the virtual marchine in the desired path.
5. Edit that path in settings.properties file located in '$CATALINA_HOME/webapps/Neoshield/WEB-INF/classes/com/dms/neoshield2/resources'.

4.4 Confidence Test

To verify the installation has been performed rightly execute in the command prompt:

```
$ cd $CATALINA_HOME/logs
$ tail –f catalina.out
```

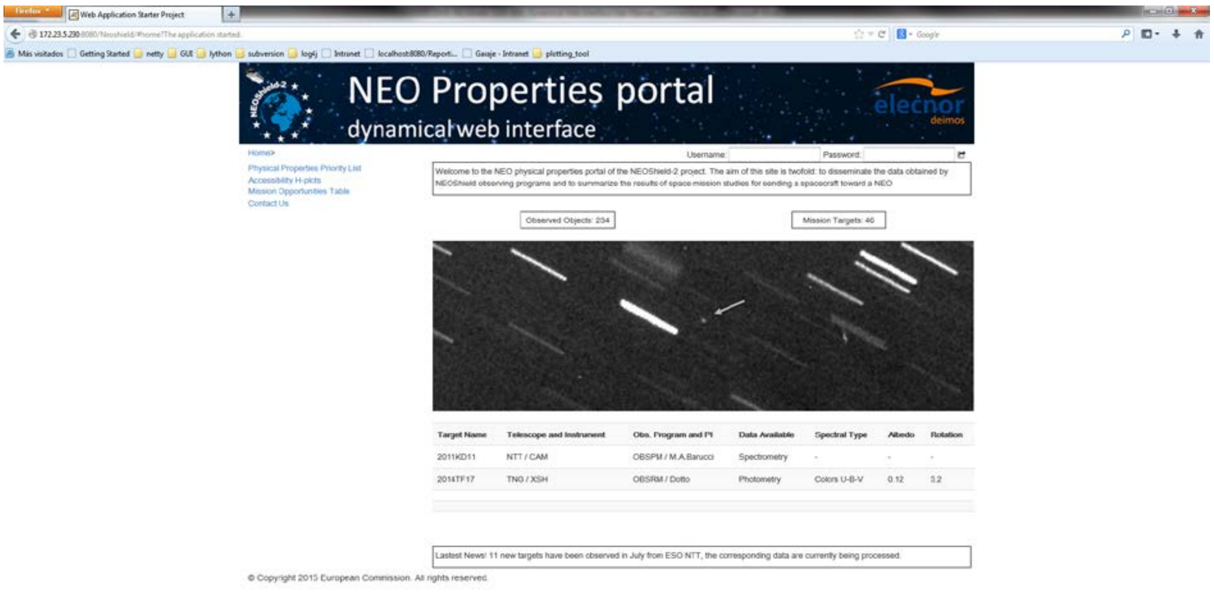


```
Using CATALINA_BASE:   /home/openeo/apache-tomcat-7.0.62
Using CATALINA_HOME:   /home/openeo/apache-tomcat-7.0.62
Using CATALINA_TMPDIR: /home/openeo/apache-tomcat-7.0.62/temp
Using JRE_HOME:        /usr/lib64/jvm/java/jre
Using CLASSPATH:       /home/openeo/apache-tomcat-7.0.62/bin/bootstrap.jar:/home/openeo/apache-tomcat-7.0.62/bin/tomcat-juli.jar
openeo@neoshield:~/apache-tomcat-7.0.62/logs >../bin/startup.sh
Using CATALINA_BASE:   /home/openeo/apache-tomcat-7.0.62
Using CATALINA_HOME:   /home/openeo/apache-tomcat-7.0.62
Using CATALINA_TMPDIR: /home/openeo/apache-tomcat-7.0.62/temp
Using JRE_HOME:        /usr/lib64/jvm/java/jre
Using CLASSPATH:       /home/openeo/apache-tomcat-7.0.62/bin/bootstrap.jar:/home/openeo/apache-tomcat-7.0.62/bin/tomcat-juli.jar
Tomcat started.
openeo@neoshield:~/apache-tomcat-7.0.62/logs >tail -f catalina.out
Aug 5, 2015 7:36:05 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory /home/openeo/apache-tomcat-7.0.62/webapps/host-manager
Aug 5, 2015 7:36:05 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deployment of web application directory /home/openeo/apache-tomcat-7.0.62/webapps/host-manager has finished in 53 ms
Aug 5, 2015 7:36:05 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-8080"]
Aug 5, 2015 7:36:05 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
Aug 5, 2015 7:36:05 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 2590 ms
```

Figure 18: Apache Tomcat message

The user can observe an information message "Server startup in 2590 ms" without exceptions, therefore the application is correctly deployed and the user can start to navigate with the url:

http://172.23.5.230:8080/Neoshield



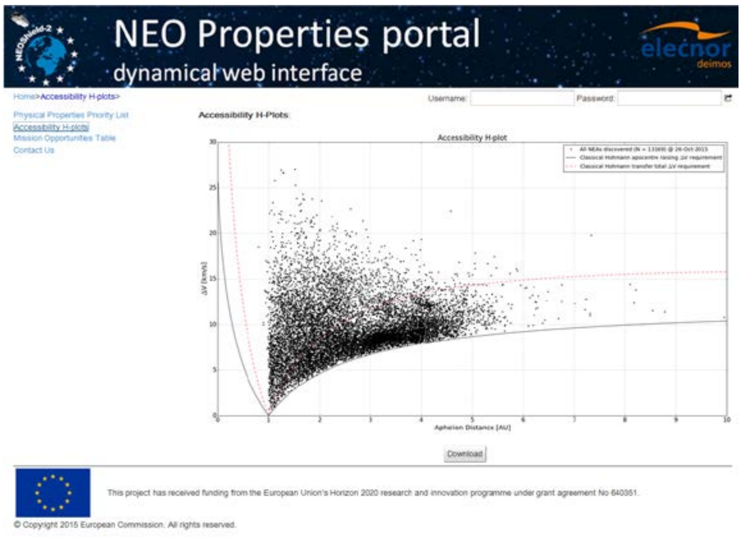
Welcome to the NEO physical properties portal of the NEOShield-2 project. The aim of this site is twofold: to disseminate the data obtained by NEOShield observing programs and to summarize the results of space mission studies for sending a spacecraft toward a NEO

Target Name	Telescope and Instrument	Obs. Program and PI	Data Available	Spectral Type	Albedo	Rotation
2011KD11	NTT / CAM	OBSPM / M.A.Barucci	Spectrometry	-	-	-
2014TF17	TNG / XSH	OBSRM / Dotto	Photometry	Colors U-B-V	0.12	3.2

Lastest News! 11 new targets have been observed in July from ESO NTT, the corresponding data are currently being processed.

© Copyright 2015 European Commission. All rights reserved.

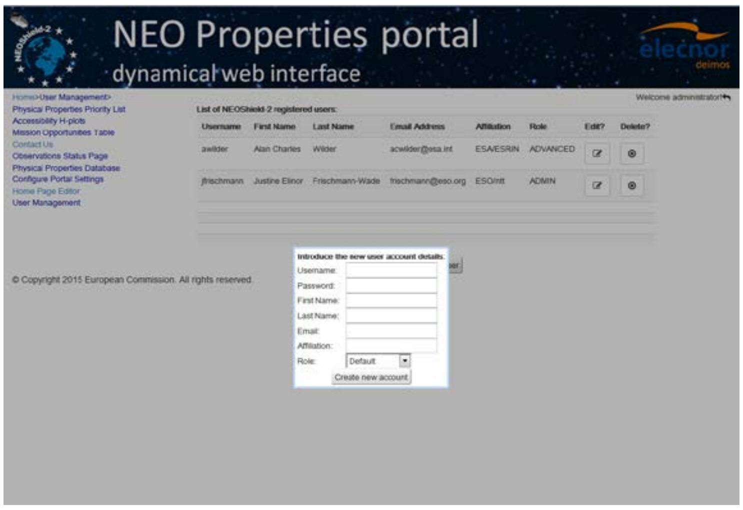
Figure 19: Neoshield Home Page


NEO Properties portal
dynamical web interface
Home>Accessibility H-plots>
Username:
Password:
Physical Properties Priority List
Accessibility H-plots
Mission Opportunities Table
Contact Us
Accessibility H-Plots:
Accessibility H-plot
All NEAs discovered (N = 13169) @ 26-Oct-2015
Classical Hohmann apocentre raising ΔV requirement
Classical Hohmann transfer total ΔV requirement
ΔV [km/s]
Aphelion Distance [AU]
Download

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 640351.

© Copyright 2015 European Commission. All rights reserved.

Figure 20: Accessibility H-plots Page



List of NEOShield-2 registered users:

Username	First Name	Last Name	Email Address	Affiliation	Role	Edit?	Delete?
awilder	Alan Charles	Wilder	acwilder@esa.int	ESA/ESRIN	ADVANCED		
jfrischmann	Justine Elinor	Frischmann-Wade	frischmann@eso.org	ESO/mtt	ADMIN		

Introduce the new user account details:

Username:	
Password:	
First Name:	
Last Name:	
Email:	
Affiliation:	
Role:	Default

Create new account

© Copyright 2015 European Commission. All rights reserved.

Figure 39: User Management Page

End of Document